UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Aleš Jagrič

Aplikacija za porazdeljeno kodiranje pretočnih video vsebin

DIPLOMSKO DELO
VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: izr. prof. dr. Miha Mraz

Ljubljana, 2013

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: 00333/2012
Datum: 04.09.2012

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

Kandidat: **ALEŠ JAGRIČ**

Naslov: **APLIKACIJA ZA PORAZDELJENO KODIRANJE PRETOČNIH VIDEO VSEBIN**

APPLICATION FOR DISTRIBUTED ENCODING OF STREAMING VIDEO

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Kandidat naj v svojem delu predstavi problematiko ponovnega kodiranja (angl. transcoding) video vsebin na zahtevo glede na karakteristike prenosnih medijev do končnih naročniških točk. Pri tem naj izpostavi kodirne metode in njihovo uporabo prikaže na konkretnih zgledih. V nadaljevanju naj kandidat zgradi aplikacijo za vodenje in administriranje postopkov ponovnega kodiranja za potrebe ponudnikov IPTV storitev.

Mentor:

izr. prof. dr. Miha Mraz

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Aleš Jagrič, z vpisno številko 63050047, sem avtor diplomskega dela z naslovom:

Aplikacija za porazdeljeno kodiranje pretočnih video vsebin

S svojim podpisom zagotavljam, da:

- Sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Mihe Mraza.
- 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 21. januarja 2013 Podpis avtorja:

Zahvala

Zahvaljujem se svojemu mentorju izr. prof. dr. Mihi Mrazu za ves vložen trud, pomoč in nasvete pri izdelavi diplomske naloge. Podjetju Fora d.o.o. se zahvaljujem za priskrbljeno strojno opremo, ki je omogočila preizkušanje aplikacije.

Predvsem se zahvaljujem svoji družini, ki me je spodbujala in mi stala ob strani skozi študij in nudila pomoč pri premagovanju ovir.

Najgloblje pa se zahvaljujem Bernardi Grešak in njeni spodbudi pri dokončanju diplomskega dela, brez katere bi še danes le občasno razmišljal, da bi bilo smotrno diplomirati v čim krajšem roku.

Kazalo

Seznam uporabljenih kratic

ACID - Atomicity, Consistency, Isolation, Durability
DTS - Decoding Time Stamp
DVB - Digital Video Broadcasting
FTP - File Transfer Protocol
GNU/GPL- (Gnu's Not Unix) General Public License
GOP- Group Of Pictures
HTTP - Hypertext Transfer Protocol
IP - Internet Protocol address
IPTV - Internet Protocol Television
MAC - Media Access Control
MPEG2 - Moving Picture Experts Group 2
MPTS - Multiple Program Transport Stream
PAT - Program Association Table
PES - Packetized Elementary Stream
PID - Program Identifier
PMT - Program Map Table
PSNR - Peak Signal-to-Noise Ratio
PTS - Presentation Time Stamp
RTSP - Real Time Streaming Protocol
SCP - Secure copy
SFTP - SSH File Transfer Protocol
SMTP - Simple Mail Transfer Protocol
SPTS - Single Program Transport Stream
SQL - Structured Query Language
SSIM - Structural Similarity
SSL/TLS - Secure Sockets Layer / Transport Layer Security
SVN - Subversion
UDP - User Datagram Protocol
URL - Uniform Resource Locator
WORA - Write Once, Run Anywhere
XML - eXtensible Markup Language

Povzetek

V okviru diplomske naloge je bila razvita aplikacija za ponovno (ang. *transcoding*) kodiranje video vsebin v namene oddajanja preko IPTV omrežja. Aplikacija omogoča prenose video vsebin z oddaljenih mest, branje XML dokumentov, ponovno kodiranje v zapisa MPEG2 in h264, prenos vsebin preko protokola SCP, porazdeljevanje dela na več računalnikov ter obveščanje uporabnikov preko e-poštnih sporočil.

Opisali smo kam v velik splet sistemov sodi razvita aplikacija ter orodja in tehnologije uporabljene v okviru razvoja. Predstavili smo postopek preizkušanja video kodirnikov za optimalno kakovost ponovnega kodiranja in videz uporabniškega vmesnika ter postopke, po katerih aplikacija deluje.

Ključne besede:

Java, MPEG2, h264, primerjava kodirnikov, porazdeljevanje dela

Abstract

The purpose of the present thesis is to develop an application, which would transcode video content from different providers to a form appropriate for broadcasting over a IPTV network. Among the capabilities of the application are downloading of remote content, parsing XML files, transcoding to MPEG2 and h264 formats, uploading finished files with the SCP protocol, distributing the workload and notification of users through e-mail.

The thesis describes, where in the great scheme of systems our application is placed. It offers an overview of tools and technologies used during development and the methodology for preizkusing optimal encoder settings and presents a summary description of the user interface along with the main application functionality and database schema.

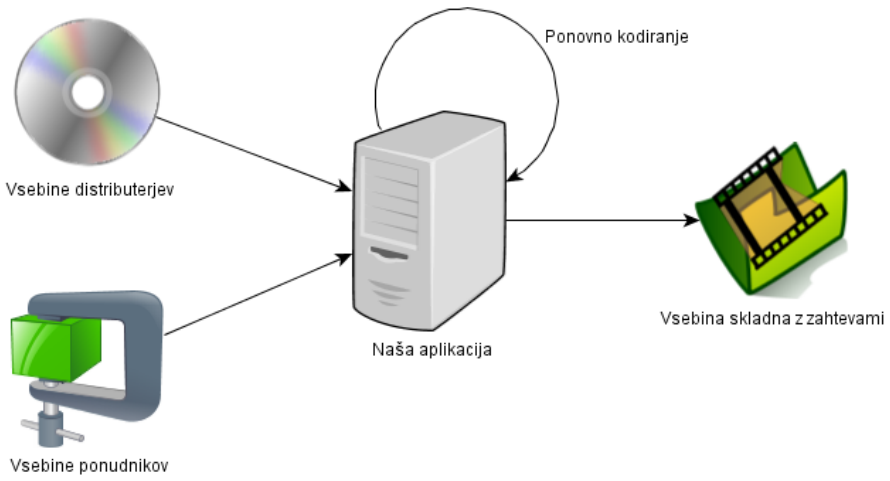
Key words:

Java, MPEG2, h264, codec comparison, workload distribution

1 Uvod

Pri dostavi video vsebin preko IPTV omrežja se ponudniki srečujejo s številnimi oblikami video ter zvočnih zapisov v večpredstavnostnih vsebinah, medtem ko so njihove terminalne naprave pri naročnikih omejene na predvajanje le peščice oblik od teh. Poleg različnih oblik zapisov se srečujejo tudi s problemom prenosa vsebin preko komunikacijskega omrežja, kjer lahko pasovna širina izredno niha glede na kvaliteto medija, ki pogosto ni v lasti operaterja in je zagotavljanje zadostne pasovne širine težavno.

Rešitev problema predstavlja preverjen video zapis, s katerim zagotovimo brezhibno predvajanje vsebin na terminalni opremi, ki z bitno hitrostjo ne presega razpoložljive pasovne širine. Ročno preverjanje skladnosti ter ponovno kodiranje vsebin je dolgotrajen, časovno potraten, monoton postopek, podvržen človeškim napakam.



Slika 1: Ponovno kodiranje vsebin v zapis skladen z zahtevami.

Na sliki 1 je shematsko predstavljen poenostavljen proces prenosa vsebin ponudnikov in distributerjev na računalnik z našo aplikacijo, kjer poteka ponovno kodiranje vsebin v obliko, ki je skladna z zahtevami IPTV omrežja.

Vsebine ponudnikov, ki imajo lastno produkcijo vsebin (na primer HBO ali FOX) in distributerjev, ki ponujajo vsebine tujih ponudnikov (na primer Cenex/Fivia ali Karantanija cinemas), so primarno namenjene predvajanju na DVD in ostalih multimedijskih predvajalnikih. Te vsebine pa so zaradi svoje oblike ter načina zapisa le redko primerne za oddajanje preko IPTV omrežja. Vsebine je zaradi tega potrebno ponovno kodirati v zapis MPEG2 in h264 ter združiti v transportni tok (ang. *transport stream*).

Cilj diplomske naloge je izdelava programske aplikacije, ki delo ponovnega kodiranja opravi s strojno natančnostjo ter hitreje s pomočjo razporejanja dela na več računalnikov. Aplikacija bo sposobna delo po začetni konfiguraciji opravljati popolnoma avtonomno.

V drugem poglavju so opisani namen aplikacije, shema sistema, kamor aplikacija sodi, ter osnovno delovanje dveh instanc aplikacije. Opisan je tudi pregled orodij in tehnologij, uporabljenih pri razvoju aplikacije. V tretjem poglavju diplomske naloge je podrobneje opisan postopek primerjave kakovosti slike ponovno kodiranih vsebin in postopek določanja argumentov kodirnikov, ki najbolje ustrezajo za splošno kodiranje video vsebin v terminalnim napravam ustrezno obliko. V četrtem poglavju je predstavljen postopek statičnega prevajanja kodirnikov za uporabo v aplikaciji, opis uporabe in nastavitev administracijskega spletnega vmesnika aplikacije. V poglavju jim sledi opis postopkov, ki jih aplikacija v teku življenjskega cikla opravlja in zaključi s shemo podatkovne baze.

V zaključku so navedene funkcionalnosti razvite v teku diplomske naloge ter možne izboljšave aplikacije.

2 Opis aplikacije in uporabljenih razvojnih orodij

V tem poglavju je opisan namen aplikacije in shema sistema v katerega sodi. Nadalje je predstavljena osnovna arhitektura aplikacije ter orodja in tehnologije, ki smo jih uporabili tekom razvoja. Predstavljene so omejitve video zapisa in metode za primerjanje kakovosti slike.

2.1 Namen aplikacije

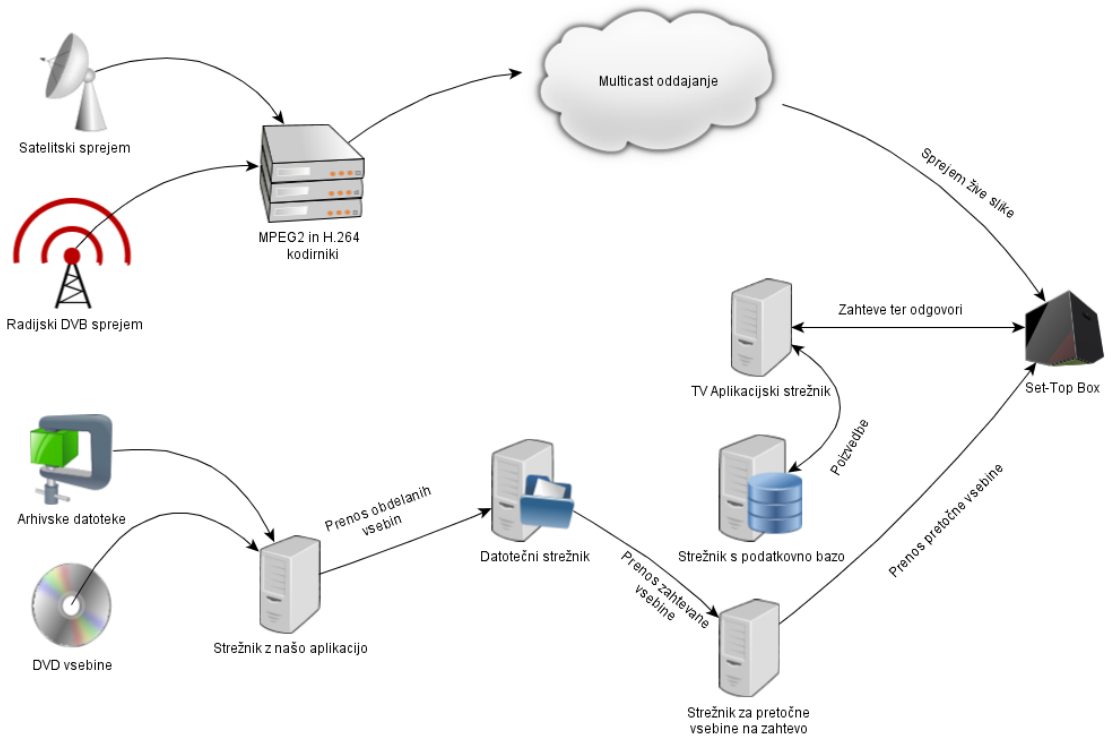
Ponudniki IP televizije in video vsebin na zahtevo zaradi zahtev končnih terminalnih naprav pri naročnikih in lastnostih omrežij, potrebujejo natanko določene oblike video vsebin za zagotavljanje kakovostne storitve. Predpisane oblike in načini kodiranja vsebin pripomorejo k večji združljivosti, lažjem vzdrževanju, preprostejšem upravljanju in olajšajo načrtovanje nadgradenj.

Distributerji in ponudniki video vsebin pogosto priskrbijo ustrezne meta podatke, a vsebine same le redko ustrezajo predpisani obliki ponudnika. Pred vnosom vsebin v svoj sistem jih morajo ponudniki ponovno kodirati v ustrezne oblike in priskrbljene meta podatke pripraviti za vnos v sistem. Pri večjem številu vsebin je ponovno kodiranje, preverjanje in priprava meta podatkov časovno potratno opravilo, zahteva dodatne zaposlene in je izpostavljeno napakam.

V okviru diplomske naloge je bila razvita aplikacija, ki je zmožna sprejeti vire video vsebin iz različnih lokacij, kot so FTP strežniki ter SCP/SFTP strežniki, poleg tega pa je sposobna spremljati tudi vsebino lokalnih map, kamor lahko odložimo datoteke za obdelavo. Ker lahko število prenesenih vsebin hitro preseže strojne zmogljivosti enega računalnika, je aplikacija zmožna delo dinamično porazdeliti v obdelavo na več med seboj povezanih računalnikov. V času, ko ni dovolj vsebin, ki bi upravičevale več vklopljenih računalnikov, lahko za varčevanje aplikacija proste računalnike ugasne in v primeru ponovnega povečanega števila vsebin ponovno vklopi.

Aplikacija omogoča centralno nastavljanje in distribucijo profilov, ki določajo, v kakšno obliko naj aplikacija kodira vsebine iz določenega vira. Za potrebe vnosa v sistem in evidenco preveri, ali je priložena slika naslovnice in iz priloženega XML dokumenta prebere meta podatke o vsebini ter z njimi ustvari nov tekstovni dokument.

Po končanem kodiranju aplikacija prenese obdelane vsebine na datotečni strežnik in shrani varnostno kopijo na svoje diskovno polje. Varnostne kopije aplikacija ob pomanjkanju prostora na diskovnem polju ali po določenem časovnem obdobju samodejno izbriše.

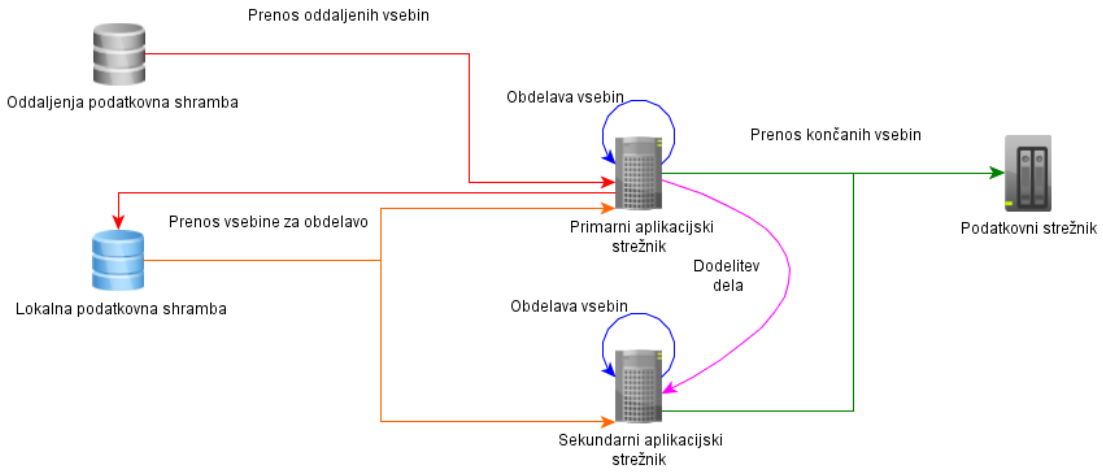


Slika 2: Shema IPTV omrežja.

Na sliki 2 je upodobljena poenostavljena shema celotnega IPTV omrežja. Televizijski signal preko satelitskega ali radijskega DVB oddajanja sprejemajo strojni MPEG2 in h264 kodirniki. Ti ga v ustrezni obliki s pomočjo multicast oddajanja pošiljajo v IPTV omrežje, kjer je na voljo končnim uporabnikom preko set-top box naprave. Vsebine z naše aplikacije so odložene na datotečni strežnik. Uporabnik vsebino izbere preko vmesnika na aplikacijskem strežniku. Izbrana vsebina se na set-top box napravi predvaja preko strežnika za pretočne vsebine na zahtevo.

2.2 Arhitektura aplikacije

Porazdeljeno delovanje aplikacije pri obdelavi vsebin je prikazano na sliki 3. Primarni aplikacijski strežnik nove vsebine iz oddaljene podatkovne shrambe prenese na lokalno podatkovno shrambo, kjer so dostopne tudi ostalim strežnikom. Vsebine v obdelavo dodeljuje primarni strežnik. Le-te z lokalne podatkovne shrambe strežniki prenesejo v obdelavo in jih po končanem postopku odložijo na podatkovni strežnik.



Slika 3: Diagram delovanja aplikacije.

2.3 Izbira orodij

Za izdelavo aplikacije smo uporabili veliko orodij in tehnologij, ki so na kratko opisana v tem poglavju. Poglaviten kriterij pri izbiri je bila odprtokodnost, prosta programska oprema in čim bolj enostavna integracija z obstoječo programsko opremo.

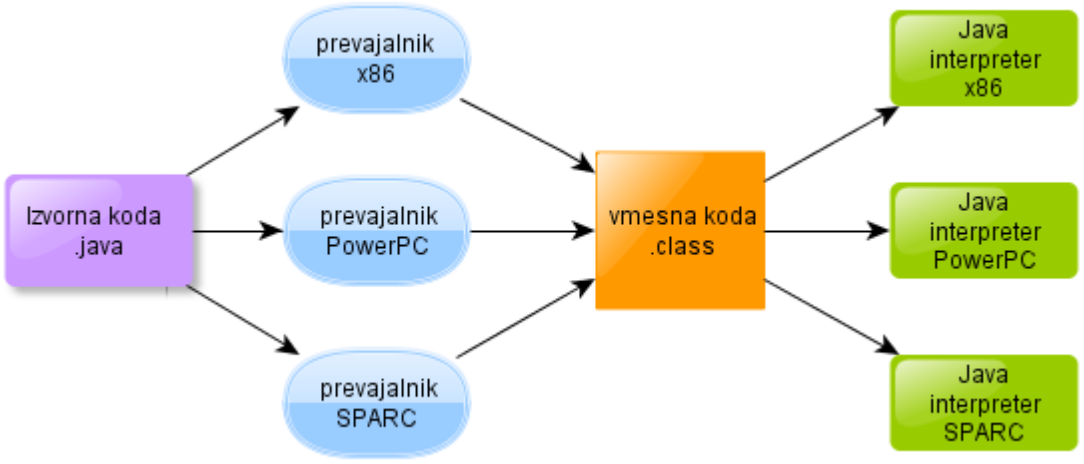
Prosta in odprtokodna programska oprema nam dovoljuje uporabo, razširjanje, spreminjanje in izboljševanje programa ali njegove izvorne kode.

2.3.1 Java

Java je prenosljiv, objektno orientiran programski jezik, katerega poglavitna prednost je WORA (ang. *write once, run anywhere* - napiši enkrat, poganjaj kjerkoli) princip delovanja, ker se izvorna koda najprej prevede v vmesno kodo (ang. *bytecode*) in se interpretira šele ob vsakem izvajanju. Razvijalcem omogoča prenosljivost med operacijskimi sistemi z malo ali nič spremembami v izvorni kodi.

Njeno prenosljivost omogoča JVM (ang. *Java virtual machine*), ki prevaja vmesno kodo v procesorju in operacijskemu sistemu primerne ukaze (slika 4). Vmesna koda se ne spreminja glede na strojno opremo ali operacijski sistem, na katerem aplikacija teče.

Prva referenčna implementacija Jave je bila razvita leta 1991 in predstavljena javnosti leta 1995, kot osrednji del Sun Microsystems Java platforme. Leta 2007 je bila večina izvorne kode Jave izdana pod GNU GPL licenco z izjemo komponent za izrisovanje besedila in 2D rasterizacije. Razvoj nadaljuje podjetje Oracle po prevzemu Sun Microsystems leta 2010.



