TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Karl Õmblus

Kõrgkäideldav võrgukoormuse jaotamisega reaalaja videoülekande lahendus

Bakalaureusetöö (9 EAP)

Juhendaja: Märt Bakhoff, MSc

Tartu 2020

Kõrgkäideldav võrgukoormuse jaotamisega reaalaja videoülekande lahendus

Lühikokkuvõte:

Maapiirkondades või väliüritustel puudub tihti stabiilne internetiühendus, mis muudab sealt videoülekannete tegemise problemaatiliseks. Bakalaureusetöö eesmärk on luua kõrgkäideldav, võrgukoormuse jaotamisega videoülekande edastusplatvorm, mille abil on võimalik töökindlalt edastada reaalaja videovoogu tingimustes, kus on olemas mitu internetiühendust, mis eraldivõetuna võivad olla ebastabiilsed. Loodud rakenduses liidetakse erinevad internetiühendused kokku ühtseks stabiilseks üleslaadimiskanaliks. Videovoog dubleeritakse mitmete serverite vahel vajaduseta sama voogu mitmekordselt üles laadida. Töökindluse kontrollimiseks viidi läbi vigade imiteerimisel põhinevad testid, mille kohaselt osutus rakenduse töökindlus väga heaks.

Võtmesõnad: Video voogedastus, kõrgkäideldavus, võrgukanalite liitmine.

CERCS: P175 (Informaatika, süsteemiteooria)

High Availability Realtime Video Delivery System With Uplink Bonding

Abstract:

Internet connection in rural areas and at outdoor events often suffers from stability issues, which makes online video broadcasting from such locations problematic. The purpose of this Bachelor's thesis is to create a high availability video streaming software solution capable of delivering a stable real time video stream in conditions where there is more than one internet connection available, each of which individually can be unstable. The created solution is based on dividing the network load by combining several distinct internet connections into one upload stream regardless of their individual speeds and latencies. The redundancy of the video stream is achieved by means of data duplication between servers without the need for duplication of uploads. Testing confirmed the solution to have excellent fault tolerence.

Keywords: Video streaming, high availability, network bonding

CERCS: P175 (Informatics, systems theory)

Sisukord

1 Sissejuhatus

Video sujuva voogedastuse eelduseks on kiire ja stabiilne internetiühendus, mille lühikenegi katkestus põhjustab nähtavaid häireid videopildis. Praktikas ei vasta internetiühendus aga kaugeltki igal pool neile kriteeriumitele, mis teeb kvaliteetse teleülekande tagamise eelkõige maapiirkondades toimuvatest sündmustest sageli problemaatiliseks.

Lisaks visuaalesteetilise kvaliteedi langusele takistab ebastabiilne või kõikuva kiirusega internetiühendus oluliselt aegkriitiliste, reaalajas toimuvate ülekannete (nt loosimised, spordivõistlused, *vox populi*-formaadis saated ja ajaloolise väärtusega sündmused) jälgimist, hindamist ning tootena esitlemist. Samuti võib ootamatu rike videovoogu laialijagavas serveris ülekande peatada.

Tavaolukorras, juhul kui kasutatav internetiühendus võimaldab ülekande paigas edastatavast videovoost kordades suuremat üleslaadimiskiirust, saab laadida videopilti samaaegselt üles mitmesse serverisse korraga. See hajutab riski, et ühe serveri rikke korral voogülekanne katkeb.

Juhul kui üleslaadimiskiirus mitmekordset samaaegset üleslaadimist ei võimalda või pole tegu tagatud ühenduskiirusega (mobiilne internet), siis ei saa kirjeldatud meetodit kasutada. Muutuva kiirusega ühenduse korral võib tekkida olukord, kus videoülekanne on küll võimalik, kuid dubleeritud laadimine ummistab kanali ning mõlemad videovood muutuvad katkendlikuks.

Töö eesmärk on luua videoülekande lahendus, mis tagab katkestusteta videovoo edastamise ebastabiilse või kõikuva kiirusega internetiühenduse tingimustes, kasutades mitme internetiühenduse liitmist ühtseks kanaliks ja samas dubleerib selle ka serverite vahel.

Töö teine peatükk käsitleb edastava videovoo loomiseks kasutatavaid tehnoloogiaid. Kolmandas peatükis kirjeldatakse rakenduse loomist, selle töötamise loogikat ning arhitektuuri. Rakenduse töökindluse tagamiseks on oluline seda väljatöötamise käigus testida, imiteerides erinevaid esineda võivaid riske ja rikkeid. Rakenduse töökindluse testimist kirjeldab töö neljas peatükk.

2 Seotud tehnoloogiad

Siin peatükis selgitatakse, millistest komponentidest koosneb videovoog ning mis viisil see luuakse. Käsitletavad tehnoloogiad on järjestatud lähtuvalt videovoo ülesehituse kihtidest seestpoolt väljapoole. Lisaks on toodud olemasolevad alternatiivid.