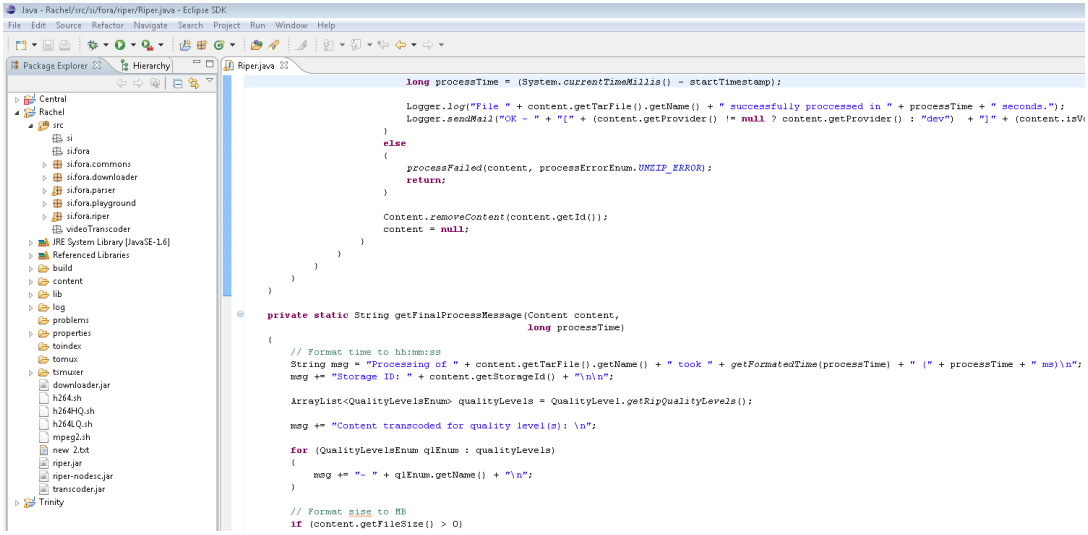
Slika 4: Java in prevajanje izvorne kode v vmesno kodo ter izvrševanje.

Javo smo izbrali zaradi njenih lastnosti in dejstva, da jo že uporabljamo v obstoječem okolju, za katerega razvijamo aplikacijo [1].

2.3.2 Eclipse

Eclipse je priljubljeno in izredno razširjeno odprtokodno razvojno okolje za razvoj v programskem jeziku Java. Podpira tudi veliko ostalih programskih jezikov, med katerimi so C, C++, Fortran, Perl, PHP, Python in Rubi. Dodatno podporo za jezike lahko enostavno namestimo preko vtičnikov.

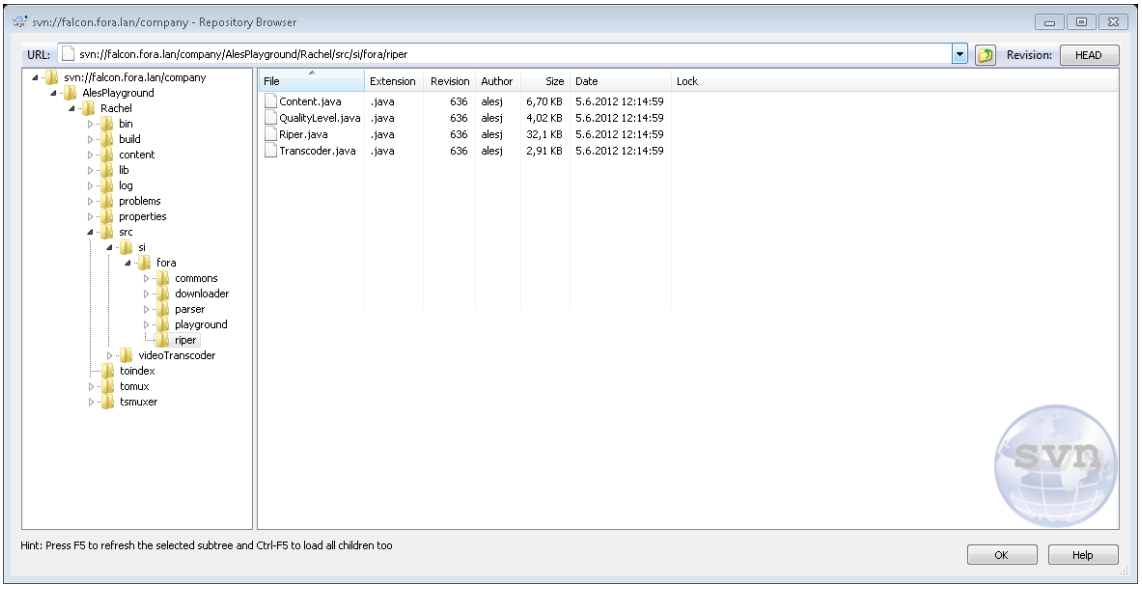
Na sliki 5 je prikazano glavo okno razvojnega orodja **Eclipse** z drevesno strukturo projekta na levi ter oknom za urejanje izvorne kode na sredini. Zaradi svojega močnega vgrajenega razhroščevalnika (ang. *debugger*), enostavnosti uporabe, integracije s sistemom za vodenje različic SVN in možnosti za razširitve, je idealna izbira za razvojno okolje [2].



Slika 5: Razvojno orodje Eclipse.

2.3.3 SVN

Apache Subversion je sistem za vodenje različic programske opreme in izvorne kode. Omogoča nam vodenje trenutnih in preteklih različic izvorne kode, dokumentacije ali aplikacij. Omogoča nam vpogled v zgodovino izvorne kode ter kdaj in kdo je naredil določeno spremembo v izvorni kodi aplikacije. S tem omogoča lažje in hitrejše odkrivanje napak ali hitro povrnitev na katero izmed prejšnjih različic. Kot centralno odložišče izvorne kode omogoča hkratni razvoj in sinhronizacijo različic preko več delovnih postaj [3].



Slika 6: Vsebina SVN repozitorija.

Za svoje delo smo uporabili namizni odjemalec TortoiseSVN, ki se integrira v lupino Windows sistema ter nam s tem omogoča enostaven pregled repozitorija s projekti (slika 6) in delo z revizijami kode na našem SVN strežniku brez uporabe ukazne vrstice.

2.3.4 Video kodirniki

FFmpeg je popolna odprtokodna zbirka programov in knjižnic za snemanje, pretvarjanje in predvajanje večpredstavnostnih vsebin, izdanih pod GNU LGPL 2.1 in GNU GPL 2 licenco. Vsebuje orodja **ffmpeg** za pretvarjanje video vsebin iz ene obliko v drugo, **ffserver** strežnik, ki je sposoben preko protokolov HTTP in RTSP oddajati video vsebine preko omrežja in **ffprobe** orodje v ukazni vrstici za prikazovanje informacij o večpredstavnostnih vsebinah.

Zmožna je delovati na vseh večjih operacijskih sistemih in arhitekturah strojne opreme. Ima izjemno podporo za dekodiranje množice različnih video zapisov. Vsa orodja so nam na voljo v ukazni vrstici, kar je idealno za uporabo v naši aplikaciji [4].

HandBrake je večnamensko odprtokodno orodje za večnitno pretvarjanje video vsebin izdano pod GNU GPL licenco. Čeprav podvaja funkcionalnost in vključuje knjižnice projekta FFmpeg, smo ga izbrali zaradi lažjega in bolj preglednega načina nastavljanja parametrov pri kodiranju v video zapis h264 s kodirnikom x264 [5].

2.3.5 Multiplekser elementarnih zapisov

Za zapis elementarnih video in zvokovnih sledi v transportni večpredstavnostni tok smo uporabili lastni multiplekser razvit v podjetju, s katerim pretvarjamo programski tok, ki ga zapišejo MPEG2 in h264 kodirniki.

Poglavitna razlika pri transportnem toku je njegova zasnova za manj zanesljive transportne medije (na primer satelitsko, antensko ali oddajanje preko IP povezav). Transportni tok vsebuje elementarne tokove razdeljene v manjše paketke po 188 bajtov, ter več mehanizmov za časovno usklajevanje in odpravo napak v primeru degradiranega prenosa.

Vsak transportni tok je sestavljen iz 188 bajtov velikih paketkov, katerim se lahko glede na način prenosa dodajo dodatni podatki za odpravo in zaznavo napak. Vsak paketek vsebuje 184 bajtov podatkov in 4 bajte rezervirane za zaglavje paketka.

Elementarni video in zvokovni zapisi, ki skupaj sestavljajo svoj programski elementarni tok ali PES (ang. *packetized elementary stream*), so v transportnem toku enolično označeni s svojo PID (ang. *packet ID*) številko v zaglavju posameznega paketka.

Transportne tokove ločimo v dve poglavitni skupini in sicer na tokove, ki vsebujejo en elementarni tok in jih imenujemo SPTS (ang. *single program transport stream*) ali več elementarnih tokov, ki jih imenujemo MPTS (ang. *multiple program transport stream*).

Ker lahko paketki na cilj prispejo v drugačnem vrstnem redu, kot so bili poslani, vsak paketek v zaglavju nosi tudi časovno oznako DTS ali PTS (ang. *decoding time stamp*, *presentation time stamp*), ki natančno označujejo čas, ko mora biti video ali zvokovni zapis dekodiran ter izrisan na ekran.

Če želi naprava transportni tok prikazati na zaslonu, potrebuje PID številko elementarnega toka. Poleg elementarnih tokov se v transportnem toku za ta namen pošiljajo signalne tabele imenovane PAT (ang. *program association table*) in PMT (ang. *program map table*), ki predstavljajo samostojen kontrolni kanal.

PAT tabela ima po standardu določeno PID številko 0x000 ter nosi informacije o PID številkah PMT tabel.

Vsaka PMT tabela opisuje svoj program, ki je sestavljen iz elementarnih video, zvokovnih ter dodatnih zapisov, kot so podnapisi ali teletext. Za vsakega izmed elementarnih zapisov PMT tabela nosi informacijo o njegovi PID številki [6].

2.3.6 FTP

FTP ali File Transfer Protocol je standardni mrežni protokol za prenos podatkov med seboj povezanimi računalniki preko lokalnega omrežja ali interneta. Čeprav v svoji prvotni implementaciji ne nudi varnosti prenosa ali avtentikacijskih podatkov (varnostni mehanizmi

kot je FTPS (FTP preko SSL/TLS) so bili dodani kasneje), je široko razširjena metoda za prenos večjih datotek in v našem premeru za prenos večpredstavnostnih vsebin distributerjev.

V aplikaciji smo uporabili *Apache Commons Net* knjižnico za vzpostavitev in prenos podatkov preko FTP protokola ter protokol SMTP za pošiljanje elektronske pošte. Knjižnica nam nudi širok spekter pogostih protokolov [7].

Podprti protokoli so sledeči:

- FTP/FTPS,
- FTP preko HTTP,
- NNTP,
- SMTP(S),
- POP3(S),
- IMAP(S),
- Telnet,
- TFTP,
- Finger,
- Whois,
- rexec/rcmd/rlogin,
- Time (rdate) in Daytime,
- Echo,
- Discard,
- NTP/SNTP.

2.3.7 SCP

SCP ali secure copy omogoča varen prenos podatkov z uporabo SSH protokola, ki je na voljo v vseh modernih Linux distribucijah ter sistemih, ki temeljijo na *NIX arhitekturi. Zaradi uporabe varnostnih mehanizmov protokola SSH, kot sta močno šifriranje ter uporaba asimetričnih ključev, omogoča zaupnost in istovetnost prenesenih podatkov.

Za prenos datotek med računalniki, kjer teče naša aplikacija in datotečnimi strežniki, smo v naši aplikaciji uporabili knjižico *Ganymed SSH-2*, ki v Javi implementira SSH2 protokol ter omogoča enostaven SCP prenos podatkov na vseh platformah brez odvisnosti od sistemskih orodji [8].

2.3.8 SQLite

SQLite je majhna implementacija sistema za upravljanje relacijskih podatkovnih baz (ang. *relational database management system*). Je integralen del aplikacije, kamor jo razvijalec vključi kot knjižnico in v primerjavi z večjimi sistemi ne uporablja ločenega procesa za dostop do podatkovne baze.

Zaradi enostavne uporabe, implementacije večine standardov SQL in njegove skladnosti s principi ACID (ang. *atomicity, consistency, isolation, durability*), je pogosta izbira za lokalno hranjenje podatkov pri manjših in prenosnih aplikacijah [9].

2.3.9 SMTP

SMTP ali *simple mail transfer protocol* je internetni standard za prenos elektronske pošte, ki je bil prvič definiran leta 1982 ter nazadnje posodobljen leta 2009. SMTP uporabljajo poštni strežniki za medsebojni prenos elektronske pošte in poštni odjemalci za oddajanje. Omogoča tudi varnostne razširitve za šifriran prenos pošte preko SSL/TLS protokola.

Aplikacija za pošiljanje obvestil ter napak uporablja protokol SMTP iz knjižice *Apache Commons Net*.

2.3.10 7zip

Vsebine v obdelavo prispejo v arhivirani obliki, zato jih je potrebno ekstrahirati pred postopkom kodiranja. Knjižnica *7-Zip-JBinding* nudi prenosljivo in od sistema neodvisno podporo za ekstrahiranje arhivov. Je veja brezplačnega odprtokodnega programa 7-zip za delo z arhivskimi datotekami pod licenco GNU LGPL. S tem nudi podporo za ekstrahiranje 7z, XZ, BZIP2, GZIP, TAR, ZIP, WIMARJ, CAB, CHM, CPIO, CramFS, DEB, DMG, FAT, HFS, ISO, LZH, LZMA, MBR, MSI, NSIS, NTFS, RAR, RPM, SquashFS, UDF, VHD, WIM, XAR in Z arhivov [10].

2.3.11 Samba

Samba je prosto programska in odprtokodna implementacija mrežnega protokola SMB/CIFS za skupno rabo datotek in tiskalnikov ter implementira tehnologije, kot so strežnik WINS in domene Active Directory.

Sambo je možno poganjati na širokem spektru operacijskih sistemov, kot so Linux, Solaris, AIX in BSD, med njimi tudi Mac OS X. S tem zagotavlja medsebojno združljivost z mrežnim datotečnim sistemom različic Microsoft Windows. Razvil jo je Andrew Tridgell leta 1991 na avstralski državni univerzi s pomočjo povratnega inženiringa protokola netbios. S poudarkom na združljivosti s takratnim Microsoft LAN Manager, je leta 1993 pod licenco GPL2 izdal prvo različico, ki je vsebovala tako strežnik kot tudi odjemalec. Svoje dokončno ime "Samba" je dobila po različici 1.5, ko se je preimenovala iz "smbserver".

Zaradi svoje vsestranske združljivosti med sistemi je optimalna izbira za zagotavljanje dostopa do skupnega odložišča datotek preko lokalnega omrežja za našo aplikacijo [11].

2.4 Omejitve in kakovost video zapisa

Naša aplikacija pretvarja vhodne video vsebine v dve našemu sistemu primerni obliki. Za svoje delo uporablja kodirnika MPEG2 ter x264. Pri končnem rezultatu se je potrebno ozirati na končno velikost datoteke, hitrost kodiranja ter seveda na kakovost slike.

2.4.1 Zahteve

Da zagotovimo nemoten prenos preko širokega spektra prenosnih medijev, kot so optična vlakna, bakrene parice, preko katerih poteka VDSL2 komunikacija ter brezžična omrežja, smo omejeni na največjo bitno hitrost pri 6600 kilobajtih na sekundo pri MPEG2 vsebinah ter

3200 kilobajtih na sekundo pri h264 vsebinah. Zvokovne sledi so zapisane s prostorskim AC3, AAC ali stereo MPA kodirnikom pri 224 kilobitih na sekundo. Video zapis pa ima velikost razmika med ključnimi slikami na 25 slik ali manj.

2.4.2 Metode primerjanja

Z dodatnimi nastavitvami kodirnika lahko znotraj zahtev iztržimo optimalno kvaliteto slike ter čas, potreben za kodiranje. Primerjava kvalitete je zelo problematična, saj ljudje subjektivno ocenjujejo kvaliteto slike, to pa z metrikami težko določimo.

Vseeno sta PSNR in SSIM široko uporabljeni objektivni metriki, ki naj bi določili kvaliteto slike glede na vhod in našo obdelano vsebino na približen način, kot to počnemo ljudje.

PSNR ali Peak Signal to Noise Ratio je metrika, ki prikaže razmerje med vhodno vsebino ter popačenjem ali šumom v video zapisu, ki se pojavi med kodiranjem [12].

S PSNR metriko pri kodirniku h264 naletimo na težave pri psiho-vizualnih optimizacijah, ki neposredno poslabšajo rezultat primerjave, saj je metrika le velik približek človeškemu zaznavanju. Človeško oko veliko bolje prepoznava razlike v sami strukturi slike in ni toliko občutljivo na šum ter spremembe v posamezni točki.

PSNR običajno izrazimo v decibelni logaritemski skali in ga izračunamo s pomočjo povprečne kvadratne napake na luma komponenti.

Luma komponenta ali svetilnost v video posnetku predstavlja kontrastno komponento slike oz. akromatski sivinski del brez barv.

Za izračun PSNR za vsako sliko najprej izračunamo povprečno kvadratno napako po formuli

$$PKN = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \,, \tag{2.1}$$

kjer **m** in **n** predstavljata dimenzije sivinskih slik **I** in **K** v pikah. **I** predstavlja našo izvirno sliko (slika 7), s **K** pa označimo naš stisnjen vzorec (slika 8).

Po izračunani PKN za našo sliko izračunamo še PSNR po formuli

$$PSNR = 10\ \log_{10}\left(\frac{MAX_I^2}{PKN}\right), \tag{2.2}$$

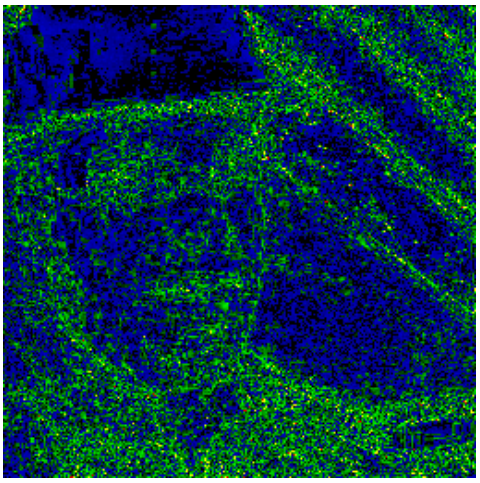
kjer je MAX_I največja vrednost piksla v sliki **I**. Največja vrednost, ki jo lahko MAX_I zavzame na 8 bitni monokromatski sliki, je 255. PSNR lahko izračunamo tudi nad barvno sliko, kjer moramo upoštevati vse barvne kanale RGB

$$PSNR = 10\ \log_{10} \frac{MAX_I^2 \cdot m \cdot n}{\sum_{i=0,j=0}^{m,n} [I(i,j) - K(i,j)]^2} \,. \tag{2.3}$$

Primerjavo dveh slik lahko izrazimo tudi z vizualizacijo PSNR vrednosti (slika 9).

Slika 7: Izvorna slika v PSNR primerjavi.

Slika 8: Stisnjena slika v PSNR primerjavi.



Slika 9: Vizualizacija šuma PSNR vrednosti med slikama.

SSIM (ang. *structural similarity*) je metrika, ki namesto primerjave šuma, ki se pojavi med kodiranjem, primerja strukturne razlike med izvornim video zapisom in našim končnim rezultatom. Metrika je v primerjavi s PSNR zahtevnejša za izračun, saj primerja vhodni zapis in rezultat s plavajočo 8x8 polj veliko matriko ter izračunane indekse na koncu združi [13].

SSIM izračunamo na luma komponenti po formuli

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}, \tag{2.4}$$

kjer $\boldsymbol{\mu}$ predstavlja povprečje po komponenti, $\boldsymbol{\sigma}$ varianco po komponenti ter kovarianco **x** in **y**, **c** pa konstanto, ki stabilizira numerično vrednost ulomka pri manjših vrednostih imenovalca in jo izračunamo po formulah

$$c_1 = (k_1 L)^2, \tag{2.5}$$

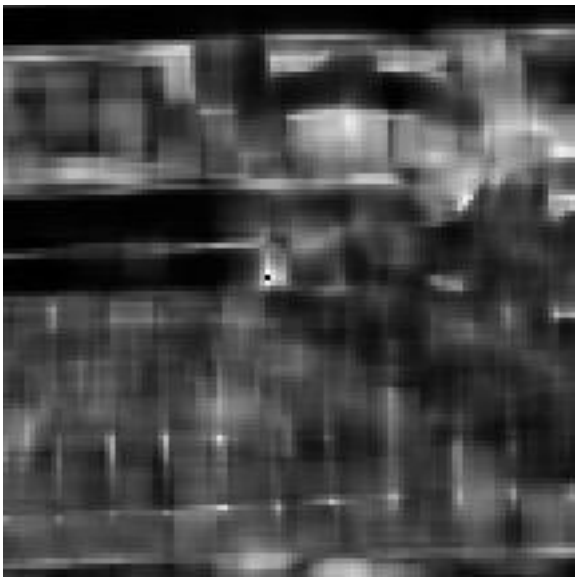
$$c_2 = (k_2 L)^2. \tag{2.6}$$

L predstavlja razmerje med največjo in najmanjšo vrednostjo pik v sliki, **k** pa zavzema vrednosti 0,01 in 0,03.

SSIM vrednosti lahko tudi vizualiziramo. Na sliki 10 je prikazana izvorna slika. Na sliki 11 je prikazana stisnjena slika. Vizualizacija razlik z metodo SSIM je prikazana na sliki 12.

Slika 10: Izvorna slika v SSIM primerjavi.

Slika 11: Stisnjena slika v SSIM primerjavi.



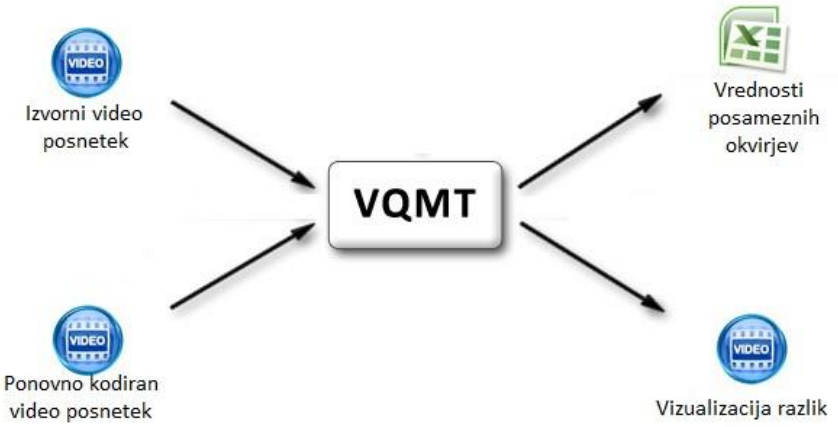
Slika 12: Vizualizacija SSIM vrednosti.

Kakovost izhodne slike je izredno subjektiven pojem. Ker obe metriki le približno določata kakovost, smo končne izdelke primerjali tudi vizualno.

2.4.3 Orodje

Za izračun SSIM metrike smo uporabili program MSU Quality Measurement Tool. Program sprejme originalni video zapis in naš rezultat (slika 13). Za posamezno sliko v posnetku poda vrednosti in vizualizacijo, kjer lahko vidimo, kje prihaja do razlik. Vrednosti so na intervalu med 1 in -1, kjer 1 pomeni, da sta primerjani sliki identični, -1 pa popolnoma drugačni.

Čas, potreben za kodiranje, smo merili z linux orodjem *time,* bitno hitrost pa smo preverjali s programom Bitrate Viewer.



Slika 13: Vhodni in izhodni podatki programa Quality Measurement Tool.

3 Kodiranje vsebin

Za preizkušanje načina kodiranja smo izbrali reprezentativne vzorce scen, ki se pogosto pojavljajo pri video vsebinah. Za vzorec smo izbrali dve sceni iz igranih filmov in dve sceni iz animiranih filmov. Vzorca igranih scen sta zavzemala počasen posnetek dialoga s statičnim ozadjem, ter hitro akcijsko sceno. Prvi vzorec animirane scene je predstavljal počasen prizor z ostrimi črtami, premikajočim ozadjem in statičnim glavnim likom. Drugi animirani vzorec pa je zavzemal panoramske posnetke, hitro gibajoče prizore ter prizore s širokim barvnim spektrom.

Kodiranje vzorcev smo izvedli z orodjema FFmpeg in HandBrake. Z orodjem FFmpeg smo vzorce kodirali s pomočjo kodirnika MPEG2 in z orodjem HandBrake vzorce kodirali s pomočjo kodirnika x264. Raziskali smo dodatne argumente za kodirnika, s katerimi smo želeli izboljšati kakovost slike znotraj podane omejitve bitne hitrosti.

Rezultate naših poizkusov smo analizirali z orodjem Bitrate Viewer [14], s pomočjo katerega smo preverili bitno hitrost rezultata. Rezultat poizkusa smo smatrali kot uspešen, če bitna hitrost ni presegla pragu 6600 kilobitov na sekundo pri kodirniku MPEG2, oziroma 3200 kilobitov na sekundo pri kodirniku x264.

Poleg analize bitne hitrosti smo primerjali tudi kakovost slike v primerjavi z izvornim vzorcem, s pomočjo orodja MSU Quality Measurement Tool [15]. Izhodne vrednosti orodja smo predstavili v obliki grafa, na podlagi katerega smo se odločili o uspešnosti poizkusa. Glede na naše zahteve, smo kot najbolj uspešen smatrali tisti poizkus, ki je imel najmanjši odklon v kakovosti slike v primerjavi z izvornim vzorcem.

Kodiranje se je izvajalo na računalniku s procesorjem AMD FX 8120 pri 3,1GHz z 8GB pomnilnika.

3.1 Razlaga parametrov za kodiranje

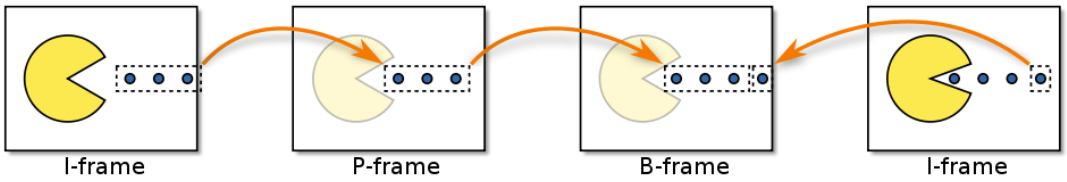
Prvotna nastavitev kodirnikov je bila nastavljena glede na osnovne zahteve, zato je bilo potrebno omejiti bitno hitrost s primerno toleranco ter razmik ključnih slik. Z nadaljnjo izbiro parametrov smo želeli izboljšati kvaliteto slike, ob upoštevanju hitrosti samega kodiranja.

3.1.1 B-okvirji

Struktura kompresiranega video posnetka je definirana v skupini slik (ang. *group of pictures – GOP).* GOP predpisuje zaporedje, v katerem so razporejeni intra in inter okvirji (slika 14).

Intra okvirji so referenčni okvirji slike in predstavljajo celotno sliko, ki je neodvisna od predhodne ali naslednje slike v zaporedju in ne potrebuje dodatnih informacij za rekonstrukcijo. Posledično je faktor kompresije znatno manjši. Vsako GOP zaporedje se začne z intra ali "I" okvirjem, temu sledijo inter P in B okvirji.

Inter okvirji se za rekonstrukcijo slike sklicujejo na druge okvirje ter s tem povečajo faktor kompresije. V "P" (ang. *predicted picture*) ali *delta* okvirih so zapisane le spremembe slike glede na prejšnji okvir. Na primer v prizoru s premikajočim predmetom ni potrebe po zapisu statičnega ozadja, ki se ni spremenilo glede na prejšnji okvir, ampak le pike, ki so se spremenile zaradi gibanja predmeta. "B" okvirji (ang. *bi-predictive picture*) omogočajo še dodatno kompresijo z zapisom sprememb glede na predhodni in naslednji okvir v zaporedju.



Slika 14: Zaporedje okvirjev in sklicevanje v GOP zaporedju.

GOP zaporedje je zapisano z dvema številkama, M in N. M definira razmik med I in P okvirji, N pa razmik med dvema I okvirjema, na primer M=3, N=12 opiše strukturo "IBBPBBPBBPBBI".

3.1.2 Makro bloki

Makro bloki so komponenta kompresirane slike ali video posnetka, ki so sestavljeni iz dveh ali več 8x8 ali 16x16 velikih kvadratov pik (ang. *pixel*). Posamezen makro blok ima zapisan svoj naslov v sliki, tip makro bloka, kvantizacijsko vrednost, vektor gibanja, bitno masko, s katero določi prisotne koeficiente, zapis o svetlosti ter razliko modre in rdeče barve.

3.1.3 Trellis kvantizacija

Gre za kvantizacijski algoritem, ki optimira zapis sledi okoli objekta pri algoritmih z diskretno kosinusno transformacijo za iskanje gibanja in lahko izboljša kompresijo ter kakovost slike z iskanjem optimalne kvantizacije za vsak posamezen makro blok.

Algoritem po izbiri vrste makro bloka in vektorja gibanja v posamezni sliki izračuna diskretno kosinusno transformacijo razlike med vhodno sliko in predikcijo gibanja v intra okvirjih, ter se odloči za zapis koeficientov diskretne kosinusne transformacije, ki imajo najmanjše razmerje popačenja. Ta proces imenujemo kvantizacija.

3.1.4 Algoritem za iskanje gibanja

Algoritmi za iskanje gibanja iščejo vektorje gibanja posameznih pik, makro blokov ali celotne slike pri spremembi slike v naslednjo pri okvirjih v video posnetkih, ter s tem preprečujejo ponoven zapis redundantnih podatkov. Ker je gibanje projekcija tridimenzionalnega okolja na dvodimenzionalno ravnino, je opravilo vse prej kot trivialno. V grobem lahko metode za iskanje vektorjev gibanja razdelimo v direktne in indirektne.

Direktne metode operirajo neposredno s pikami slike, z iskanjem ujemajočih blokov med slikami, iskanjem fazne korelacije z Bayesovimi ocenjevalci ali optičnim tokom.

Indirektne metode se iskanja lotijo s pomočjo algoritmov za zaznavanje robov ter podobnosti iščejo s pomočjo statističnih funkcij.

3.1.5 Izbira video izsekov

Za preizkušanje kodiranja potrebujemo vzorce, ki se pogosto pojavljajo pri video vsebinah. Med najbolj pogostimi scenami so počasni premiki kamere z nekaj rezi med dialogom, zelo hitre akcijske scene, kjer ima kodirnik veliko opravka s hitrimi rezi, premiki kamere in večjim številom majhnih delcev. Vsebine z risano ali animirano 3D vsebino pa so izredno občutljive zaradi svojih ostrih robov ter linearnih premikov kamere okoli lika, pri čem se premika ozadje.

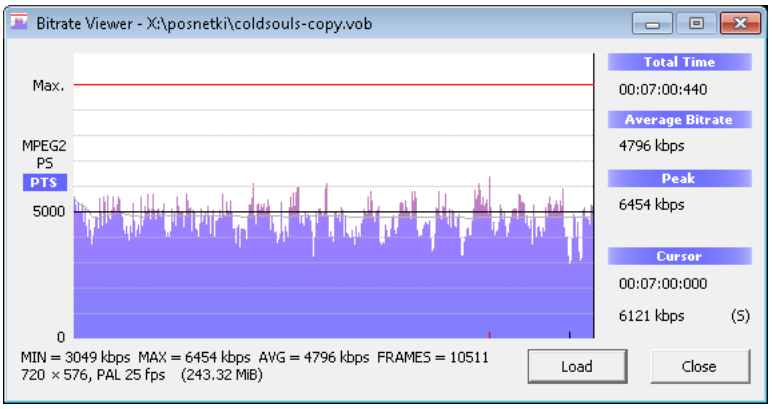
Za preizkušanje smo izbrali 4 izseke dolge 7 minut in jim izmerili bitno hitrost za kasnejšo primerjavo. Za počasne filmske scene smo izbrali izsek iz filma *Cold Souls* (http://www.imdb.com/title/tt1127877/), hitro akcijsko sceno pa iz filma *War, inc.* (http://www.imdb.com/title/tt0884224/). Risana in animirana izseka smo izbrali iz 3D animirane risanke *Brave* (http://www.imdb.com/title/tt1217209/) ter *The Secret World of Arrietty* (http://www.imdb.com/title/tt1568921/).

3.2 Priprava video izsekov

Počasno sceno z dialogom smo s spodnjim ukazom izrezali iz filma *Cold Souls* med 00:09:00 in 00:16:00.

```
ffmpeg -ss 540 -t 420 -i coldsouls.vob -vcodec copy -an coldsouls-copy.vob
```

Bitna hitrost izseka je prikazana na sliki 15. S slike lahko razberemo, da je povprečje bitne hitrosti 4796 kilobitov na sekundo ter najvišja vrednost 6454 kilobitov na sekundo. Ker je izvor vzorca DVD medij, je že zapisan v obliki MPEG2 in tako že prvotno ne presega naših omejitev, kar znatno olajša delo kodirnika.

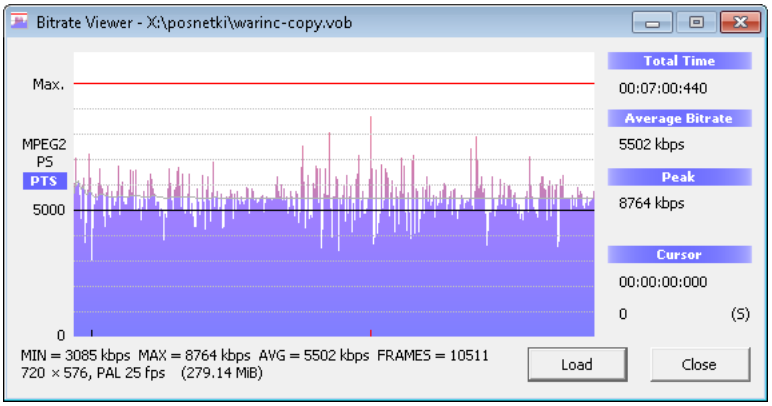


Slika 15: Bitna hitrost izvirnega vzorca Cold Souls.

Hitro akcijsko sceno smo s spodnjim ukazom izrezali iz filma *War, inc.* med 01:02:58 in 01:09:58 z bitno hitrostjo izseka na sliki 16.

```
ffmpeg -ss 3778 -t 420 -i vojna_dd.vob -vcodec copy -an warinc-copy.vob
```

S slike lahko razberemo, da je povprečje bitne hitrosti 5502 kilobitov na sekundo in najvišja vrednost 8764 kilobitov na sekundo. Svetlejše obarvani vrhovi predstavljajo večje odklone bitne hitrosti od povprečja pri hitrih akcijskih spremembah prizora. Pri prikazani bitni hitrosti bi pri končnih uporabnikih lahko prišlo do popačenja slike zaradi napak v prenosu. Pri kodiranju je bilo potrebno te odklone primerno omejiti pod določen najvišji prag.



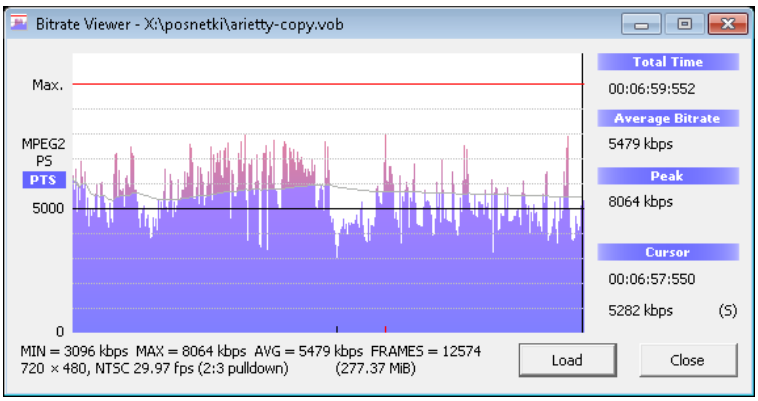
Slika 16: Bitna hitrost izvirnega vzorca War, Inc.

Počasno sceno z dinamično osvetlitvijo in ostrimi črtami smo s spodnjim ukazom vzeli iz animiranega filma *The Secret World of Arrietty* med 00:09:25 in 00:16:25. Bitna hitrost izseka je prikazana na sliki 17.

```
ffmpeg -ss 565 -t 420 -i arietty.vob -vcodec copy -an arietty-copy.vob
```

S slike lahko razberemo, da je povprečje bitne hitrosti 5479 kilobitov na sekundo in svojo najvišjo vrednost zavzame pri 8064 kilobitih na sekundo. V primerjavi s sliko 16, kjer bitna

hitrost preseže omejitev le za sekundo ali dve, tukaj dosega previsoke vrednosti za daljše časovno obdobje. Delo kodirnika je bilo pri tem vzorcu veliko bolj zahtevno.

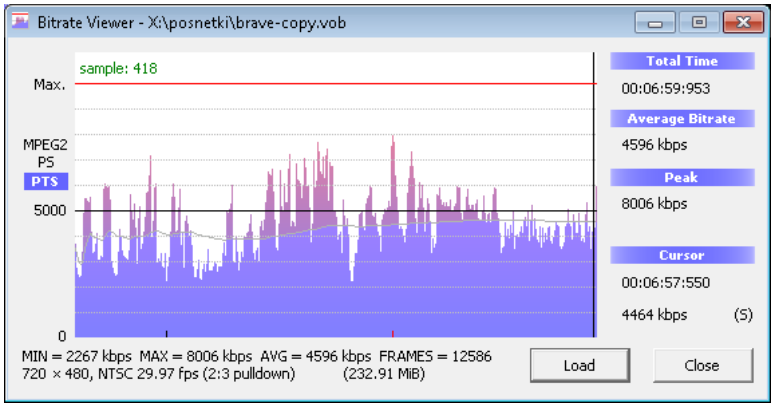


Slika 17: Bitna hitrost izvirnega vzorca Arietty.

Iz filma *Brave* smo s spodnjim ukazom izrezali mešano sceno s panoramskimi posnetki, hitrimi in počasnimi premiki med 00:05:10 in 00:12:10. Bitna hitrost izseka je prikazana na sliki 18.

```
ffmpeg -ss 310 -t 420  -i brave.vob -vcodec copy -an brave-copy.vob
```

S slike razberemo povprečno bitno hitrost pri 4596 kilobitih na sekundo in najvišjo vrednost pri 8006 kilobitih na sekundo. Tako kot na sliki 17, tudi tukaj bitna hitrost preseže dovoljen prag za daljše časovno obdobje. Na grafu bitne hitrosti lahko hitro opazimo, kdaj so v prizoru prisotne hitre spremembe ter povečano število podrobnosti. Začetek in konec vzorca vsebujeta počasne panoramske prizore ter dialog med liki, kar je na grafu opazno kot zmanjšana bitna hitrost v primerjavi s sredino, kjer so prisotni daljši hitri prizori.



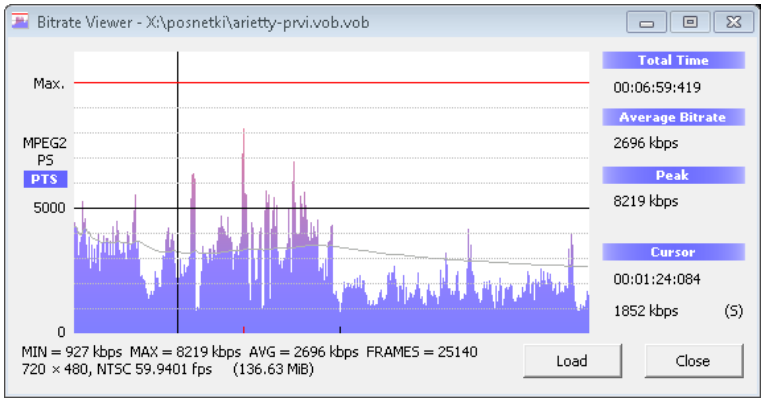
Slika 18: Bitna hitrost izvirnega vzorca Brave.

3.3 Izbira parametrov za kodirnik MPEG2

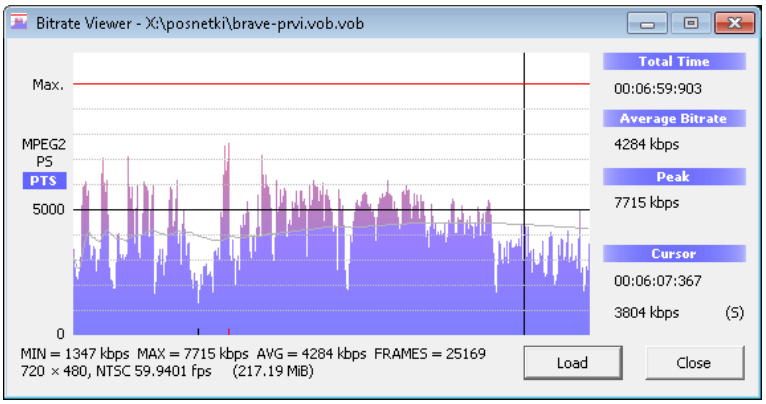
Prvi preizkus smo izvedli z osnovnimi parametri. Omejili smo bitno hitrost s primerno toleranco z argumenti -**b**, -**minrate**, -**maxrate** in -**bufsize** ter nastavili razmik ključnih slik z argumentom -**g**. Z argumenti -**an** smo onemogočili obdelavo zvočne sledi, s -**threads** kodirniku dodelili 7 procesorskih jeder za hitrejšo obdelavo vsebine in s -**f** nastavili izhodni profil kot DVD zapis.

```
time ffmpeg -i <vzorec> -vcodec mpeg2video -b:v 4000k -minrate 500k -maxrate 6500k -bufsize 1835k -g 25 -an -threads 7 -an -f vob -y <vzorec>-prvi.vob
```

Pri našem prvem poizkusu smo opazili, da mehanizem za omejevanje bitne hitrosti orodja FFmpeg pri kodiranju v MPEG2 obliko ne deluje optimalno. Pri vzorcih *The Secret World of Arrietty in Brave* je prišlo do prekoračenja največje dovoljene bitne hitrosti. Prekoračitev je prikazana na slikah 19 in 20.



Slika 19: Prekoračenje omejitve bitne hitrosti vzorca Arietty.



Slika 20: Prekoračenje omejitve bitne hitrosti vzorca Brave.

Argumente prvega preizkusa smo zaradi prekoračitve omejitve bitne hitrosti zavrgli kot neprimerne.

Argumente za drugi preizkus smo iskali v več iteracijah. Primerno omejitev bitne hitrosti smo dosegli z argumenti v spodnjem ukazu.

```
time ffmpeg -i <vzorec> -vcodec mpeg2video -b:v 2800k -minrate 500k -maxrate 6500k -bufsize 1835k -g 25 -an -threads 7 -an -f vob -y <vzorec>-drugi.vob
```

Nato smo izvedli preizkuse, kjer smo dodajali argumente za izboljšanje slike: **bf** za kodiranje z B okvirji, **trellis** za uporabo trellis kvantizacije, **mbd** za določanje algoritma izbire makroblokov ter **cmp** in **subcmp,** s katerima določamo algoritem za iskanje gibanja. Končne vzorce smo primerjali s programom *MSU Quality Measurement Tool.*

V tretjem preizkusu smo za kodiranje vzorca v ukaz iz drugega preizkusa dodali argument **-bf 2**, s katerim je kodirnik vzorec zapisal s pomočjo B-okvirjev. S tem smo želeli ohraniti kakovost zapisa slike vzorca pri nižji bitni hitrosti.

```
time ffmpeg -i <vzorec> -vcodec mpeg2video -b:v 2800k -minrate 500k -maxrate 6500k -bufsize 1835k -g 25 -bf 2 -an -threads 7 -an -f vob -y <vzorec>-tretji.vob
```

Za četrti preizkus smo v ukaz dodali argument **-trellis 2**, s katerim je kodirnik pri zapisu vzorca uporabil trellis kvantizacijo. S tem smo želeli zmanjšati popačenje slike v okolici premikajočih objektov.

```
time ffmpeg -i <vzorec> -vcodec mpeg2video -b:v 2800k -minrate 500k -maxrate 6500k -bufsize 1835k -g 25 -bf 2 -trellis 2 -an -threads 7 -an -f vob -y <vzorec>-cetrti.vob
```

V petem preizkusu smo v ukaz dodali argument **-mbd rd** za izbiro makro blokov na podlagi najboljšega razmerja med kakovostjo slike in bitno hitrostjo. S tem smo želeli sliko pri dani bitni hitrosti zapisati z najvišjo možno kakovostjo.

```
time ffmpeg -i <vzorec> -vcodec mpeg2video -b:v 2800k -minrate 500k -maxrate 6500k -bufsize 1835k -g 25 -mbd rd -bf 2 -trellis 2 -an -threads 7 -an -f vob -y <vzorec>-peti.vob
```

Z argumentom **-cmp 2** smo v ukazu za šesti preizkus določili algoritem za iskanje gibanja.