2.1 AAC audiokoodek

Advanced Audio Coding (AAC) on digitaalse heli kadudega pakkimise standard, mis jätab inimkõrvale kuuldamatu või üldisest helinivoost väheeristuva spektriosa esitamata. Võrreldes mp3-ga vajab AAC pakkimise formaat sama kvaliteediga monoheli edastamiseks 70% ribalaiusest ning eeldab stereoheli edastamisel kiirust 128 kbit/s [1]. Helifaili AAC formaadis esitamine ei ole koodekipõhiselt tasuline [2].

2.2 H.264 videokoodek

H.264 on 2003. aastal loodud video kadudega pakkimise standard, mis on viimase kolme aasta jooksul osutunud enimkasutatavaks [3].

Video kompresseerimine võimaldab edastada sisu tingimustes, kus edastatav ribalaius on piiratud. Kompresseerimata kujul 24-bitilise värvisügavusega 1920×1080 pikslimõõduga HD video salvestamiseks või edastamiseks vajalik ribalaius Euroopas levinud kaadrisageduse juures on järgmine: 24 × 1920×1080 x 25 = 1.24 Gbit/s.

Kadudega kompresseerimise korral toimub ribalaiuse valimine pildi kvaliteedi ning kasutada oleva ribalaiuse vahelise kompromissina.

H.264 laialdane levik võimaldab kodeerida video ühekordselt erinevate operatsioonisteemidega arvutite ning eri tootjate mobiilseadmete tarbeks, ilma et seda oleks vaja serveris omakorda erinevate koodekitega ümber transleerida. Fakt, et tänapäeval suudab enamik seadmeid H.264 videosignaali dekodeerida, on taganud kõnealuse toote laia kasutatavause ka hilisemate ja efektiivsemate pakkimisalgoritmide kõrval [4].

Videoülekande puhul tagab märkimisväärse andmemahu kokkuhoiu koodeki omadus defineerida kaadri muutumatud osised mitteedastavateks. Selle tulemusena lisatakse videovoogu esmalt kogu pildi infot kandev võtmekaader (ingl *I frame*), millele lisatakse kaadri muudatuste ulatust defineerivad P (ingl *predicted*) ja B (ingl *bidirectionally*

predicted) kaadrid [5]. Sellist kaadrite jada nimetatakse pildigrupiks (GOP, ingl *group of pictures*), mille dekodeerimise järel ilmneb uuesti täismahus võtmekaader [6].

Võtmekaadrite arvulise suuruse vähendamine võimaldab säästa andmemahtu, ent säilitab kodeerimisvigade realiseerumise korral visuaalse defekti (enamasti udused laigud) järgmise võtmekaadrini.

2.3 MPEG-2 transportvoog

MPEG-2 transportvoog (ingl *MPEG transport stream*) on digitaalse konteineri standard, mis võimaldab mitu edastusvoogu üheks vooks konsolideerida. Multipleksitavatel voogudel võib olla ka erinev ajabaas [7].

Sellisel viisil liidetakse ühtseks andmevooks H.264 video ja AAC heli, mis omakorda võimaldab edastada ühes infovoos mitut kaamerarakurssi ja/või mitmesisulist (sh mitmekeelset) heliriba.

2.4 Voogedastus

Voogedastus (ingl *streaming*) on meetod edastada ja presenteerida multimeedia sisu meediafaili tervikut esitusseadmesse laadimata. Meediaedastus muutub võimalikuks hetkest, mil arvuti mällu on laetud dekodeerimiseks piisav hulk andmeid.

Reaalaja voogedastuse all mõistetakse audio või video otseülekannet.

2.5 HTTP reaalaja voogedastus

HTTP reaalaja voogedastus (ingl *HTTP Live Streaming, HLS*) on Apple loodud voogedastuse protokoll [8]. HLS formaat võimaldab voogedastada nii staatilist kui dünaamilist reaalaja meediat.

HLS voogedastuse protokoll tükeldab MPEG-2 kapseldatud meedia väikesteks (enamasti vältusega 5-10 sekundit) failideks laiendiga .ts (MPEG-2 transpordivoog), edaspidi viidatud kui segment. Kirjeldatud protsessi käigus loob programm taasesitusloendi (ingl *playlist*) faililaiendiga .m3u8, milles staatilise sisu korral loetletakse kõik segmenteeritud komponendid. Sealjuures ei tohi pildigrupp (GOP) olla pikem segmendi pikkusest [9].

Reaalajas toimuva voogedastuse tingimustes loob programm esitusloendi 5-10 viimati edastatud segmendifailiga [10].

Sama videovoogu on võimalik kodeerida mitut eri kvaliteeti tagava seadistustega. Sellisel juhul luuakse veel üks esitusloend, mis viitab erineva kvaliteediga esitusi sisaldavatele loenditele [11]. Samal viisil võib kõrgkäideldavuse tagamiseks viidata ka mitmele sama kvaliteediastmega esitlusloenditele, mis asuvad erinevates serverites [12].

Failide edastamine vaatajale toimub standardset veebiserverit ja HTTP protokolli kasutades, mis võimaldab voogedastust läbi enamike tulemüüride ning puhverserverite.

Taasesitamisel laetakse esmalt .m3u8 esitlusloend, mille põhjal laetakse järjest alla .ts faile, mis üksteise järele liidetuna loovad kasutajale ühtse videovoo.

HLS esitlusloend näeb välja järgmine:

```
#EXTM3U
#EXT–X–TARGETDURATION:10
#EXT–X–VERSION:4
#EXT–X–MEDIA–SEQUENCE:1
#EXTINF:10.0,
fileSequence1.ts
#EXTINF:10.0,
fileSequence2.ts
#EXTINF:10.0,
fileSequence3.ts
```

HLS editlusloendi ülesehitus on järgmine:

- EXTM3U: markeerib, et tegemist on hls esitlusloendiga.
- EXT-X-TARGETDURATION: tüki maksimaalne pikkus.
- EXT-X-VERSION: esitlusloendi failiformaadi versiooninumber. See kirjeldab, millised lisavõimalused võivad esitlusloendi failis kasutusel olla [13].
- EXT-X-MEDIA-SEQUENCE: esitlusloendis oleva esimese segmendifaili järjekorranumber.
- EXTINF: meediafaili lisainfo, mis peab sisaldama vähemalt täpset kestvust.
- Failinimi, millele eelmise rea info viitab, vajadusel koos suhtelise kataloogiteega selleni.

Kahte viimast rida korratakse vastavalt sellele, mitut viimast segmenti soovitakse esitlusloendis näidata.

HLS voogedastuse korral luuakse iga segmendi laadimiseks uus ühendus, seega on see edastus vastupidav lühiajalistele võrgukatkestustele, ilma et kasutaja näeks edastatavas videos katkestust. HLS toetab ka video krüpteerimist [14], kuid seda selles töös ei käsitleta. HLS peamisteks eelisteks võib lugeda laialdase toe erinevatel seadmetel ning edastuskindluse kvaliteedi; puudusteks on suur latentsus (viivitus võib olla kuni 30 sekundit) [15].

2.6 FFmpeg raamistik

FFmpeg on juhtiv vabavaraline multimeedia raamistik, mis võimaldab kodeerida peaaegu kõiki meediaformaate, antiiksetest kaasaegseteni [16]. FFmpeg projekt pakub tehniliselt parimat võimalikku lahendust rakenduste arendajatele ja lõppkasutajatele. FFmpeg raamistiku üks osa on ilma graafilise kasutajaliideseta käsurea tööriist ffmpeg [17]. Ffmpeg utiliit võimaldab ühe käsuga lugeda arvuti heli- ja videosisendit, kompresseerida need vastavalt AAC ja H.264 formaati. Seejärel kapseldatakse need MPEG-2 transportvooks ning salvestatakse tulemuse HLS voogedastuse protokollile vastavaks esitlusloendiks ja segmendifailideks.

2.7 Alternatiivsed videoülekande programmid

Tavapärased videoedastustarkvarad, näiteks Wirecast [18] ja Restream [19], võimaldavad edastada videovoogu samaaegselt mitmesse erinevasse serverisse. Lisavõimalusena on võimalik kasutada erinevates serverites ka erinevaid kvaliteeditasemeid või edasutsprotokolle. Wirecast on küll tasuline, kuid lisaks videoedastusele on tegu täisfunktsionaalse tarkvaralise videostuudioga. Sellist mitmekordset üleslaadimist soovitab ka voogedastusplatvorm Akamai [20].

Mitme internetiühendusega voogedastuse võimalust pakub Teradek BOND [21]. Lahendus hajutab riski internetiühenduse probleemide korral, kuid üleslaadimine toimub ainult ühte serverisse, mille rikke korrel katkeb kogu videoedastus [22]. Infot üleslaadimisel kasutatavate tehnoloogiate kohta ei ole avaldatud, lahendust on võimalik soetada ainult riistvaralisena.

Teine samalaadne mitut ühendust paralleelselt kasutav lahendus on Mushroom Streamer PRO [23]. Tööpõhimõte on sarnane Teratek BOND tööpõhimõttele koos sama puudusega: jaotatud videovoog kombineeritakse kokku dubleerimata seadmes [24]. Sarnaselt Teradekile ei avaldata seadme töötamise tagamaid, lahendust on võimalik soetada ainult riistvaralisena.

Mõlemaid internetiühendusi liitvaid lahendusi saab kasutada kõrgkäieldavuse loomiseks dubleeritult, kuid sel juhul kahekordistuvad ka nõuded kasutatava internetiühenduse ribalaiustele. Kasutades 4G USB modemeid, kahekordistub ka vajalike ühenduste arv, suurendades kulusid veelgi.

Lõputöös loodava voogedastusplatvormi eesmärk on ühendada mõlemad lahendused: video üleslaadimine üle mitme internetiühenduse ja kõrgkäideldavus, jagades voog mitme serveri vahel. Erinevalt Akamai soovitatud lahendusest toimub dubleerimine serverite poolel. Seepärast tuleb videovoog laadida üles ainult ühekordselt, vähendades sellega vajalikku ühenduskiirust mitmekordselt.

3 Loodava rakenduse arhitektuur

Rakenduse eesmärk on luua veakindel voogedastusplatvorm, mille abil on võimalik töökindlalt edastada reaalaja videovoogu, kasutades mitut internetiühendust. Lisaks luuakse kõrgkäideldav serverite võrgustik ning koormusjaotusega veakindel väljundsüsteem videovoo edastamiseks suurele vaatajahulgale.