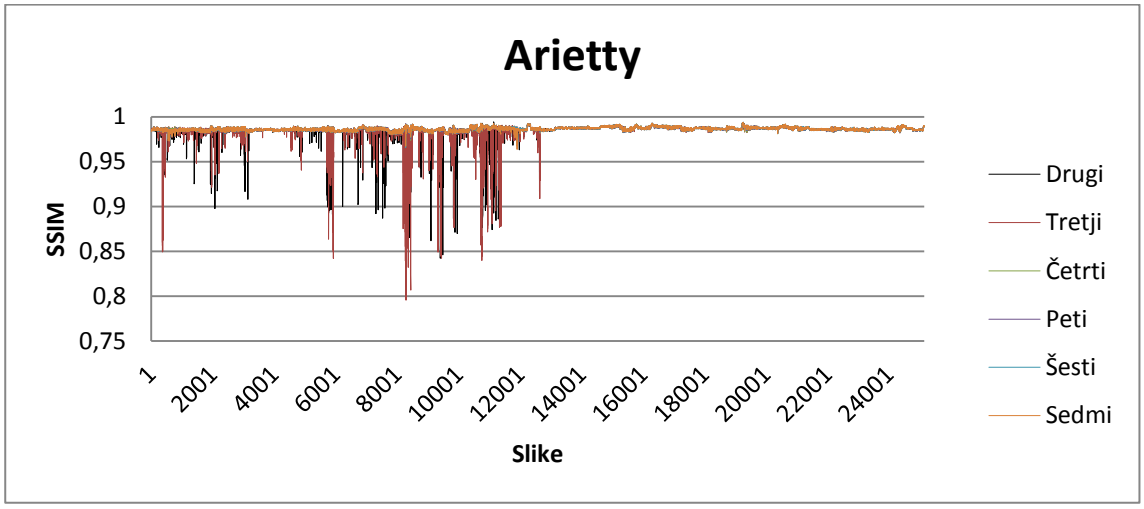
```
time ffmpeg -i <vzorec> -vcodec mpeg2video -b:v 2800k -minrate 500k -maxrate 6500k -bufsize 1835k -g 25 -mbd rd -bf 2 -trellis 2 -cmp 2 -an -threads 7 -an -f vob -y <vzorec>-sesti.vob
```

V sedmem preizkusu pa smo z dodatnim argumentom **-subcmp 2** želeli izboljšati natančnost algoritma za iskanje gibanja.

```
time ffmpeg -i <vzorec> -vcodec mpeg2video -b:v 2800k -minrate 500k -maxrate 6500k -bufsize 1835k -g 25 -mbd rd -bf 2 -trellis 2 -cmp 2 -subcmp 2 -an -threads 7 -an -f vob -y <vzorec>-sedmi.vob
```

3.3.1 Kodiranje vzorca "The Secret World of Arrietty"

Analiza vzorcev *The Secret World of Arrietty* nam pokaže, da kvaliteta drugega in tretjega preizkusa zaradi pomanjkanja argumentov za optimizacijo kakovosti slike izredno odstopa v kvaliteti izvirnega vzorca (slika 21).

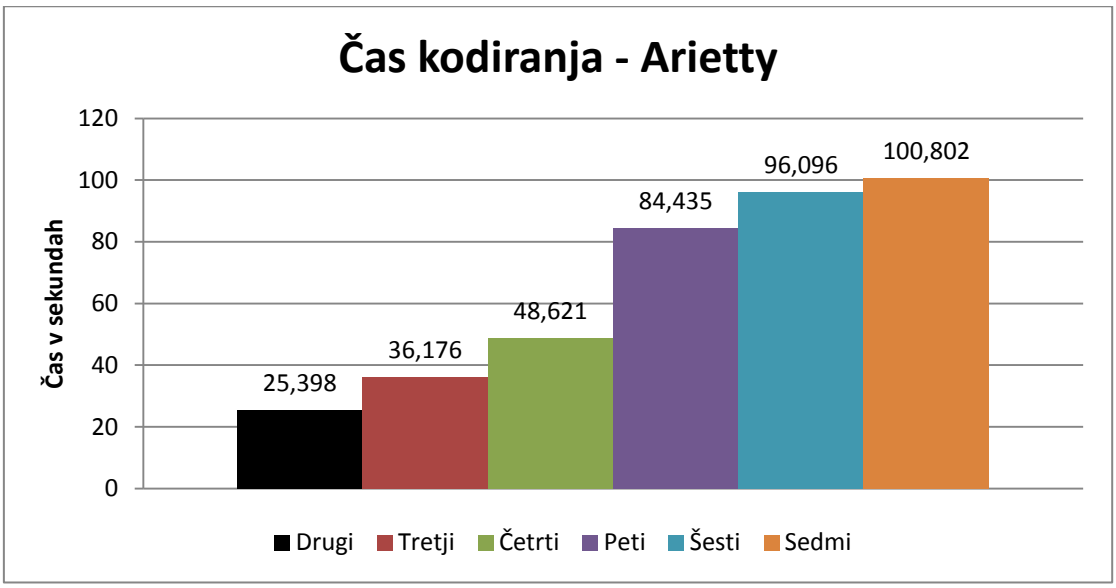


Slika 21: SSIM ocena vseh preizkusov podobnosti slike za vzorec *The Secret World of Arrietty.*

Med ostalimi preizkusi ni opazne razlike (slika 22) v kvaliteti slike. Vsi so sliko zapisali z več kot 98% podobnostjo glede na izvirnik.

Slika 22: SSIM ocena primernih preizkusov podobnosti slike za vzorec *The Secret World of Arrietty.*

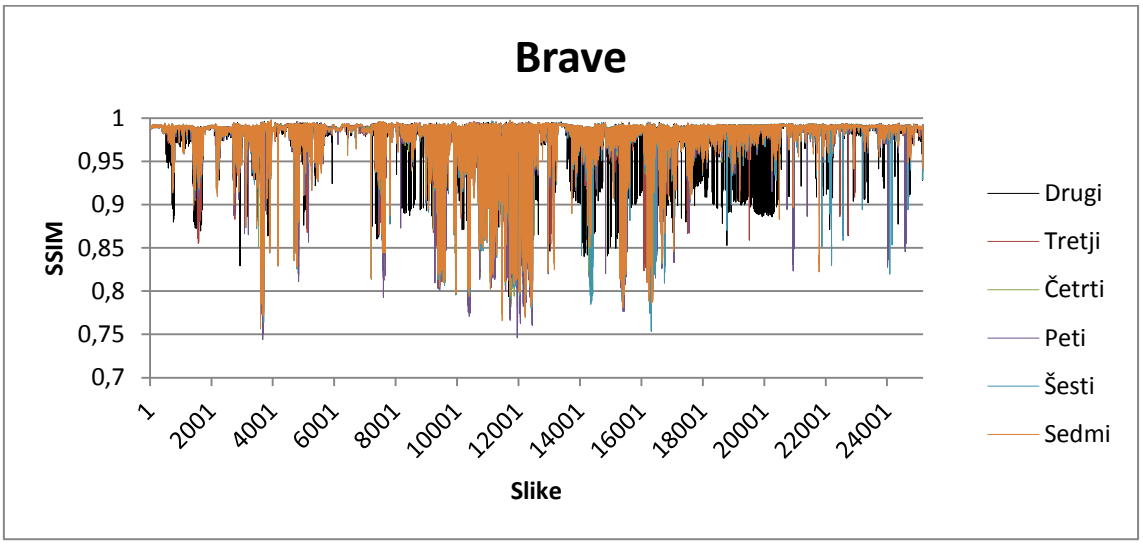
Drugi in tretji preizkus sta z delom zaključila najhitreje, a sta zaradi večjih razlik v zapisu slike manj primerna. Četrti preizkus je sliko zapisal z enako natančnostjo, a veliko hitreje kot preostali preizkusi, ter s tem postal primarni kandidat. Čas kodiranja je prikazan na sliki 23.



Slika 23: Graf časa obdelave vzorcev *The Secret World of Arrietty.*

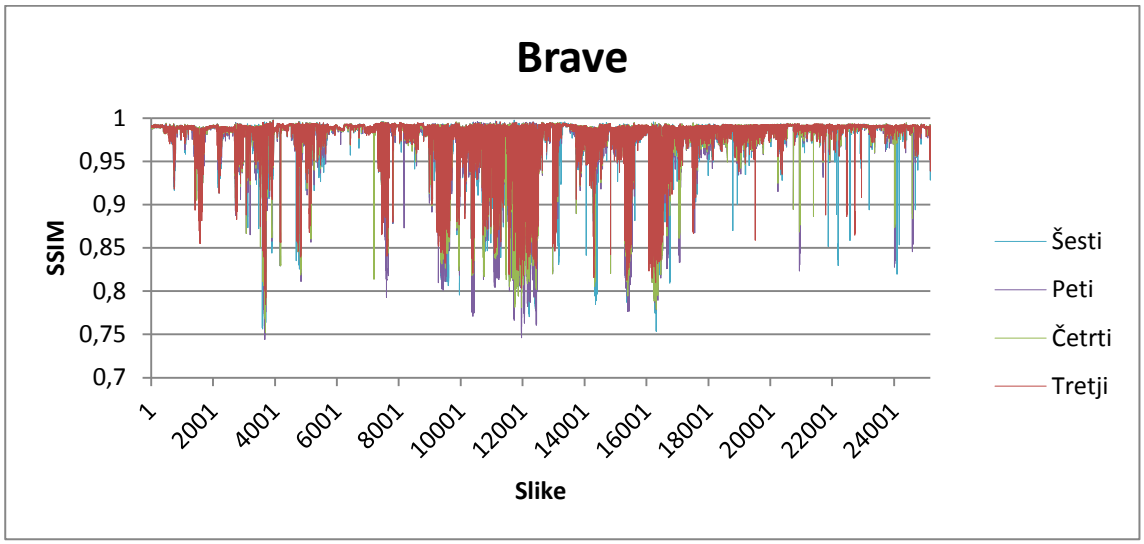
3.3.2 Kodiranje vzorca "*Brave*"

Analiza vzorcev *Brave* pokaže razmeroma porazno obnašanje vseh preizkusov in do 25% slabšo kakovost slike napram izvirniku (slika 24).



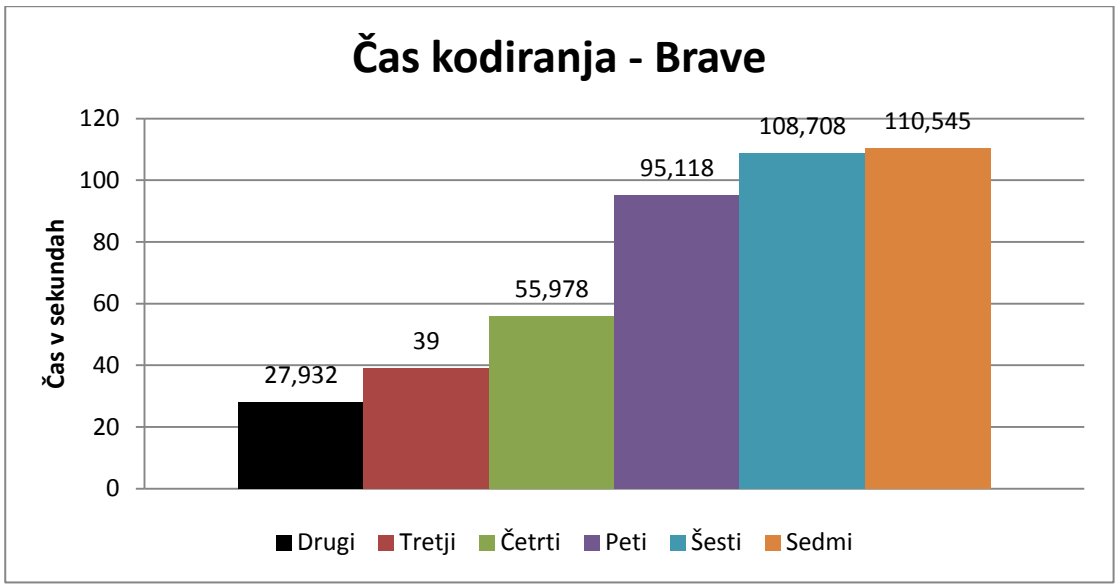
Slika 24: SSIM ocena vseh preizkusov podobnosti slike za vzorec *Brave.*

Po kvaliteti slike sta se najbolje odrezala tretji ter četrti preizkus, malenkostno slabša sta bila peti in šesti. Hitre spremembe scen, veliko število barv in gibanja ter jasni robovi likov so tu za nastavljene omejitve v bitni hitrosti predstavljali prevelik zalogaj, kar je privedlo do znatnega znižanja kakovosti končne slike (slika 25).



Slika 25: SSIM ocena primernih preizkusov podobnosti slike za vzorec *Brave*.

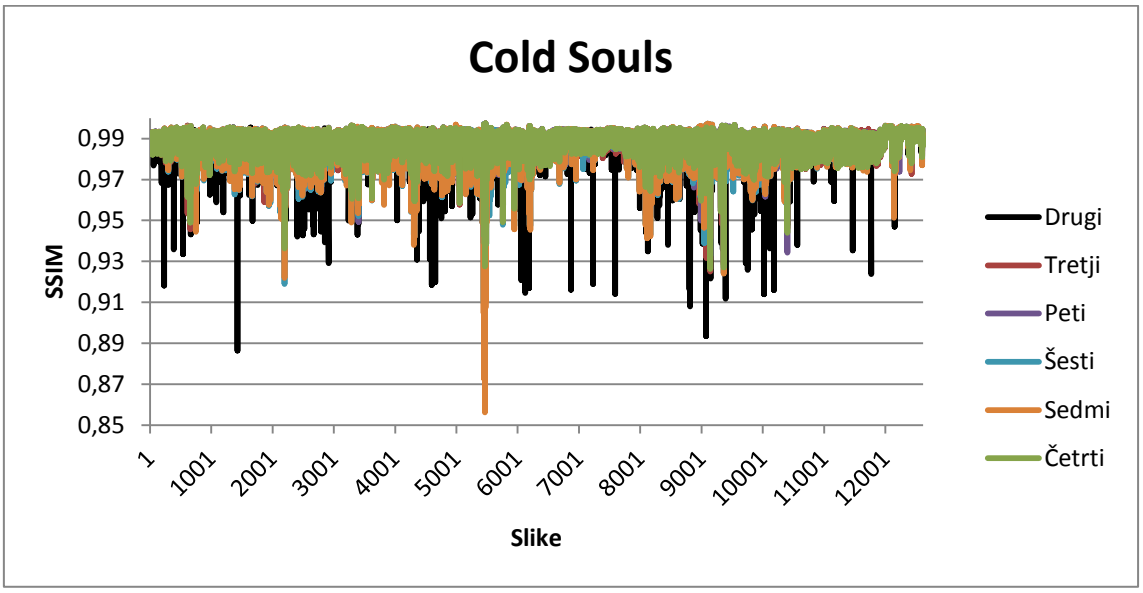
Izmed primernih kandidatov sta tretji in četrti preizkus delo zaključila najhitreje (slika 26). Bolj časovno potratni preizkusi niso dosegli znatno boljše kakovosti slike, da bi upravičili njihovo izbiro. Tretji preizkus je delo opravil hitro ter z najboljšo kakovostjo slike. Četrti preizkus mu po kakovosti tesno sledi, a z rahlo daljšim časom obdelave.



Slika 26: Graf časa obdelave vzorcev *Brave*.

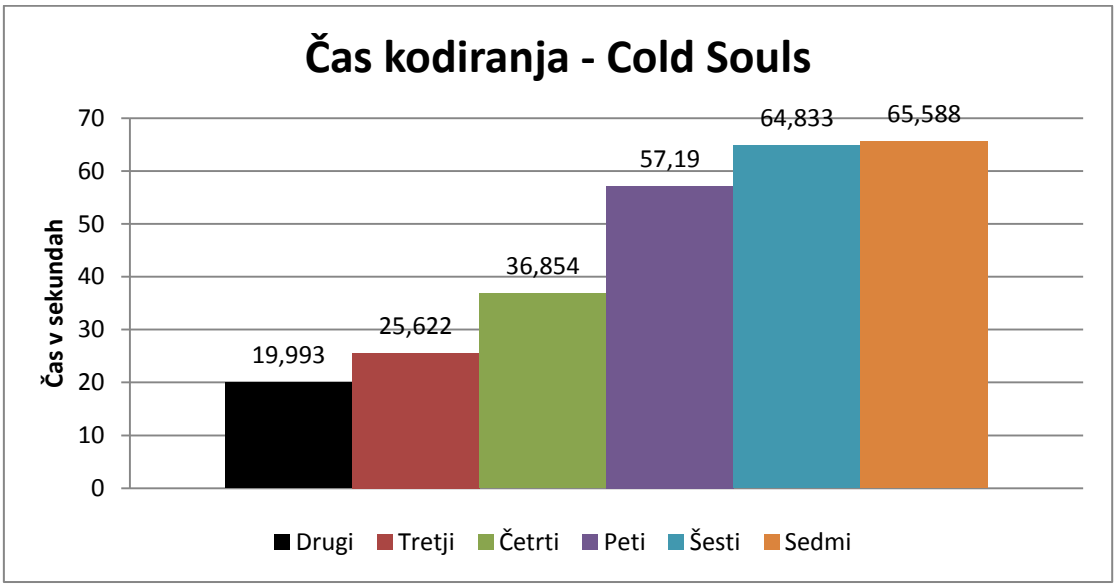
3.3.3 Kodiranje vzorca "Cold Souls"

Analiza prvega filmskega vzorca pokaže zadovoljive rezultate pri skoraj vseh preizkusih (slika 27). Večje odstopanje je opazno pri sedmem in drugem preizkusu, preostali pa se prekrivajo z izjemo točkovnih razlik. Najbolje sta se odrezala tretji ter četrti preizkus s povprečno 97% podobnostjo glede na izvirnik.



Slika 27: SSIM ocena vseh preizkusov podobnosti slike za vzorec *Cold Souls*.

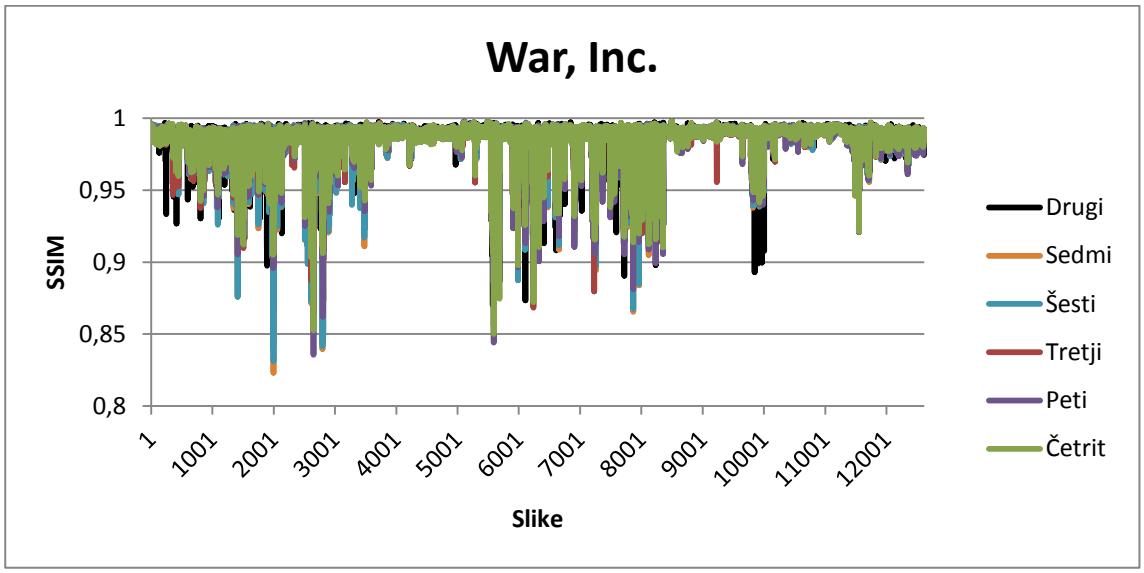
Tretji in četrti preizkus sta se časovno pričakovano odrezala najbolje (slika 28). Ostali preizkusi imajo daljši čas obdelave, ter skoraj enako kvaliteto slike kot tretji in četrti, zato ne upravičijo uporabe.



Slika 28: Graf časa obdelave vzorcev Cold Souls.