Lõputöös on valitud edastusprotokolliks HLS, sest see on universaalne, laialt toetatud ning võimaldab videovoo edastamist tükkidena. Tükkidena edastus tähendab, et iga segmenti saab kliendile edastamise seisukohast käsitleda kui iseseisvat faili, mis ei ole teistega seotud. See annab suure eelise RTMP protokolli ees, mille kasutamisel ka lühiajalise internetiühenduse katkestuse korral katkeb kogu voog [25].

Nõuded loodavale tarkvarale:

- video edastatakse minimaalse viivitusega ülekande paigast suurele hulgale vaatajatele;
- video üleslaadimisel võttepaigast serveritesse kasutatakse ühenduse ribalaiust minimaalselt, sh ei toimu sama video topelt üleslaadimist;
- tööd jätkatakse katestusteta ka üksiku internetiühenduse katkemisel;
- tööd jätkatakse ka siis, kui internetiühenduste kiirused on erinevad ja ebastabiilsed;
- olukorras, kus video kodeerimistarkvara lõpetab vea tõttu töö, jätkatakse tööd kiiresti;
- tööd jätkatakse katkestusteta ka sisend- või keskserveri rikke korral;
- tööd jätkatakse katkestuseta või minimaalse katkestusega ka väljundserveri rikke korral;

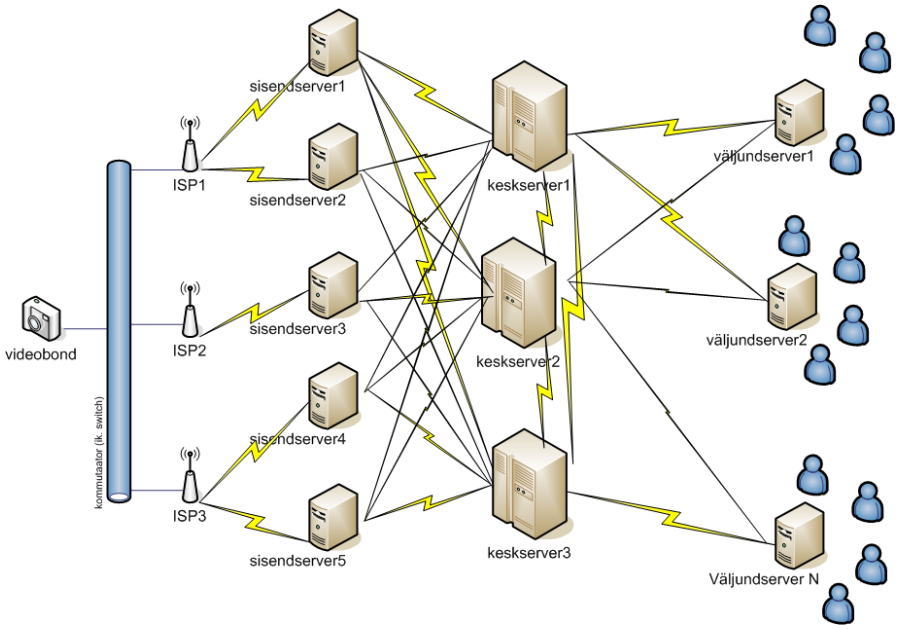
Loodava tarkvara töökindlus tagatakse järgmiselt:

- ülekandepaigas mitme internetiühenduse korraga kasutamine;
- üleslaetavate andmete jagamine eri ühenduste vahel vastavalt ühenduse kiirusele;
- videovoogu vastuvõtvate sisendserverite, talletavate keskserverite ja vaatajatele edastavate väljundserverite liiasus;

- kvoorumi kasutamine andmete kohale jõudmise tagamiseks;
- automaatne keskserveris puuduolevate pakettide või segmentide taastamine teistest keskserveritest;

Süsteem koosneb viiest eraldiseisvast komponendist:

- videobond - edastab video üle mitme internetiühenduse sisendserveritele;
- sisendserver - võtab videobondilt tulnud paketid, dubleerib need keskserveritele;
- keskserver - liidab paketid HLS segmentideks, loob taasesitusloendi;
- väljundserver - jagab videovoo edasi vaatajatele, toimib puhverserverina;
- veebiserver - videoülekande avalik liides vaataja jaoks;



Joonis 1. Videoülekandelahenduse tööpõhimõtte plokkskeem.

Loodava lahenduse üldist tööpõhimõtet illustreeriv plokkskeem on toodud joonisel 1.

Kõrgkäideldavuse tagamiseks loetakse iga segmendi osa, pakett, õnnestunult üleslaetuks juhul, kui selle kättesaamist ning salvestamist on kinnitanud rohkem kui pooled

keskserveritest. Seda vajalikku künnist nimetatakse kvoorumiks. Kvoorum leitakse valemiga

$$kvoorum = \lfloor \frac{keskserverite\_arv}{2} \rfloor + 1$$

Kõrgkäideldavuse tagamiseks peab olema süsteemis vähemalt kolm serverit.

Kvoorumi kasutamisega on tagatud, et keskserveri rikke korral on videovoo loomiseks vajalikud paketid kättesaadavad.