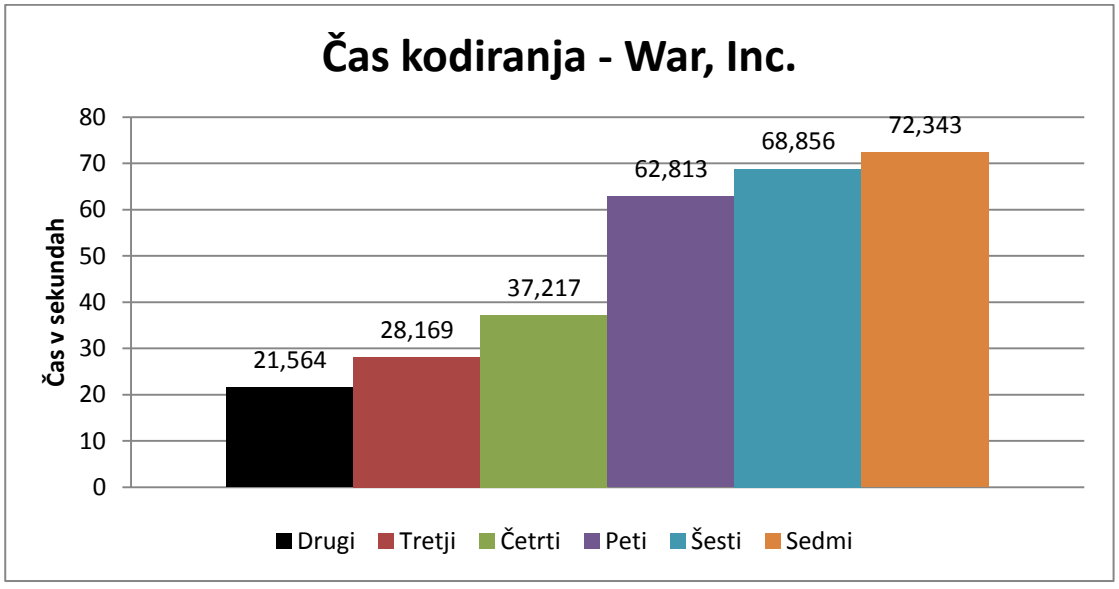
3.3.4 Kodiranje vzorca "War, Inc."

V našem drugem filmskem vzorcu so se preizkusi pričakovano obnesli malenkostno slabše zaradi hitrega gibanja in rezov (slika 29). Vsi preizkusi z razliko nekaterih točkovnih odklonov so se obnesli podobno ter v povprečju dosegli 94% podobnost napram izvirniku.



Slika 29: SSIM ocena vseh preizkusov podobnosti slike za vzorec *War, Inc.*

S hitrostjo obdelave ter kakovostjo zapisa pričakovano tudi pri tem vzorcu vodita tretji ter četrti preizkus (slika 30).



Slika 30: Graf časa obdelave vzorcev War, Inc.

3.3.5 MPEG2 kodirnik- zaključek

Pri izbranih vzorcih smo lahko opazili, da je kvaliteta končnega zapisa ter njegova podobnost napram izvirniku dosegla zlato sredino pri preizkusu z uporabo B okvirjev ter pri preizkusu, kjer smo uporabili B okvirje skupaj s trellis kvantizacijo.

Trelis kvantizacija je najbolj prišla do izraza pri risanem vzorcu *The Secret World of Arrietty*, kjer je bilo moč opaziti razliko med 85% podobnostjo brez uporabe, ter 98% podobnostjo napram izvirniku z uporabo trellis kvantizacije. Z manjšo časovno razliko v obdelavi med obema nizoma argumentov smo lahko dosegli dramatično izboljšanje v kakovosti slike.

Bolj napredna argumenta, kot sta mehanizem za izbiro makro blokov in algoritem za iskanje gibanja, sta se izkazala za zelo specifična pri različnih vrstah video vsebin, zato bi jih bilo potrebno nastavljati glede na vrsto le-te.

Za potrebe naše aplikacije po generičnem kodiranju, ki zajame veliko večino vsebin, so se za najbolj primernega izkazali argumenti četrtega preizkusa.

3.4 Izbira parametrov za kodirnik x264

Na podlagi ugotovitev MPEG2 kodirnika o kompleksnosti naprednih argumentov, smo za osnovo argumentov x264 kodirnika vzeli prednastavljena profila *fast* ter *faster* za kodiranje prvih vzorcev. Profila imata dodana argumenta za razmik ključnih slik ter omejitev bitne hitrosti.

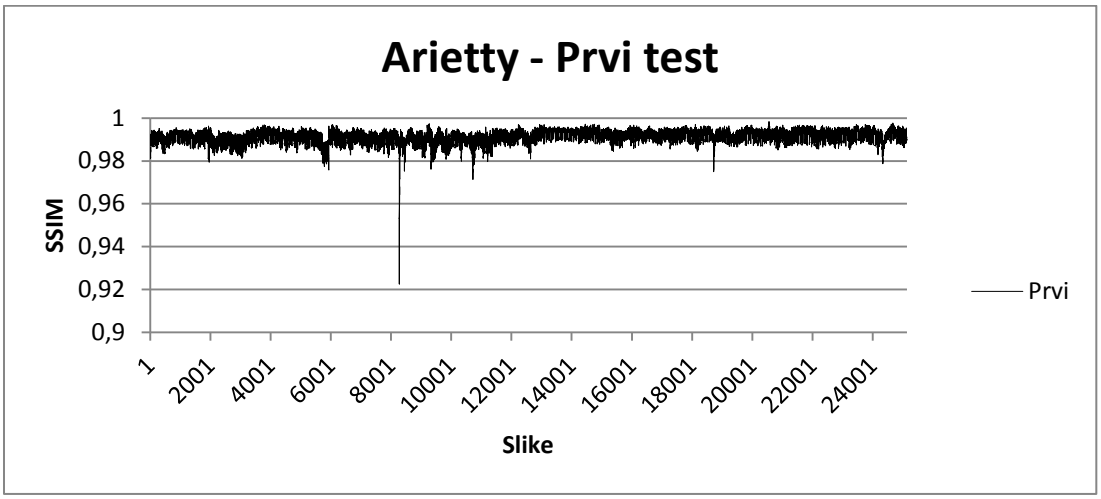
Prvi preizkus smo izvedli s prednastavljenim profilom *fast* parametri [16], omejili smo bitno hitrost s primerno toleranco z argumenti -**b**, **vbv-maxrate** ter **vbv-bufsize** in nastavili razmik ključnih slik z argumentom **keyint**. Z argumenti -**a none** smo onemogočili obdelavo zvočne sledi in s -**f** nastavili izhod kot mkv datoteko. Nastavitev jeder, ki jih ima orodje na razpolago, ni bilo potrebno nastavljati, ker orodje *HandBrake* to počne samodejno.

```
time HandBrakeCLI -i brave-copy.vob -o brave-copy.vob.mkv -f mkv --crop 0:0:0:0 --strict-anamorphic -e x264 -b 2500 -x264-preset faster -a none -x vbv-maxrate=2500:vbv-bufsize=1000:keyint=25
```

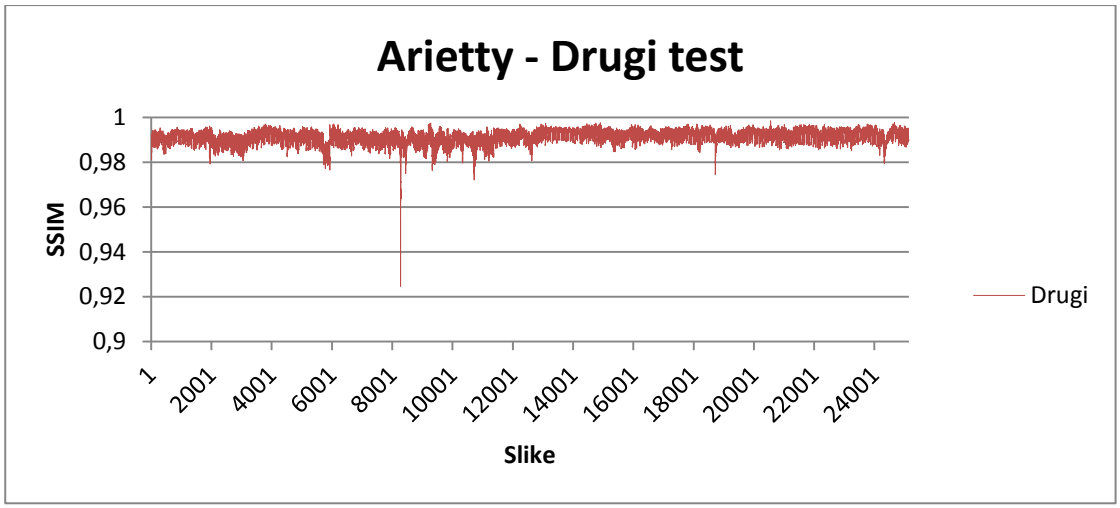
Drugi preizkus smo izvedli s spremenjenim profilom *faster*, ostalih argumentov nismo spreminjali.

3.4.1 Kodiranje vzorca "The Secret World of Arrietty"

Analiza vzorcev *The Secret World of Arrietty* nam pokaže le malenkostne razlike, ki so opazne šele v tretji decimalki v podobnosti napram originalni sliki med obema preizkusoma (sliki 31 in 32). Kljub nižji bitni hitrosti kodirnik x264 obdrži več kot 98% podobnost napram izvorni sliki.

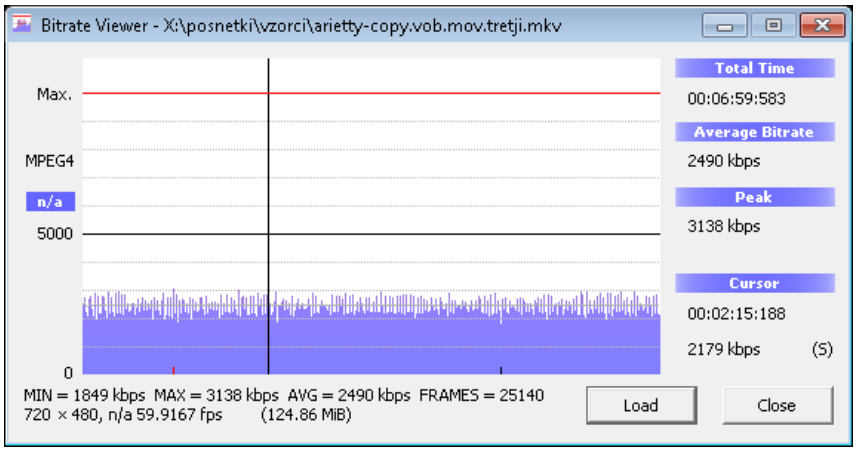


Slika 31: SSIM ocena prvega h264 preizkusa podobnosti slike za vzorec *The Secret World of Arrietty.*



Slika 32: SSIM ocena drugega h264 preizkusa podobnosti slike za vzorec *The Secret World of Arrietty.*

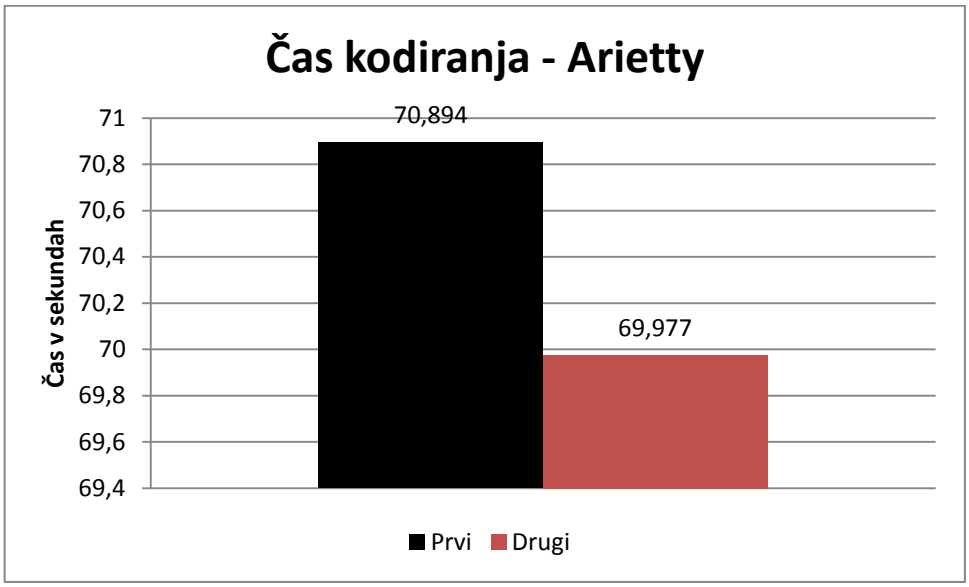
Na sliki 33 je prikazana bitna hitrost vsebine po kodiranju v zapis h264.



Slika 33: Bitna hitrost h264 profila faster za vzorec *The Secret World of Arrietty.*

S slike grafa bitne hitrosti lahko razberemo, da bitna hitrost s povprečjem 2490 kilobitov na sekundo in z najvišjo vrednostjo 3138 kilobitov na sekundo, primerno upošteva podane omejitve.

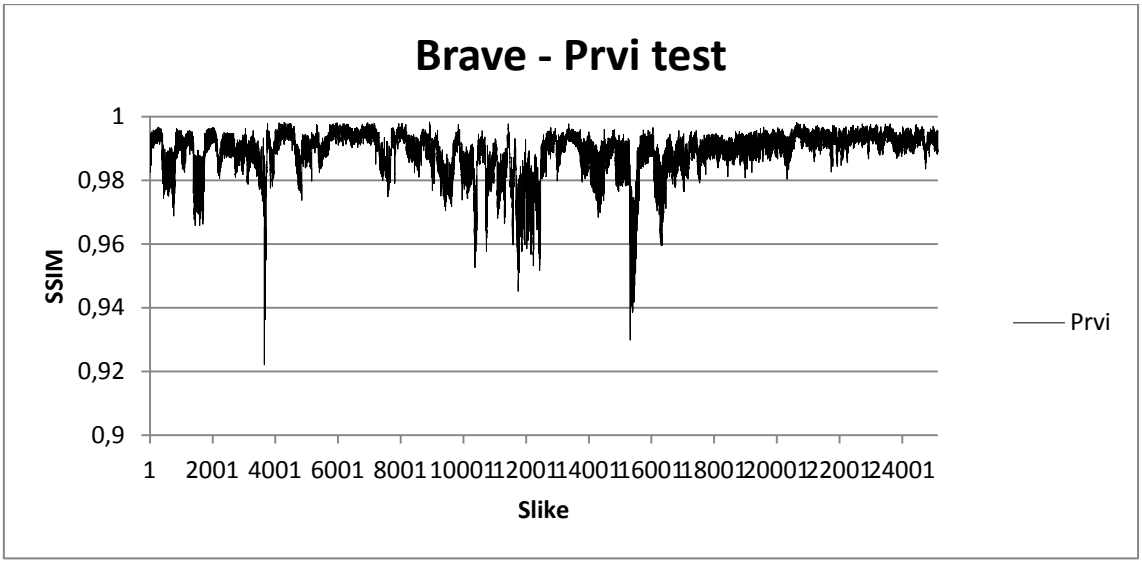
Razlika v času, potrebnem za kodiranje z različnima profiloma, je pri tem vzorcu zanemarljiva (slika 34). Čas kodiranja je napram kodirniku MPEG2 daljši za 43%, kar je glede na uporabo nekaterih naprednejših algoritmov razumljivo.



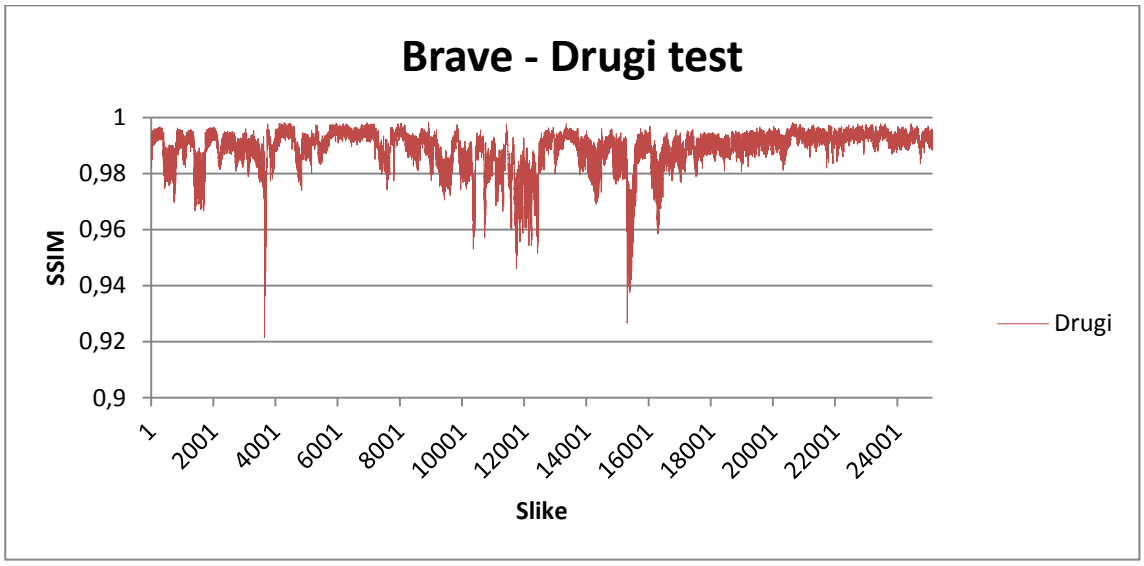
Slika 34: Graf časa potrebnega za kodiranje v h264 vzorca *The Secret World of Arrietty*.

3.4.2 Kodiranje vzorca "*Brave*"

Pri izseku iz vsebine *Brave* med preizkusoma ponovno opazimo zanemarljivo razliko v podobnosti slike (sliki 25 in 26). Kodirnik x264 se je v povprečju z več kot 96% podobnostjo obnesel neprimerno bolje, kot kodirnik MPEG2 pri enakem vzorcu.

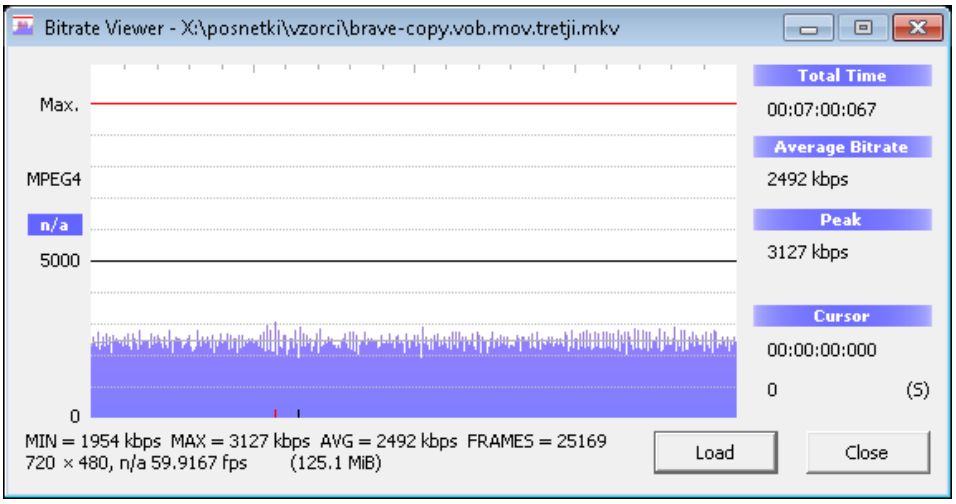


Slika 35: SSIM ocena prvega h264 preizkusa podobnosti slike za vzorec Brave.



Slika 36: SSIM ocena drugega h264 preizkusa podobnosti slike za vzorec Brave.

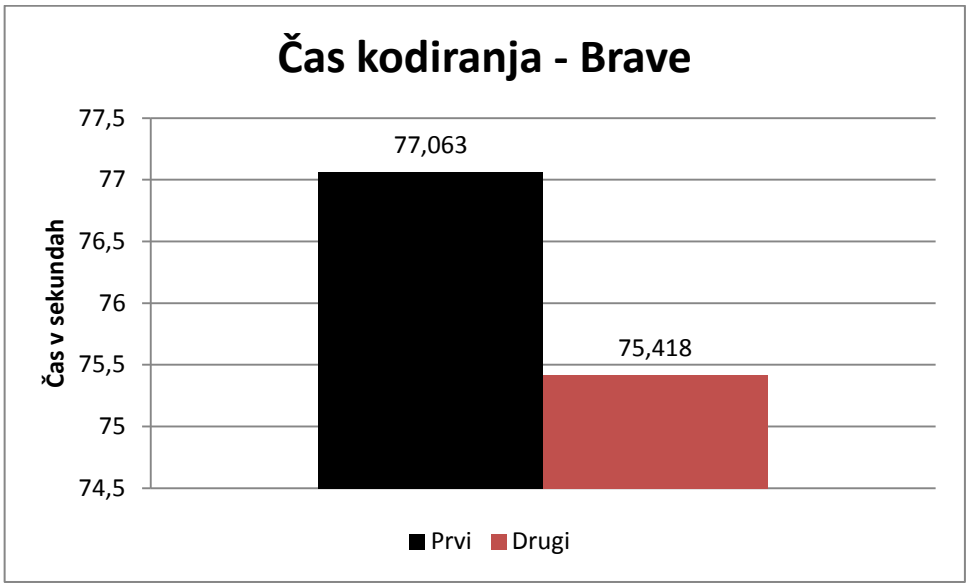
Na sliki 37 je prikazana bitna hitrost vsebine po kodiranju v zapis h264.



Slika 37: Bitna hitrost h264 profila faster za vzorec Brave.

S slike grafa bitne hitrosti lahko razberemo, da bitna hitrost s povprečjem 2492 kilobitov na sekundo in z najvišjo vrednostjo 3127 kilobitov na sekundo, primerno upošteva podane omejitve.

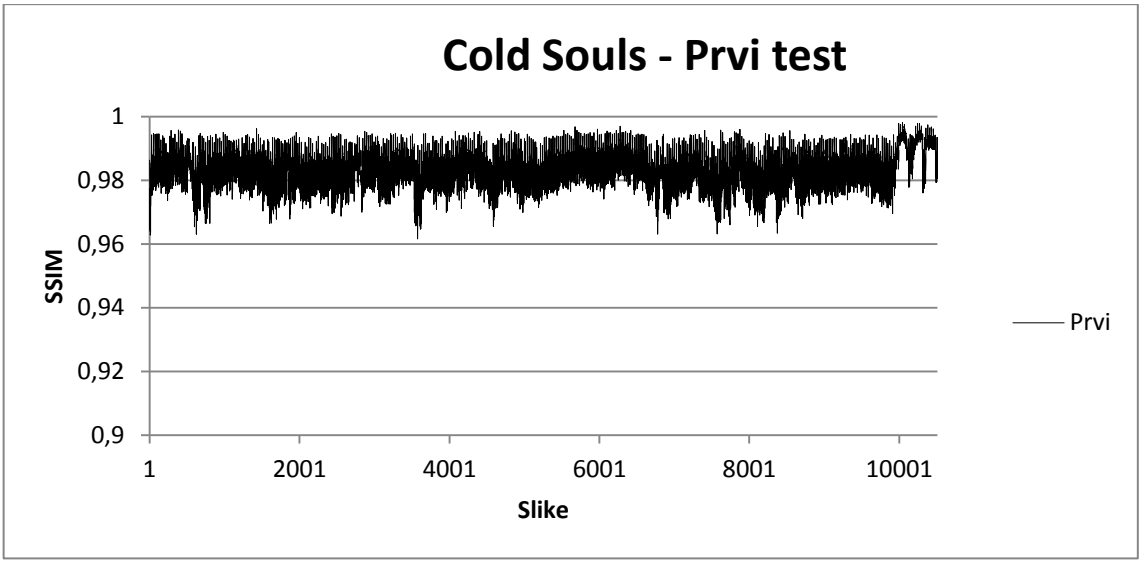
Ponovno opazimo za 37% daljšo obdelavo video vzorca napram MPEG2 kodirniku ter nekoliko večjo razliko med profilom *fast* in *faster*, a je še vedno zanemarljiva (slika 38).



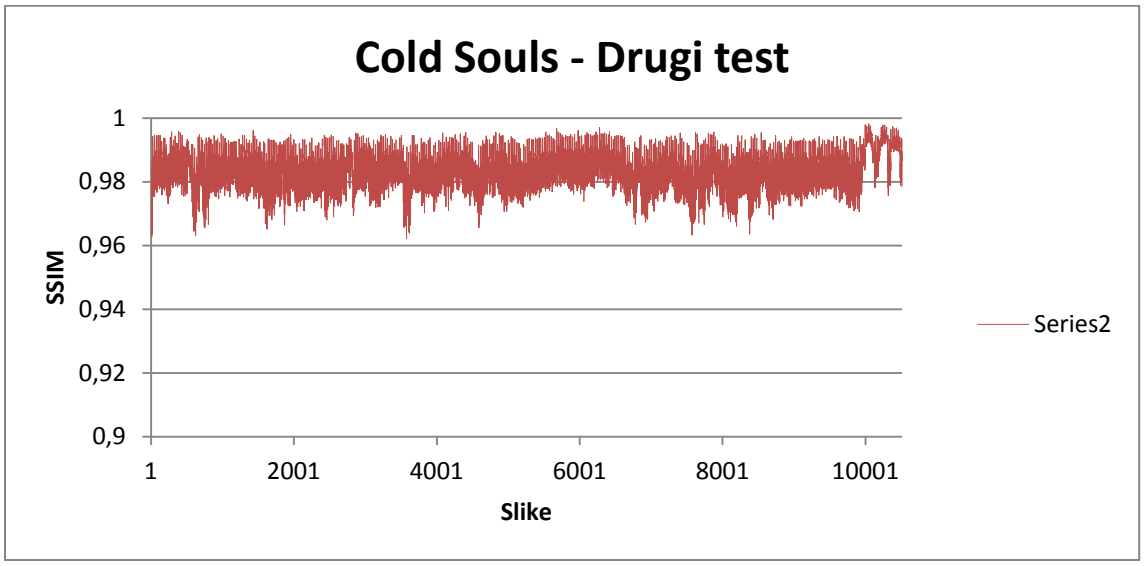
Slika 38: Graf časa potrebnega za kodiranje v h264 vzorca Brave.

3.4.3 Kodiranje vzorca "Cold Souls"

Z analizo podobnosti slike napram izvirniku lahko hitro ugotovimo, da se je kljub počasnim premikom ter večinoma statični sliki, kodirnik MPEG2 mestoma obnesel malenkostno slabše kot kodirnik h264 (sliki 39 in 40).

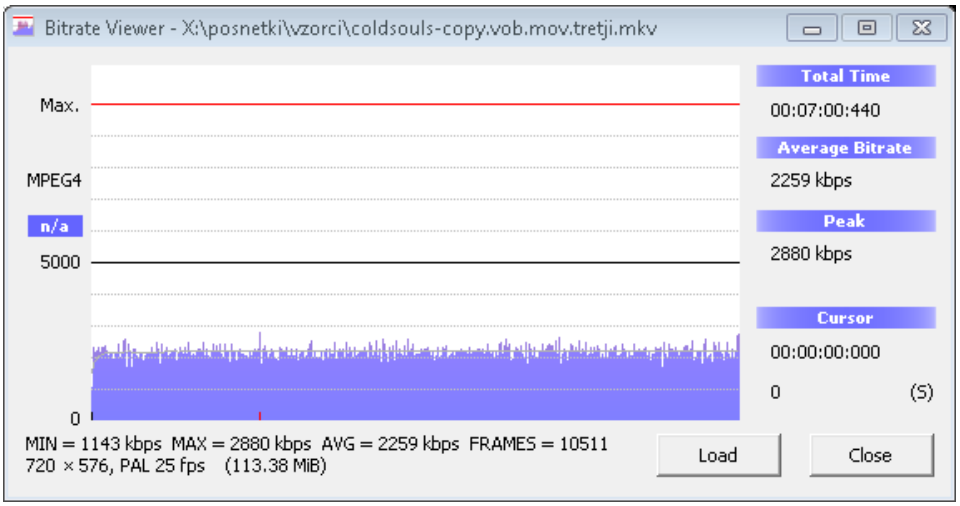


Slika 39: SSIM ocena prvega h264 preizkusa podobnosti slike za vzorec Cold Souls.



Slika 40 - SSIM ocena drugega h264 preizkusa podobnosti slike za vzorec Cold Souls.

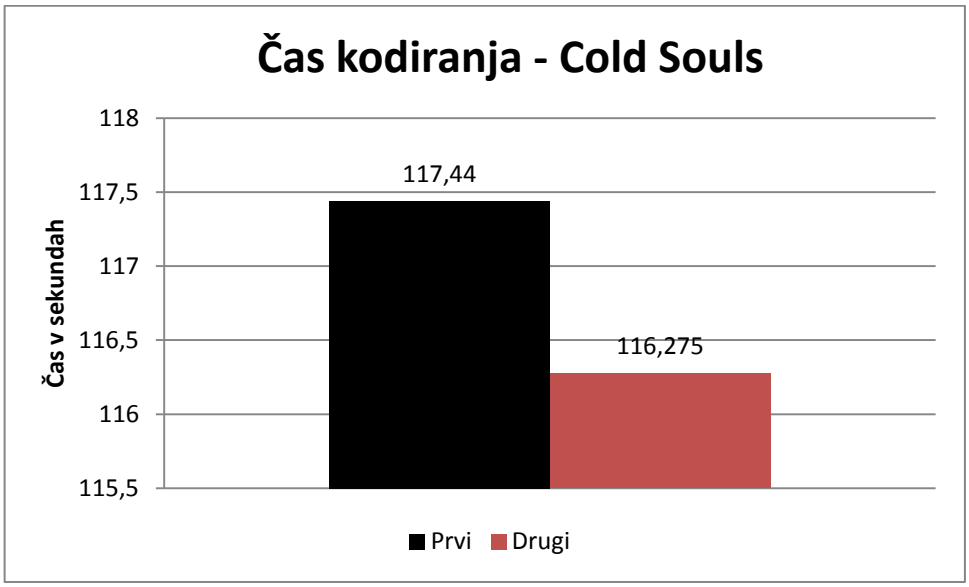
Na sliki 41 je prikazana bitna hitrost vsebine po kodiranju v zapis h264.



Slika 41: Bitna hitrost h264 profila faster za vzorec Cold Souls.

S slike grafa bitne hitrosti lahko razberemo, da bitna hitrost s povprečjem 2259 kilobitov na sekundo in z najvišjo vrednostjo 2880 kilobitov na sekundo, primerno upošteva podane omejitve.

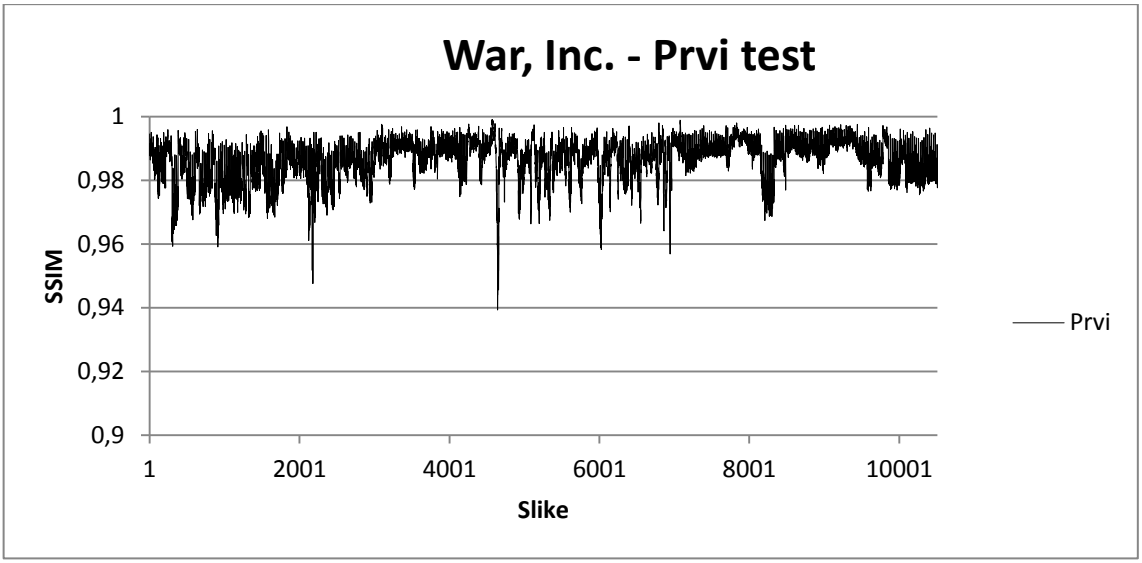
Presenetljivo je kodirnik x264 za obdelavo filmskega video posnetka potreboval dlje časa (slika 42), kot za risane in animirane vzorce, kar je ravno nasprotno s primerjavo kodirnika MPEG2. Kodirnik x264 vsebino obdela za faktor 3,6 hitreje od realnega časa, kar je znatno počasneje napram faktorju 11,6 pri kodirniku MPEG2.



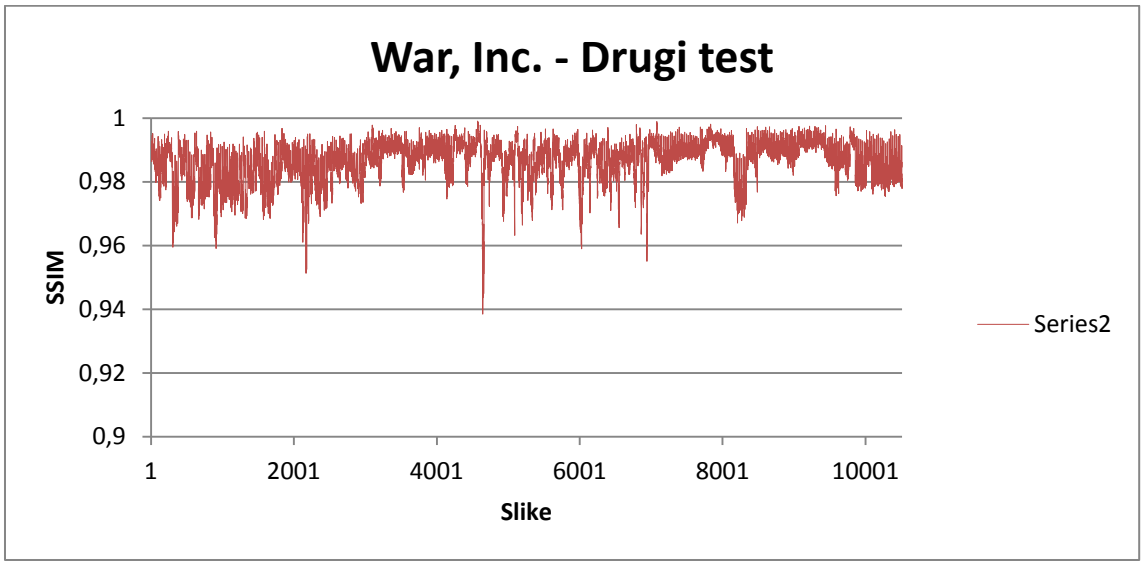
Slika 42: Graf časa potrebnega za kodiranje v h264 vzorca Cold Souls.

3.4.4 Kodiranje vzorca "War, Inc."

Naš hitri akcijski video vzorec je kodirnik x264 obdelal z največ 6% odklonom v podobnosti slike napram izvirnem vzorcu in nalogo opravil veliko bolje kot kodirnik MPEG2, ki je imel 15% odklon. Opazne razlike med profiloma *fast* in *faster* ni moč opaziti (sliki 43 in 44).

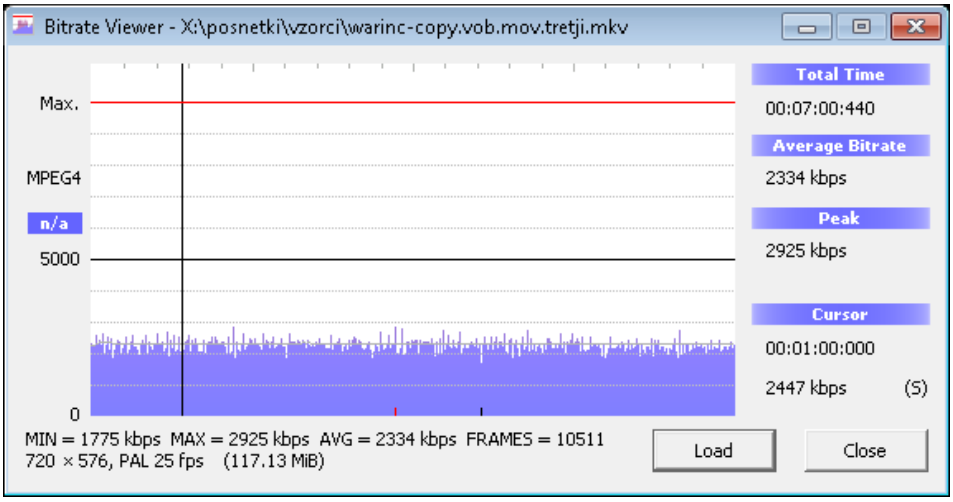


Slika 43: SSIM ocena prvega h264 preizkusa podobnosti slike za vzorec War, Inc.



Slika 44: SSIM ocena drugega h264 preizkusa podobnosti slike za vzorec War, Inc.

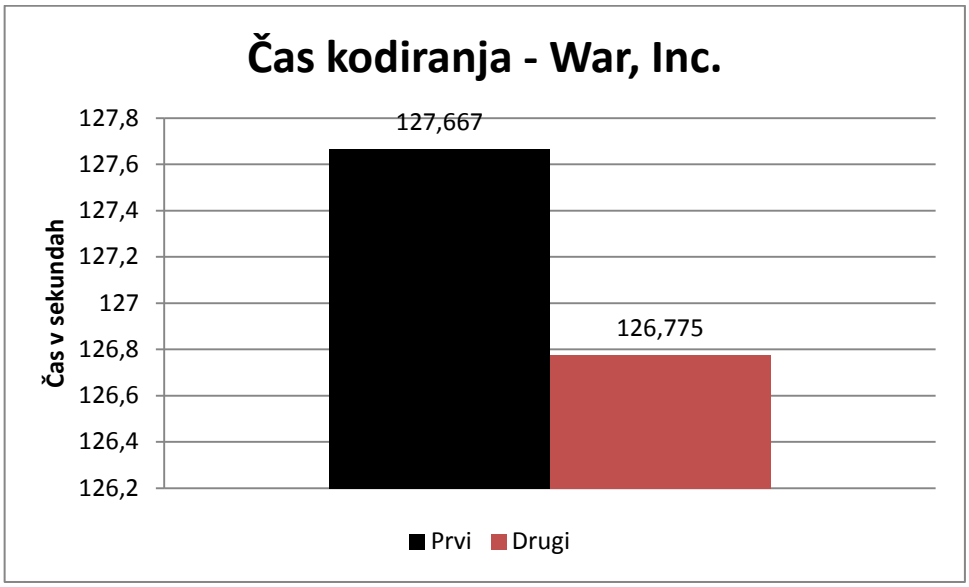
Na sliki 45 je prikazana bitna hitrost vsebine po kodiranju v zapis h264.



Slika 45: Bitna hitrost h264 profila faster za vzorec War, Inc.

S slike grafa bitne hitrosti lahko razberemo, da bitna hitrost s povprečjem 2334 kilobitov na sekundo in z najvišjo vrednostjo 2925 kilobitov na sekundo, primerno upošteva podane omejitve.

Čas obdelave je znatno daljši v primerjavi s kodirnikom MPEG2, kar smo že opazili pri prejšnjem vzorcu, saj so algoritmi za iskanje gibanja izredno procesorsko potratni pri scenah s hitrim gibanjem in velikim dinamičnim razponom barv. Kodirnik x264 vsebino obdela za faktor 3,3 hitreje od realnega časa, pri največ 6% odklonu, kar je še vedno sprejemljivo za našo uporabo (slika 46).



Slika 46: Graf časa potrebnega za kodiranje v h264 vzorca War, Inc.

3.4.5 h264 kodirnik- zaključek

Pri izbranih vzorcih smo lahko opazili, da se kvaliteta končnega zapisa ter njegova podobnost napram izvirniku z uporabo profila *fast* in *faster* praktično ne razlikuje.

Težav, z omejevanjem bitne hitrosti, ki so se pojavljale pri kodiranju v MPEG2, tu ni bilo moč opaziti. Vsi so se določene meje držali izredno striktno. Kljub polovični omejitvi bitne hitrosti, čeprav na račun daljše obdelave, smo s kodirnikom x264 dosegli večjo in bolj konstantno podobnost slike kot s kodirnikom MPEG2.

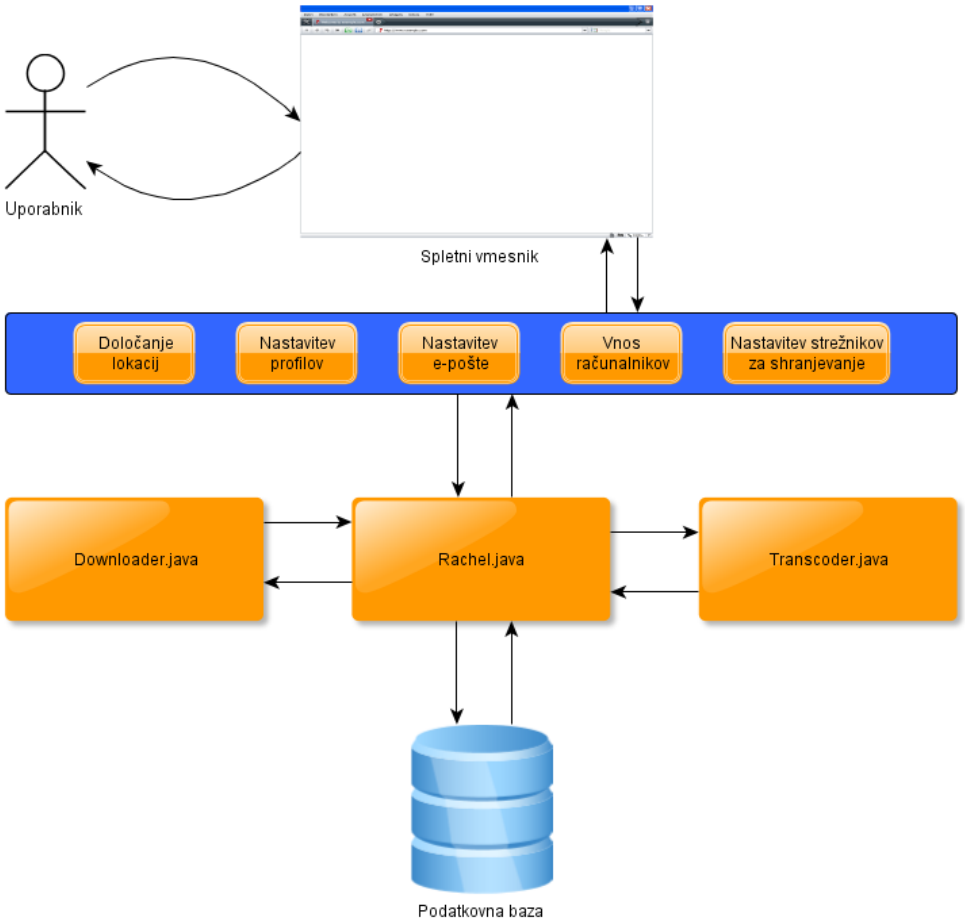
Z visokim zaupanjem lahko trdimo, da bo kodirnik x264 lahko brez težav znotraj omejitev, pogojenih s prenosnim medijem, zagotovil visoko kvaliteto slike z malo ali nič popačenja napram izvirniku, ter je tako primeren kandidat za prihodnjo zamenjavo kodirnika MPEG2 tudi pri višjih bitnih hitrostih.