Järgmistes alampeatükkides on täpsemalt kirjeldatud videoedastusplatvormi komponentide töö.

3.1 Videobond

Rakenduse peamine eesmärk on lugeda sisse ffmpeg abil loodud HLS reaalaja videovoog ning edastada see segmendifailidena sisendserveritele.

Rakenduse loomiseks on kasutatud programmeerimiskeelt C, sest sel viisil on võimalik luua kiire ning vähese ressursinõudlikkusega rakendus. Ressursisäästlikkus video edastamisel on oluline, sest samas arvutis toimub samal ajal video reaalajas kodeerimine, mis on samuti ressursinõudlik. Samal põhjusel puudub rakendusel ka graafiline kasutajaliides, mille puudumise eeliseks on ka see, et kui kodeerivas arvutis kasutatakse Linux operatsioonisüsteemi, siis saab ressursikulu veelgi vähendada, kui jätta aknahaldur käivitamata.

Videobondi kogu suhtlus võrguga toimub ainult läbi sisendserverite, mis omakorda vahendab päringu kõigile keskserveritele. Töö alustamisel logib videobond keskserveritesse sisse kasutades edastava videovoo nime ja parooli. Sisselogimine toimub läbi vabalt valitud sisendserveri. Järgmisena küsitakse alustatava videovoo viimase üles laetud segmendi järjekorranumber. Kasutusele võetakse kõigi keskserverite vastustest suurim, sest nii ei kirjutata juba salvestatud voogu üle ning nummeratsioonis mahajäänud serverid jätkavad uue järjekorranumbriga.

Iga ffmpeg abil valminud HLS segment jaotatakse omakordse väiksemateks pakettideks, mis lisatakse üleslaadimisjärjekorda. Videobondist saabunud paketid pannakse keskserveris uuesti segmendifailideks kokku ja koostatakse HLS esitlusloend. Üleslaadimisjärjekorda lisamisel kasutatakse serveri numeratsiooni, mis on ffmpeg abil loodud esitlusloendi numeratsioonist sõltumatu.

Videobond peab arvet numeratsioonide omavahelise seose üle ning protsessi taakäivitamisel jätkatakse tööd võimaluse korral poolelijäänud segmendifaili juurest. Sel viisil ei laeta sama segmendifaili mitu korda erineva järjekorranumbriga üles.

Iga sisendserveri jaoks luuakse eraldi lõim (ingl *thread*), mille eesmärk on antud serverisse üleslaadimisjärjekorrast pakette saata. Eri serverite üleslaadimised toimuvad teineteisest sõltumatult. Töötsükli alguses võtab lõim üleslaadimisjärjekorrast vanima paketi, mida hakkab üles laadima.

Kuni ühenduses ei toimu viga ega viiteid, laetakse iga pakett üles vaid ühe korra ühte sisendserverisse. Paketi nominaalne üleslaadimisaeg leitakse valemiga

$$nominaalaeg = segmendi\_pikkus * lõimede\_arv/pakette\_segmendi\_kohta$$

mis kehtib eeldusel, et pakettide arv >= lõimede arvuga.

Normaalseks tööks peab keskmine üleslaadimisaeg olema eelpool toodud valemi abil leitud nominaalajast lühem, vastasel juhul jääb ajaühikus üleslaetavate videokaadrite arv väiksemaks kui reaalajalise esitluse jaoks tarvis on. Vaataja jaoks tähendaks see katkestust ning video ajas maha jäämist kuni üleslaasimispuhvri täieliku täitumiseni, mis omakorda lõpetaks programmi töö.

Üleslaadimise kiirendamiseks olukorras, kus vaid üks lõim ei jõua seda teha nominaalse ajaga, lisatakse sama pakett tagasi üldjärjekorda. Kuna ülekande ajaline viide on HLS tööpõhimõtte tõttu mitu korda suurem kui ühe segmendi pikkus, annab see võimaluse tagasitõstetud pakett siiski enne vea teket serverisse laadida. Üleslaadimise järjekord on sorteeritud paketi loomise aja järgi, seega tagasitõstetud pakett on hiljem lisatutest kõrgema prioriteediga. Samal ajal ei katkestata esimese lõime üleslaadimist, sest on võimalus, et esimene lõim saab oma tööga valmis enne, kui seda saaks töö üle võtnud lõim.

Paketi üldjärjekorda tagasitõstmise aeg sõltub järjekorras olevate pakettide arvust. Kui mõni lõim ootab jõude, võib paketi tõsta järjekoda tagasi kohe nominaalaja täitumisel, pikema järjekorra puhul hiljem. Liiga varajane tagasitõstmine suurendaks ebavajalike kordussaatmise arvu.

Kõigi internetiühenduste maksimaalseks kasutamiseks peab pakettide hulk ühe HLS segmendi kohta olema vähemalt sama suur, kui on ühenduste arv. Suurema hulga pakettide korral on paketi suurus väiksem, mis tagab väiksema ajakao juhul, kui vea korral on pakett vaja uuesti saata. Lisaks tuleb arvestada igale paketile lisatavat päist, mis liiga

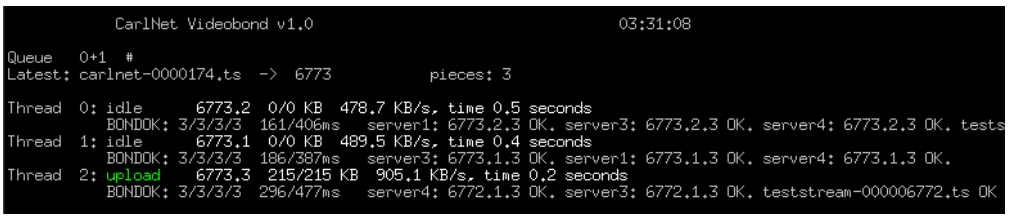
väikese paketi korral muutub mahult oluliseks lisaballastiks. Kasutades HTTP/2 toega servereid, saab paketi päiste ülekulu vähendada.

Kui üleslaadimisjärjekord siiski ühenduskiiruste languse tõttu pikeneb, kasutatakse alternatiivset, väikesema andmemahuga videovoogu, mis annab ka väiksema mahuga paketid.

Kahe kvaliteediastmega HLS videovoo loomine käsureal ffmpeg abil toimub järgmiselt:

```
$ ffmpeg -framerate 25 -i /dev/video0  -threads 8 \
-vcodec libx264 -strict -2 -crf 18 -profile:v baseline -maxrate 1500k
    -bufsize 1835k -pix_fmt yuv420p -flags -global_header -g 125 -c:a
     libfdk_aac -b:a 160k -ac 2 -ar 44100 -cutoff 18000 -hls_time 5 -
    hls_list_size 6 -hls_wrap 9999999  -hls_segment_filename '~/stream
    /test-%07d.ts' -start_number 1 '~/stream/test.m3u8' \
-vcodec libx264 -strict -2 -crf 18 -profile:v baseline -maxrate 500k
    -bufsize 1835k -pix_fmt yuv420p -flags -global_header -g 125 -c:a
    libfdk_aac -b:a 64k -ac 1 -ar 44100 -cutoff 18000 -hls_time 5 -
    hls_list_size 6 -hls_wrap 9999999 -hls_segment_filename '~/stream-
    low/test-%07d.ts' -start_number 1 '~/stream-low/test.m3u8'
```

Kõrgkäideldava ning veakindla üleslaadimise tagamiseks peab programmi käivitaval arvutil olema vähemalt kaks, soovitatavalt aga rohkem, internetiühendust. Marsruutimistabelis tuleb erinevate sisendserverite võrguühendused suunata erinevate interneti-ühenduste kaudu. Rakenduse jaoks ei ole tehniliselt määrav, kas võrguühendused on läbi kommutaatori (ingl *switch*) ühendatud ühe võrgukaardiga või on kodeerival arvutil eraldi võrgukaardid iga ühenduse jaoks. Töökindluse tõstmiseks on soovitatav kasutada vähemalt kahte võrgukaarti ning jaotada internetühendused nende vahel. Ühenduskiiruse paremaks kasutamiseks on soovitatav seadistada iga internetiühendus kasutama mitut erinevat sisendserverit.



Joonis 2. Videobond kasutajaliidese vaade terminalis.

Joonisel 2 on esitatud ekraanitõmmis kasutajaliidesest. Seal toodud näites on kasutusel kolm sisendserverit, iga HLS segment tehakse kolmeks paketiks (ingl *pieces*), üleslaadimisjärjekorras (ingl *queue*) ei ole hetkel ühtegi paketti, üks pakett on üleslaadimisel. Iga lõime kohta näidatakse eraldi üleslaadimise ja serverite statistikat. Esimesel real on näha lõime staatus ning üleslaetava segmendi ja paketi järjekorranumber, üles laetava paketi suurus kilobaitides, üleslaadimise kiirus ning üleslaadimisele kulunud aeg. Teisel real on sisendserveri vastus. Vastus "BONDOK" viitab olukorrale, mil pakett õnnestus serveritele edastada. Vea korral on vastuseks "BONDERR". Järgnev jada kujul 1/2/3/4 on info serverite kohta: seadistatud neli serverit, lubatud aja jooksul saadi ühendus kolme serveriga, kaks serverit andsid protokollile kohase vastuse (paketi sobimise või sobimatuse kohta ning selle põhjuse) ning üks server võttis paketi vastu. 161/406ms näitab, et kvoorumi positiivsete vastuste saamiseks läks aega 161 ms ning kogu sisendserveri töö võttis aega 406 ms. Ülejäänud reas kuvatakse kuni terminali akna lõpuni (ülejäänud lõigatakse maha) keskserverite vastused paketi kättesaamise kohta. Sellisel viisil antakse kasutajale videoülekande ajal reaalajaline ülevaade kogu võrgu tööst.

3.2 Sisendserver

Sisendserveri ülesanne on vahendada andmete liikumist videobondi ning keskserverite vahel.