4 Razvoj aplikacije

V tem poglavju je opisana arhitektura aplikacije, statično prevajanje kodirnikov in predstavljen spletni grafični vmesnik. Podrobneje je opisan potek izvajanja opravil aplikacije ter zgradba podatkovne baze.

4.1 Arhitektura

Aplikacija je zgrajena iz štirih poglavitnih delov ali razredov (slika 47). To so spletni uporabniški vmesnik Webgui.java, ki omogoča administracijo aplikacije, glavni razred Rachel.java, ki skrbi za podatkovno bazo ter razporejanje vsebin za obdelavo, Downloader.java, ki skrbi za prenos vsebin z oddaljenih strežnikov ter Transcoder.java, ki vsebine obdela glede na določen profil.



Slika 47: Strukturni diagram aplikacije.

Uporabnik neposredno komunicira samo s primarnim računalnikom, na katerem teče instanca aplikacije s spletnim vmesnikom. Preostala komunikacija poteka samodejno v ozadju med posameznimi instancami aplikacije.

4.2 Statično prevajanje orodij FFmpeg in HandBrake

Eden izmed ciljev naše aplikacije je popolna prenosljivost med operacijskimi sistemi in strojno opremo. Da zagotovimo delovanje orodij FFmpeg in HandBrake na različnih linux distribucijah brez vseh potrebnih odvisnosti, je bilo potrebno orodji statično prevesti. Prevajanje je potekalo na distribuciji Ubuntu 12.04 LTS.

4.2.1 FFmpeg

V prvem koraku smo namestili vse potrebne knjižice ter orodja za prevajanje programske kode z ukazom:

```
apt-get -y install build-essential checkinstall git libfaac-dev
libjack-jackd2-dev libmp3lame-dev libopencore-amrnb-dev yasm
libsdl1.2-dev libopencore-amrwb-dev libtheora-dev libva-dev
libvdpau-dev libvorbis-dev libx11-dev libxfixes-dev texi2html
zlib1g-dev
```

Po končani namestitvi potrebnih orodji in knjižnic smo iz git repozitorja prenesli zadnjo različico x264 kodirnika za obdelavo vhodnih vsebin in morebitno kodiranje v zapis h264. S spodnjim zaporedjem ukazov smo prenesli izvorno kodo kodirnika x264, jo statično prevedli, ter namestili:

```
git clone git://git.videolan.org/x264
./configure --enable-static
make
```

V zadnjem koraku smo prenesli še izvorno kodo orodja FFmpeg, ter jo statično prevedli s spodnjim zaporedjem ukazov:

```
git clone --depth 1 git://source.ffmpeg.org/ffmpeg

./configure --extra-version=0.10.2-fora --arch=amd64 --
prefix=/opt/ffmpeg --enable-gpl --enable-libfaac --enable-libmp3lame
--enable-libopencore-amrnb --enable-libopencore-amrwb --enable-
libtheora --enable-libvorbis --enable-libx264 --enable-nonfree --
enable-version3 --disable-shared --enable-static --extra-libs=-static
--extra-cflags=--static --disable-ffserver --disable-ffplay

make
make install
```

Po končanem prevajanju se v mapi ffmpeg/bin nahaja statično prevedena binarna izvršljiva datoteka *ffmpeg*, s katero bo naša aplikacija opravljala kodiranje v MPEG2.

4.2.2 HandBrake

Pri orodju za kodiranje v zapis h264 smo prevedli samo različico za uporabo v ukazni lupini HandBrakeCLI. S spletne strani orodja smo prenesli datoteko z izvorno kodo HandBrake-0.9.6.tar.bz2, ki smo jo nato prevedli z argumentom, ki onemogoči prevajanje grafičnega vmesnika:

```
./configure --disable-gtk
cd build
make -j4
```

Zgornje zaporedje ukazov nam je prevedlo dinamično povezano binarno datoteko orodja. Za statično različico je bilo potrebno izvršiti še prevajanje z dodanim argumentom "-static" v ukazu:

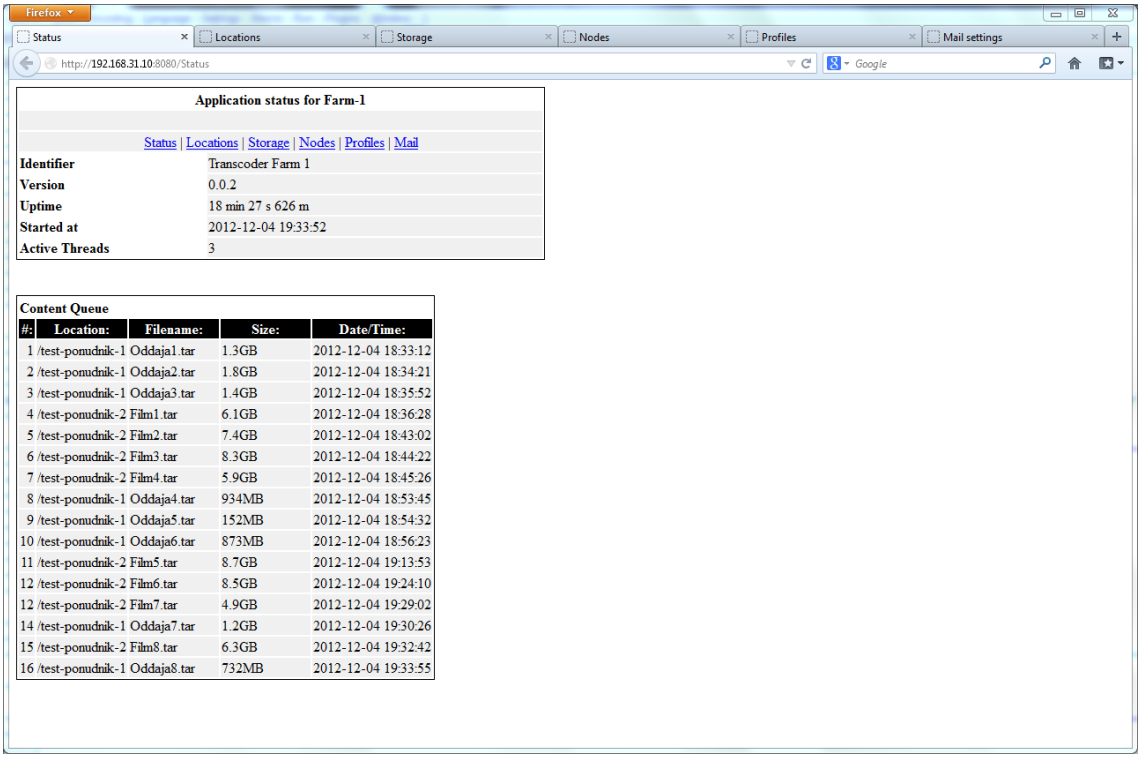
```
/usr/bin/g++ -pipe -Wl,-S -fmessage-length=0 -Wall -g0 -O3 -I./libhb/
-I./contrib/include -o HandBrakeCLI -Wl,--start-group
test/parsecsv.o test/test.o ./libhb/libhb.a ./contrib/lib/liba52.a
./contrib/lib/libass.a ./contrib/lib/libavcodec.a
./contrib/lib/libavformat.a ./contrib/lib/libavutil.a
./contrib/lib/libdca.a ./contrib/lib/libdvdnav.a
./contrib/lib/libdvdread.a ./contrib/lib/libfaac.a
./contrib/lib/libfontconfig.a ./contrib/lib/libfreetype.a
./contrib/lib/libmkv.a ./contrib/lib/libmpeg2.a
./contrib/lib/libmp3lame.a ./contrib/lib/libmp4v2.a
./contrib/lib/libogg.a ./contrib/lib/libsamplerate.a
./contrib/lib/libswscale.a ./contrib/lib/libtheora.a
./contrib/lib/libvorbis.a ./contrib/lib/libvorbisenc.a
./contrib/lib/libx264.a ./contrib/lib/libxml2.a
./contrib/lib/libbluray.a -lbz2 -lz -lfribidi -lpthread -ldl -lm -
Wl,--end-group -static
```

4.3 Grafični vmesnik

Aplikacija v primeru, da smo jo določili kot primarno instanco, ob izvajanju v ločeni niti požene preprost spletni vmesnik, preko katerega lahko uporabnik nastavlja lastnosti celotnega sistema.

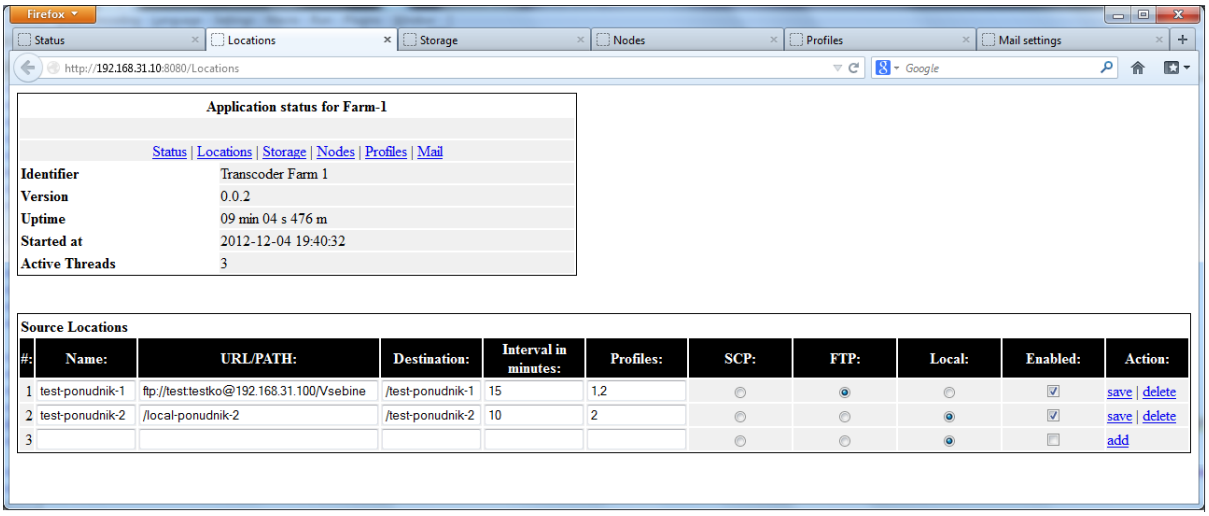
Osnovni zaslon na sliki 48 nam omogoča hitri pregled stanja aplikacije. Podan je čas izvajanja aplikacije, datum in ura zagona, število aktivnih niti, ter pregled vsebin v čakalni vrsti. Vsaka vsebina ima podane informacije o imenu datoteke, polno pot do njenega mesta na

datotečnem sistemu, velikost datoteke ter datum in uro, ko je bila vsebina ustvarjena ali naložena. Vse vsebine so v vrsti razvrščene naraščajoče, saj se tako vsebine obdelajo v enakem vrstnem redu, kot so bile naložene.



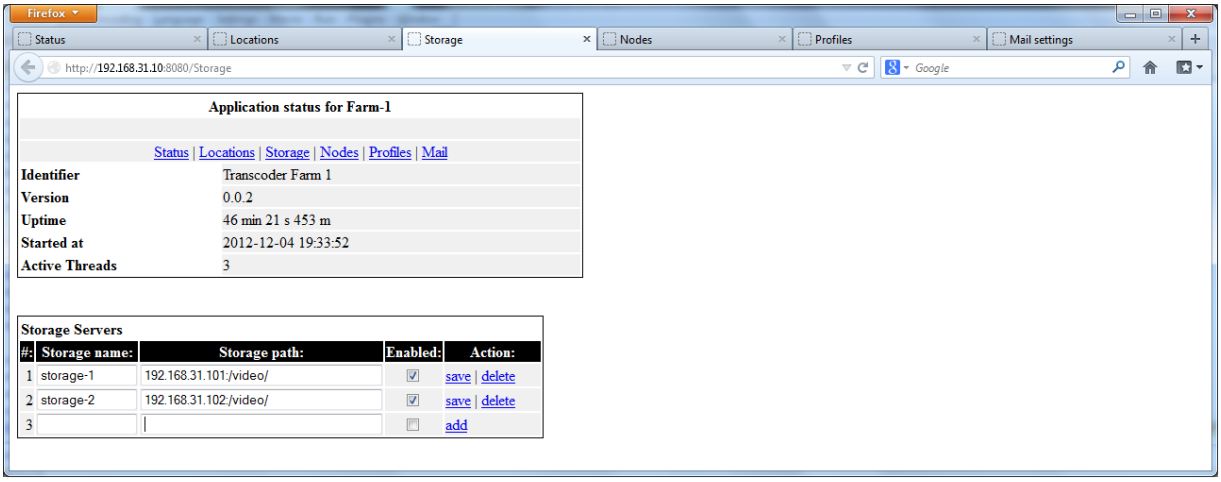
Slika 48: Pregled stanja aplikacije.

V zaslonu »locations« na sliki 49 so uporabniku na voljo nastavitve oddaljenih mest, katere aplikacija spremlja ter prenaša nove vsebine na lokalni datotečni sistem. Vsakemu mestu je potrebno določiti ime in polno pot do mesta, ki ga želimo spremljati. V primeru oddaljenih mest preko protokola FTP uporabniško ime in geslo določimo neposredno v URL (npr. ftp://uporabniškoime:geslo@strežnik), polno pot na lokalnem datotečnem sistemu, kamor naj aplikacija vsebine odlaga, časovni interval, v katerem naj aplikacija preverja za nove vsebine, identifikacijske številke profilov ločene z vejico, s katerimi naj aplikacija vsebine s tega mesta obdela ter način dostopa do oddaljenega mesta. Na voljo so protokoli FTP, SCP ter lokalni datotečni sistem. Vsako posamezno lokacijo lahko tudi po potrebi onemogočimo.



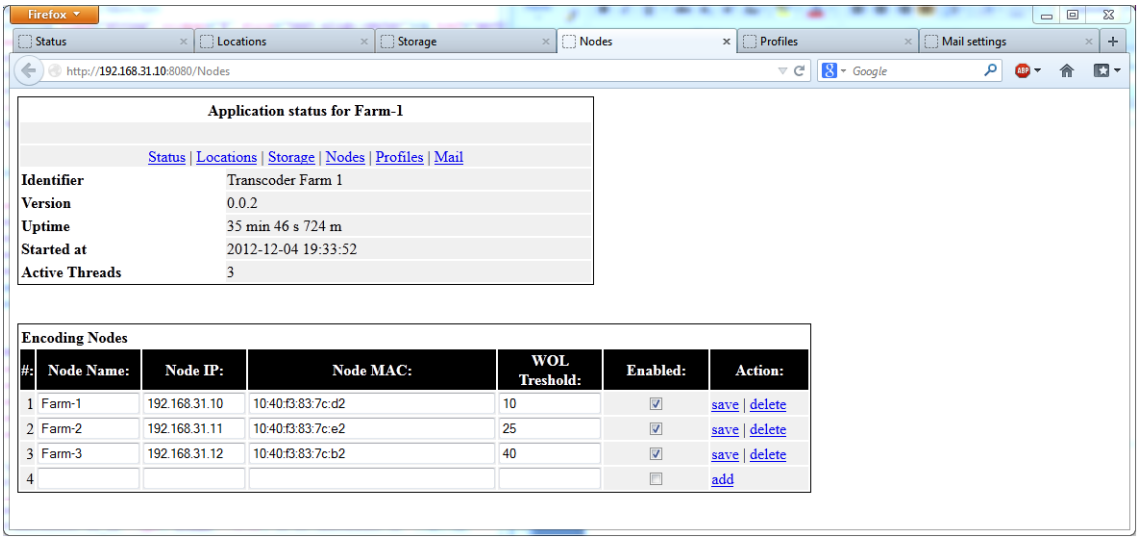
Slika 49: Določanje oddaljenih virov datotek.

Zaslon »Storage« na sliki 50 uporabniku nudi zmožnost nastavljati datotečne strežnike, kamor se bodo končane vsebine po obdelavi prenesle. Vsakemu datotečnemu strežniku je potrebno določiti ime, ki se bo uporabljalo za enolično identifikacijo mesta, kjer se vsebine nahajajo ter pot do datotečnega strežnika, ki vključuje IP naslov strežnika ter polno pot do mesta, kjer naj aplikacija vsebine odloži. Vsako mesto je mogoče onemogočiti, če želimo začasno prekiniti odlaganje nanj.



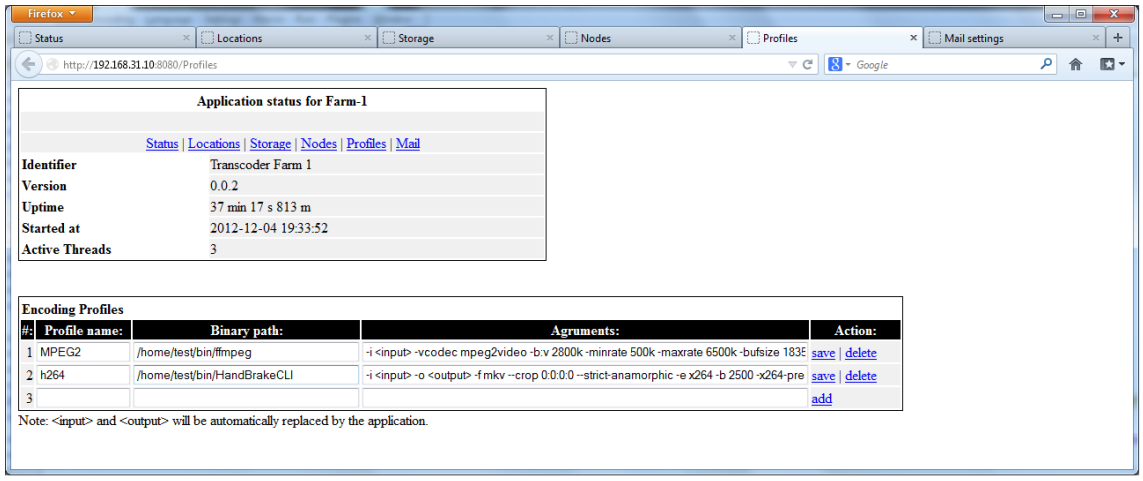
Slika 50: Nastavljanje datotečnih strežnikov.

Uporabnik na zaslonu »Nodes«, ki je prikazan na sliki 51, nastavlja dodatne računalnike, ki bodo vsebine obdelovali. Vsakemu računalniku je potrebno določiti enolično ime, naslov IP, ki se uporablja tudi za omejevanje dostopa do funkcij za distribucijo konfiguracijskih datotek, fizični MAC naslov mrežne kartice, katerega aplikacija uporablja za vklapljanje računalnikov preko omrežja ter prag števila vsebin v vrsti, nad katerim aplikacija vklopi dodatne računalnike oz. v primeru manjšega števila ugasne. Dodeljevanje vsebin ter zbujanje je mogoče posameznemu računalniku tudi onemogočiti.



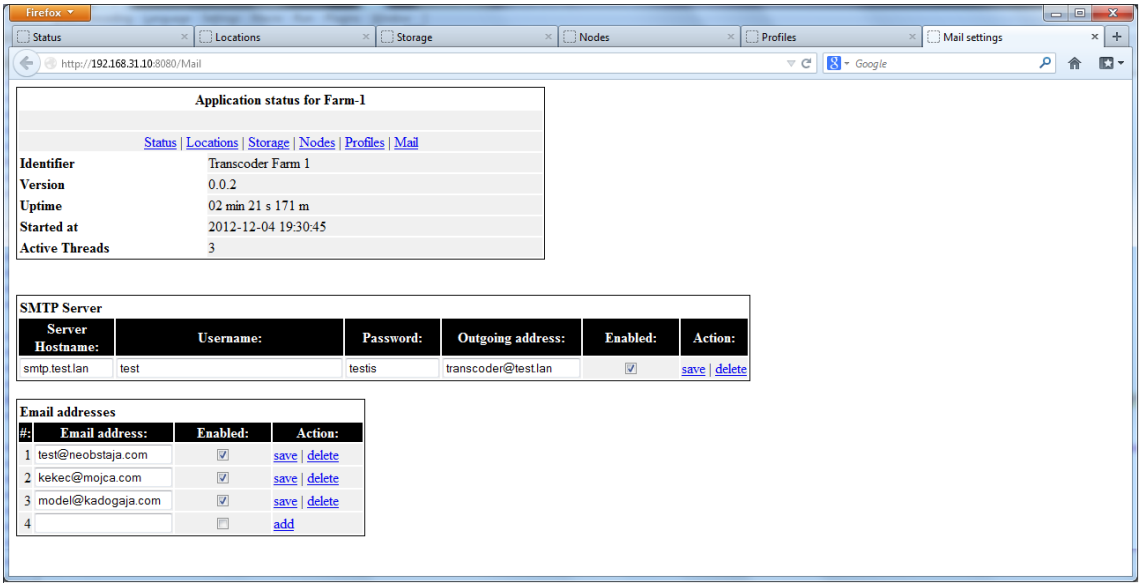
Slika 51: Nastavljanje računalnikov za obdelavo vsebin.

Profile, s katerimi aplikacija posamezno vsebino obdela, uporabnik določi na zaslonu »Profiles« (slika 52). Vsakemu profilu je potrebno podati ime, pot do izvršilne datoteke na lokalnem datotečnem sistemu ter argumente, ki se uporabijo pri klicu izvršilne datoteke. V polju za argumente izvršilni datoteki uporabnik mesto vhodne ter izhodne datoteke določi z značkama »<input>« ter »<output>«, katere aplikacija v času izvajanja samodejno nadomesti s primernimi imeni trenutne vsebine.