Sisendserveri loomiseks on kasutatud programmeerimiskeelt PHP. Sellisel viisil on võimalik kulude kokkuhoiuks majutada sisendservereid kodulehe majutusteenuse pakkujate juures, kusjuures sageli piisab kõige odavamast teenuspaketist. Sisendserveri ja keskserveri võib ka samasse serverisse (virtuaalmajutusse) paigaldada ja ühtse komponendina kasutada, kui täiendavate sisendserverite järele vajadust ei ole. Ühe sisendserveri kaudu liigub vaid osa kogu videoülekande andmevoost.

Piiratud ühenduskiiruse tõttu laeb videobond iga paketi sisendserverisse üles vaid ühekordselt. Sisendserver kontrollib paketi kontrollsummat (MD5), ning saadab vigase paketi puhul videobondile kiirelt vastuse ilma keskserverite järel ootamata. Ebaõnnestumise korral palutakse videobondil pakett uuesti laadida, kusjuures pole oluline millisesse sisendserverisse sama pakett järgmisel korral laetakse.

Korrektse kontrollsummaga pakett edastatakse kõigile keskserveritele. Iga keskserver kontrollib paketi omakorda üle ning annab sobivuse korral sisendserverile positiivse

vastuse. Kui keskserverite kvoorum on saavutatud, saadetakse positiivne vastus edasi videobondile, misjärel loetakse selle paketi edastus õnnestunuks. Kui kõik keskserverid annavad vastuse, tagastatakse see koheselt; kui mõni keskserver rikke tõttu aga vastust anda ei suuda, saadetakse vastus eelseadistatud aegumisvälba järel. Koos paketi vastuvõtu kinnitusega tagastatakse ka statistika keskserverite kohta, mis võimaldab kogu süsteemi tööd ülekande paigas paremini jooksvalt jälgida.

Lisaks vahendab sisendserver videobondi ja keskserverite vahel sisselogimist ja sessiooniinfot. Sisendserver edastatavat infot ei puhverdata ja sessioone ei halda. Seepärast ei ole ka oluline, milliseid sisendservereid kasutatakse; ka sisselogimine tuleb teha vaid ühe sisendserveri vahendusel, mis edastab selle info kõigile keskserveritele. Kohapealse puhverdamise puudumine annab võimaluse majutada sisendserverit ka olukorras, kus puudub igasugune võimalus PHP protsessi poolt faile kirjutada.

Sisendserveril kasutajaliidest ei ole; vea korral logitakse info logifaili, ning tagastatakse lühikujul ka videobondile koos paketi tagasilükkamisteatega.

3.3 Keskserver

Keskserveri peamine ülesanne on panna üksikutest pakettidest kokku HLS segmendid, need salvestada, pidada arvet nende järjestuse üle ning taasluua jooksvalt .m3u8 esitlusloend.

Programmeerimiskeeleks on valitud PHP, peamiselt sel põhjusel, et oleks võimalik kasutatada sisend- ja väljundserveritega osaliselt sama koodibaasi.

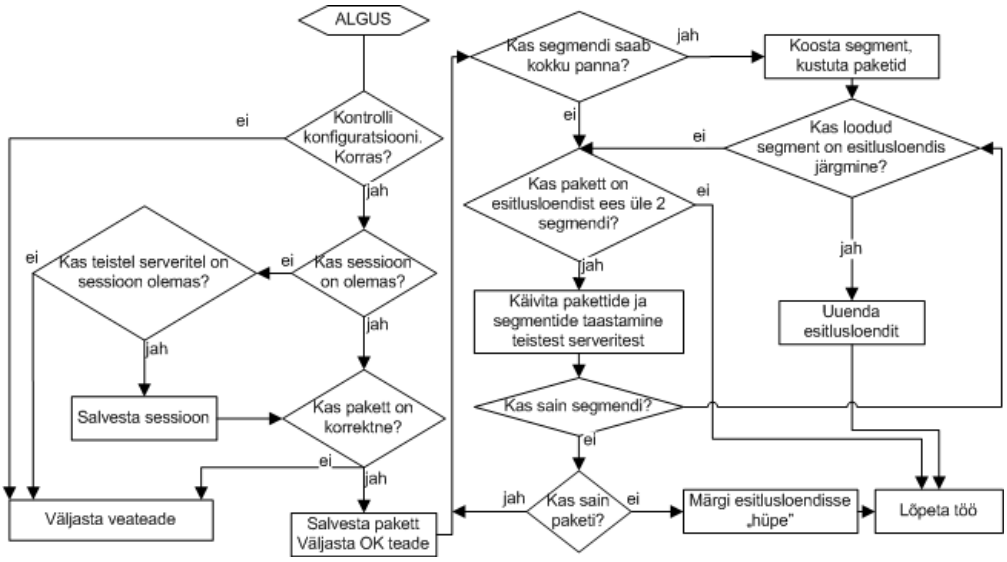
Enne videoülekande alustamist peab videovoo edastaja (videobond) logima sisse kasutades videovoo nime ning selle parooli välistamaks autoriseerimata video ülekannet. Videobondile tagastatakse sessioonivõti, mis kogu videoülekande jooksul edastatakse iga paketiga. Iga paketi saabumisel kontrollitakse sessioonivõtme kehtivust. Kui keskserveris vea tõttu vastav sessioonivõti puudub, siis enne veateate väljastamist kontrollitakse selle olemasolu ka teistelt keskserveritelt.