Slika 52: Nastavljanje profilov za kodiranje vsebin.

Za obveščanje o končanih vsebinah ter napakah, če se le te pripetijo, uporabnik na zaslonu »Mail«, ki je prikazan na sliki 53, nastavi odhodni poštni strežnik ter prejemnike sporočil. Poštni strežnik potrebuje določeno polno domensko ime ali naslov IP, uporabniško ime ter geslo, če je le-to zahtevano ter e-poštni naslov, s katerega naj aplikacija pošilja odhodna sporočila. Prejemnike je možno tudi začasno onemogočiti.



Slika 53: Nastavljanje poštnega strežnika.

4.4 Izvajanje aplikacije

4.4.1 Prenos nove vsebine

Aplikacija v intervalu, ki je nastavljen s strani uporabnika, za vsako oddaljeno mesto prenese seznam datotek. Pridobljen seznam primerja z lokalno shranjenim seznamom, kjer so navedene že vse prenesene vsebine ter njihov čas zadnje spremembe. S pomočjo lokalnega seznama izloči že vse prenesene vsebine ter seznam datotek na oddaljenem mestu po petih sekundah ponovno osveži.

Med prvim ter drugim osveženim seznamom aplikacija primerja velikost ne prenesenih datotek. Če se velikost datoteke v tem času ni spremenila, se smatra, da je datoteka na oddaljenem mestu shranjena v celoti, ter jo prenese na lokalni datotečni sistem.

Primarna instanca aplikacije prenesene vsebine z lokalnega datotečnega sistema po prenosu uvrsti v čakalno vrsto. Datoteke so v čakalni vrsti razvrščene naraščajoče glede na časovno oznako datoteke. Tako se zagotovi obdelava vsebin v enakem zaporedju, kot so bile prenesene.

4.4.2 Vklapljanje dodatnih računalnikov preko omrežja

Primarna instanca aplikacije na 30 minut preveri število vsebin v čakalni vrsti in ga primerja s pragom, ki je nastavljen na posameznem računalniku. V primeru, da je število vsebin v vrsti večje od praga, aplikacija sestavi ter pošlje WOL (ang. *wake on lan*) paketek na naslov za razpršeno oddajanje (angl. *broadcast address*) za vsak posamezen računalnik.

WOL paketek je običajen UDP, ki vsebuje MAC naslov ciljnega računalnika in ga pošljemo na naslov za razpršeno oddajanje na vrata 9.

Prvih 6 bajtov paketka sestavlja šestnajstiška vrednost 0xff, vsakih naslednjih 6 bajtov paketka do zapolnitve paketka, zavzema MAC naslov računalnika. Velikost paketka je določena kot 16x ponovitev 6 bajtnega zapisa MAC naslova z dodatnimi 6 bajti za zaglavje paketka. Ker dostava UDP paketkov preko omrežja ni zagotovljena, je paketek smotrno poslati večkrat.

Paketek v javi lahko sestavimo z dokaj enostavno metodo:

```
public void wol(String MAC) {
        int PORT = 9;
        byte[] macBytes = getMacBytes(MAC);
        byte[] bytes = new byte[6 + 16 * macBytes.length];
        for (int i = 0; i < 6; i++) {
             bytes[i] = (byte) 0xff;
        }
        for (int i = 6; i < bytes.length; i += macBytes.length) {
           System.arraycopy(macBytes, 0, bytes, i, macBytes.length);
        }
        InetAddress address = InetAddress.getByName("192.168.31.255");
        DatagramPacket packet = new DatagramPacket(bytes, bytes.length,
address, PORT);
          DatagramSocket socket = new DatagramSocket();
          socket.send(packet);
          socket.close();
        }
```

Mrežna kartica ciljnega računalnika ob prejetju paketka računalnik vklopi.

4.4.3 Prenos nastavitev

Aplikacija ob zagonu naloži lokalne nastavitve, kjer dobi podatke o svojem imenu ter IP naslov primarne instance. Po uspešnem zagonu, ter pred vsako zahtevo za novo vsebino aplikacija preveri ali so na voljo novejše nastavitve s spodnjim HTTP zahtevkom.

```
http://192.168.31.10:8080/Content?action=config&node=Farm-3&serial=43
```

Primarna instanca preveri, ali se serijska številka nastavitev ujema s trenutno veljavnimi nastavitvami. V primeru, da so na voljo novejše nastavitve, jih vrne v obliki XML datoteke. Datoteke z nastavitvami se generirajo glede na vlogo računalnika.

Tako ima primarna instanca v konfiguracijski datoteki dodatne nastavitve, ki pri ostalih niso prisotne.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE CONFIG SYSTEM "config.dtd">
<CONFIG>
  <NODE Name="Farm-1"
        Serial="44"
        Primary="192.168.31.10:8080"
        Role="Primary"/>

<Location Name="test-ponudnik-1"
      Path="ftp://test:testko@192.168.31.100/Vsebine"
Dest="test-ponudnik-2" Int="15" Profile="1,2" Type="FTP"/>

  <Location Name="test-ponudnik-2" Path="/local-ponudnik-2"
Dest="test-ponudnik-2" Int="10" Profile="2" Type="Local"/>

</CONFIG>
```

Polna oblika datoteke z nastavitvami pri ostalih instancah:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE CONFIG SYSTEM "config.dtd">
<CONFIG>
  <NODE Name="Farm-3"
        Serial="44"
        Primary="192.168.31.10:8080"
        Role="Secondary"/>

  <Profile Name="MPEG2"
           Binary="/home/test/bin/ffmpeg"
           Args="-i <input> -vcodec mpeg2video -b:v 2800k -minrate
500k -maxrate 6500k -bufsize 1835k -g 25 -an -threads 7 -an -f vob
-y <output>"/>

  <Profile Name="h264"
           Binary="/home/test/bin/HandBrakeCLI"
           Args="-i <input> -o <output> -f mkv --crop 0:0:0:0 --
strict-anamorphic -e x264 -b 2500 -x264-preset faster -a none -x
vbv-maxrate=2500:vbv-bufsize=1000:keyint=25"/>

  <Storage Name="storage-1" Path="192.168.31.101:/video/"/>
  <Storage Name="storage-2" Path="192.168.31.102:/video/"/>

  <SMTP Host="smtp.test.lan"
        User="test"
        Pass="testis"
        Addr="transcoder@test.lan"/>

  <Mail Addr="test@neobstaja.com"/>
  <Mail Addr="kekec@mojca.com"/>
  <Mail Addr="model@kadogaja.com"/>

</CONFIG>
```

4.4.4 Obdelava vsebine

Aplikacija po zagonski inicializaciji preko protokola HTTP pri primarni instanci zahteva novo vsebino s klicem:

http://192.168.31.10:8080/Content?action=getnew&node=Farm-3

Odgovor prejme v obliki xml dokumenta:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE CONTENT SYSTEM "content.dtd">
<CONTENT>
        <DATA Location="/test-ponudnik-1"
            Filename="Oddaja1.tar"
            Profiles="MPEG2,h264"/>
</CONTENT>
```

Podano datoteko aplikacija s skupne lokalne podatkovne shrambe nato prenese na svoj lokalni datotečni sistem ter arhiv ekstrahira. Veljavna arhivska datoteka mora vsebovati XML ali besedilno datoteko z metapodatki o vsebini, naslovno sliko ter datoteko z video vsebino.

Iz besedilne ali XML datoteke aplikacija razbere ime ponudnika, ime oddaje, kratek opis oddaje, časovni interval, med katerim mora biti vsebina na voljo uporabnikom, md5 zgoščeno vrednost za preverjanje integritete, ter ime datoteke video vsebine.

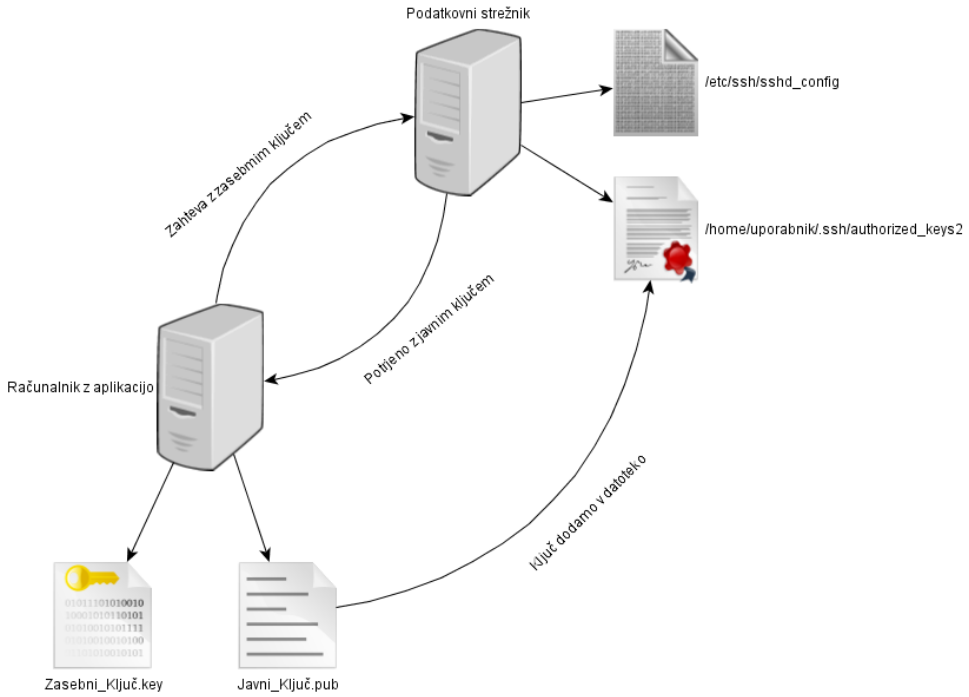
```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE ADI SYSTEM 'ADI.DTD'>
<ADI>
  <Asset>
     <Metadata>
        <AMS Provider="test-provider-1"
             Product="Oddaja1;"
             Asset_Name="610234897"
             Description="Testna oddaja stevilka 1"
             Creation_Date="2012-12-04"
             Asset_ID="SHOW0001351512288005"
             Verb="" />
  <App_Data App="VOD" Name="Type" Value="movie"/>
  <App_Data App="VOD" Name="Title" Value="Testna Oddaja - test"/>
  <App_Data App="VOD" Name="Display_Run_Time" Value="00:04"/>
  <App_Data App="VOD" Name="Run_Time" Value="00:04:07"/>
  <App_Data App="VOD" Name="Year" Value="2012"/>
  <App_Data App="VOD" Name="Category" Value="5"/>
  <App_Data App="VOD" Name="Genre" Value="Oddaja"/>
  <App_Data App="VOD" Name="Provider" Value="test-provider-1"/>
  <App_Data App="VOD" Name="Start_Time" Value="2012-10-26 21:00"/>
  <App_Data App="VOD" Name="End_Time" Value="2013-12-31 22:59"/>
  <App_Data App="VOD" Name="Audio_Type" Value="Stereo"/>
  <App_Data App="VOD" Name="Content_CheckSum"
Value="e81d60052473e42735bb12e0089258c2"/>
      </Metadata>
        <Content Value="Oddaja1.mpg"/>
      </Asset>
</ADI>
```

V primeru, da časovni interval še ni pretekel, aplikacija video datoteko obdela s profili določenimi v odgovoru primarne instance. Po končani obdelavi aplikacija izvirno vsebino shrani kot varnostno kopijo ter obdelane vsebine prenese na datotečni strežnik s pomočjo SCP protokola.

Za prenos vsebin na datotečni strežnik aplikacija uporabi avtentikacijo s pomočjo asimetričnih ključev (slika 54). Zasebni ključ, s katerim se aplikacija predstavi strežniku za hrambo podatkov, je varno zapisan na posameznem računalniku. Datotečni strežnik ključ preveri v seznamu dovoljenih ključev v datoteki */root/.ssh/authorized_keys2* z vsebino:

```
from="192.168.31.10" ssh-rsa AAAAB3NzaC1yc2EAAAAB..+VuPPXtH1gO
C/GLDyHP9+ QeEUcFK7QIFVb1fQ/GGojK3GKlmMkA/NowNk+fgLKQw== Farm-3
```

S pomočjo ključev enostavno odpravimo potrebo po uporabniškem geslu, omogočimo avtomatizirane prenose ter omejimo dostop do strežnika na samo določene računalnike.



Slika 54: Diagram SSH avtorizacije z javnim in zasebnim ključem.

Po končanem prenosu aplikacija obvesti določene naslovnike po elektronski pošti in primarni instanci sporoči, da je obdelava vsebine končana preko protokola HTTP.

http://192.168.31.10:8080/Content?action=complete&node=Farm-3&location=/test-ponudnik-1&filename=Oddaja1.tar&storage=storage-1

4.5 Podatkovna baza

Podatkovna baza je sestavljena iz sedmih tabel (slika 55). Tabela *MailAddr* vsebuje enolično identifikacijsko številko, e-poštni naslov za obveščanje ter logično vrednost ali je naslov omogočen.

Tabela *MailServer* vsebuje nastavitve poštnega strežnika, enolično identifikacijsko številko, logično vrednost ali je strežnik omogočen, polno domensko ime, uporabniško ime, geslo ter e-poštni naslov za odhodno pošto.

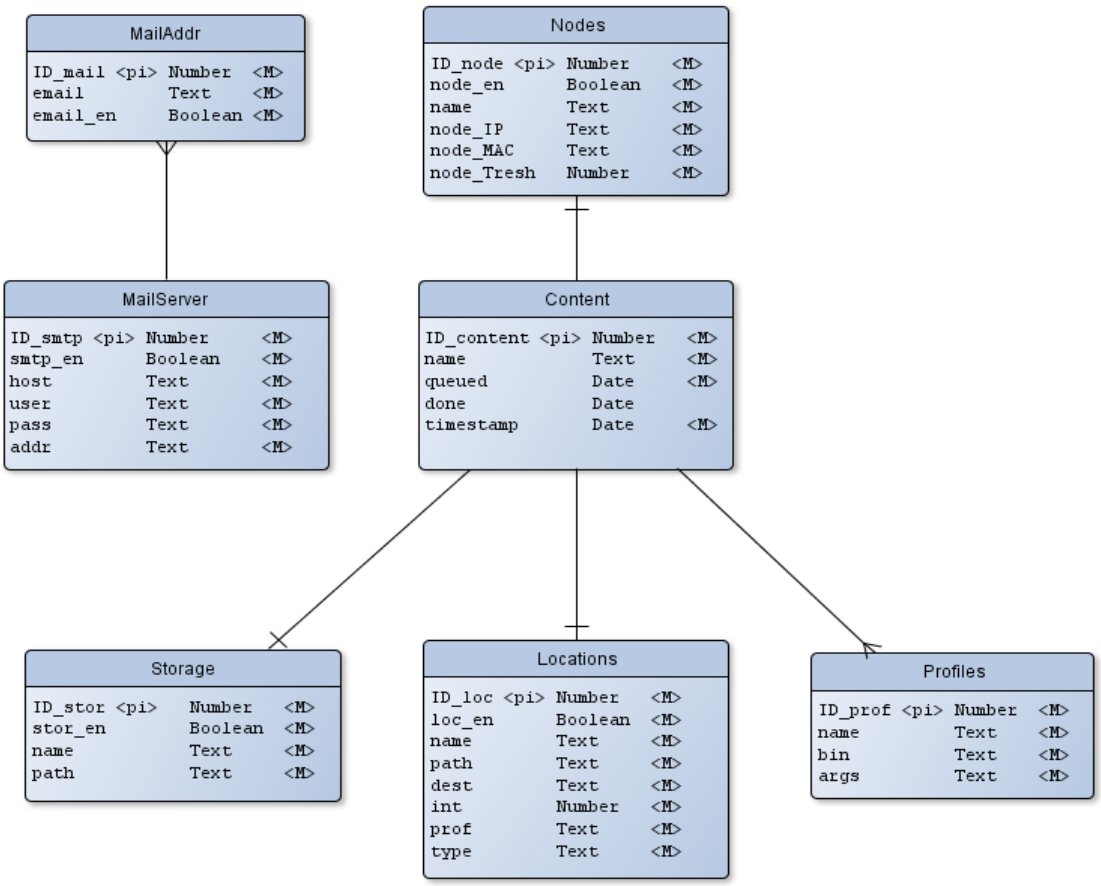
Tabela *Storage* hrani podatke o datotečnih strežnikih, enolično identifikacijsko številko, logično vrednost ali je strežnik omogočen, ime ter pot.

Tabela *Profiles* ima zapisano enolično identifikacijsko številko profila, ime, pot do izvršilne datoteke ter argumente.

Tabela *Locations* za vsako oddaljeno lokacijo hrani enolično identifikacijsko številko, logično vrednost ali je omogočena, ime, polno pot, ciljno pot, časovni interval, številke profilov ter tip oddaljene lokacije.

Tabela *Nodes* hrani identifikacijsko številko, logično vrednost ali je računalnik omogočen, njegov IP ter MAC naslov in prag, nad katerim se računalnik vklopi.

Tabela *Content* za vsako vsebino hrani identifikacijsko številko, ime, datum in uro, ko je bila uvrščena v čakalno vrsto, datum in uro, ko je bila obdelava končana ter prvotno časovno oznako datoteke.



Slika 55: Struktura podatkovne baze.

5 Zaključek

V diplomski nalogi je predstavljena aplikacija za administracijo obdelave video vsebin v obliko primerno oddajanju preko IPTV omrežji. Razvita je bila funkcionalnost razporejanja dela na več računalnikov s skupnim spletnim vmesnikom za nastavitve in mehanizem za distribucijo le-teh. Razvita aplikacija je zaradi zasnove in implementacije popolnoma prenosljiva med različnimi operacijskimi sistemi z izjemo binarnih datotek kodirnikov, katere bi bilo potrebno ponovno prevajati za posamezne sisteme.

Ker je aplikacija v konstantnem razvoju, so možne še številne izboljšave:

- Prenosi novih vsebin iz oddaljenih mest se v trenutni zasnovi lahko izvajajo samo na računalniku kjer teče primarna instanca aplikacije. Nastavitve oddaljenih mest bi lahko nadgradil z zmožnostjo določanja računalnika, na katerem naj se prenosi izvajajo.
- Aplikacija za prvi zagon potrebuje IP naslov računalnika, kjer teče primarna instanca. Z implementacijo protokola za odkrivanje preko omrežja s pomočjo *multicast* oddajanja, bi to pomanjkljivost lahko odstranili.
- Podatkovna baza se hrani le na primarnem računalniku in se ne usklajuje z ostalimi računalniki v sistemu. Prihodnje izboljšave bi lahko prinesle sinhrone in odložene posodobitve podatkovne baze med računalniki.
- V primeru izpada primarnega računalnika bi se obdelovanje vsebin ustavilo. Z implementacijo plavajočih vlog po vzoru *enak z enakim* ali P2P omrežji, bi s kombinacijo prejšnje izboljšave dosegli popolno odpornost sistema v primeru odpovedi.
- Aplikacijo bi lahko nadgradili s sposobnostjo odkrivanja napak v video posnetkih, kot so črni zasloni, prekomerno popačenje slike in krajša dolžina posnetka napram izvirniku. Izboljšavo bi lahko implementirali z SSIM metriko, ki smo jo uporabili za preverjanje podobnosti končnega zapisa z izvirnikom.
- Trenutna implementacija spletnega vmesnika za administracijo od uporabnika ne zahteva uporabniškega imena in gesla , kar bi lahko v večjih in bolj odprtih okoljih predstavljalo veliko težavo. Z implementacijo osnovnega varnostnega mehanizma bi to pomanjkljivost odpravili.

6 Literatura

[1] (2012) "History of Java". Dostopno na:
http://ei.cs.vt.edu/~wwwbtb/book/chap1/java_hist.html

[2] (2012) "The Eclipse foundation". Dostopno na:
http://wiki.eclipse.org/Main_Page

[3] (2012) "TortoiseSVN subversion control". Dostopno na:
http://tortoisesvn.net/about.html

[4] (2012) "FFmpeg documentation". Dostopno na:
http://ffmpeg.org/documentation.html

[5] (2012) "HandBrake open source video transcoder". Dostopno na:
http://handbrake.fr/details.php

[6] (2012) "MPEG-2 Transport vs. Program Stream". Dostopno na:
www.vbrick.com/docs/VB_WhitePaper_TransportStreamVSProgramStream_rd2.pdf

[7] (2012) "Apache Commons Net™ library." Dostopno na:
http://commons.apache.org/net/

[8] (2012) "Ganymed SSH-2 for Java." Dostopno na:
http://www.cleondris.ch/opensource/ssh2/

[9] (2012) "SQLite documents" Dostopno na:
http://www.sqlite.org/docs.html

[10] (2012) "Native cross-platform library to extract archive formats." Dostopno na:
http://sourceforge.net/projects/sevenzipjbind/

[11] (2012) "Samba - Opening Windows to a Wider World " Dostopno na:
http://www.samba.org/

[12] (2012) "PSNR peak-to-peak signal-to-noise ratio metric." Dostopno na:
http://www.compression.ru/video/quality_measure/info_en.html#psnr

[13] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity", *IEEE transactions on image processing*, št. 4, zv. 13, str. 600-610, 2004.

[14] (2012) "Bitrate Viewer" Dostopno na:
http://www.winhoros.de/docs/bitrate-viewer/

[15] (2012) "MSU Video Quality Measurement Tool" Dostopno na: http://compression.ru/video/quality_measure/video_measurement_tool_en.html

[16] (2012) x264 preset reference. Dostopno na: http://dev.gentoo.org/~beandog/x264_preset_reference.html