Kui mõni pakett on ühenduse vea tõttu kaduma läinud, siis tulenevalt sisendserveri kvoorumi kontrollist peab see pakett olema olemas mõnes teises keskserveris. Videobond ei loobu ühe ja sama paketi kordussaatmisestest enne, kui sisendserver on kinnitanud, et miinimumhulk servereid on teatanud paketi kättesaamisest.

Siiski on võimalik olukord, kus kolmeks jagatud segmendi üks pakett on serverites A

ja B, teine serverites B ja C ning kolmas serverites A ja C. Sellisel juhul on videobondi poolt vaadatuna kõik paketid edukalt üles laetud, kuid siiski mitte ükski server ei saa segmendifaili kokku panna. See olukord on lahendatud selliselt, et kui keskserver tuvastab paketi puudumise, siis kontrollitakse juhuslikus järjekorras, võttes arvesse keskserveri konfiguratsioonis määratud prioriteetide kaalu, kõiki keskservereid, kuni vastav pakett või terve segment leitakse.

Keskserveri algoritmi tööd ühe paketi vastuvõtmisel ning esitlusloendi moodustamisel illustreerib joonis 3. Kui parandusprotsess ei suuda segmenti siiski kokku panna, märgitakse esitlusloendisse katkestuskoht ning jätkatakse järgmise õnnestunult kokkupandud segmendiga. Kui videomängija puhvri ajaline kestvus on suurem vahelejäetud osast, kogeb vaataja pildis hüpet ajas edasi. Vastasel juhul, puhvri tühjenemisel, tekib videos lühiajaline katkestus puhvri täitmiseks (puhverdamine), peale mida videovoog jätkub.



Joonis 3. Keskserveri tööpõhimõtte diagramm.

Nõudlikkus keskserveri riistvarale on suurem, sest HLS segmentide kokkupanemine peab toimuma kiirelt, lisaks vajatakse palju kettapinda kuna säilitatakse kogu videovoog.

Ka keskserveritel puudub kasutajaliides.

3.4 Väljundserver

Väljundserveri peamine ülesanne on videovoo edastamine kasutajatele ning vajaliku ribalaiuse hajutamine eri serverite vahel.

Programmeerimiskeeleks on valitud PHP, sest see võimaldab luua väikese kuluga suurema hulga servereid, kasutades selleks tavalise veebimajatuse teenusepakkujaid.

Sõltuvalt ühenduskiirusest ning videovoo ribalaiusest suudab üks keskserver teenindada näiteks 1Gbit/s ühenduse ning 3Mbit/s videovoo korral 250 väljundserverit. Edastatavad segmendid puhverdatakse väljundserveris ning seetõttu laetakse need keskserverist alla vaid ühekordselt. Samade parameetrite korral suudab väljundserver omakorda teenindada kuni 250 vaatajat.

Kõigist keskserveritest valitakse juhuslikult, kuid arvestades serverite konfiguratsioonis seadistatud keskserverite prioriteete, üks keskserver. Keskserveri rikke korral võtab iga ebaõnnestunud katse valimisprotsessis aega kuna tuleb oodata ühenduse aegumist (ingl *timeout*). Valiku õnnestamisel eelistatakse võimaliku ajakao vältimiseks ühte ja sama keskserverit, kuni see on töökorras.

Samas tagatakse veakindlus ning pidevalt toimiv videovoog ka keskserveri rikke või katkestuse korral: ühenduse aegumisel valitakse juhuslikkuse põhimõttel järgmine server, mis jääb eelisserveriks kuni uue võimaliku rikkeni. Ülekande vaataja jaoks ei toimu serverite vahetamisel videovoos vähimatki katkestust.

Kuna väljundserveri edastatavad andmemahud on kõige suuremad, tuleb jälgida serveri majutuse paketi andmemahtu, lisaks ka muid teenusepakkuja piiranguid.

Väljundserveril puudub küll kasutajaliides, kuid on olemas rakendusliides (ingl *API*) statistika väljastamiseks, mida kasutab veebiserveris olev statistika rakendus.

3.5 Veebiserver

Veebiserver tagab vaatajale ligipääsu videovoole. Kasutajale kuvataval veebilehel on otselink videovoo esitlusloendile, võimaldamaks eraldiseisva multimeediarakenduse kasutamist ja HTML5 videoaken video otse veebilehel vaatamiseks.

Veebiserveris genereeritakse esitlusloend, milles on omakorda viited erinevatele väljundserverite esitlusloenditele. Kui üks väljundserver rikke tõttu enam ei vasta, valib videot esitav meediamängija võimaluse korral järgmise väljundserveri ning sõltuvalt

videot esitava programmi töökindlusest toimub ümberlülitumine kasutajale märkamatult.

HLS on voogedastuses väga levinud formaat, kuid kuna enamasti kasutatakse seda staatilise sisu voogedastamiseks, siis paraku see meetod reaalaja video korral iga meediamängijaga alati ei tööta. Juhuks, kui automaatne ümberlülitus ei toimi, on esitusloendis väljundserverid juhuslikus järjekorras. Kui kasutaja video edastamise rikke korral veebilehe taaslaeb, on esimene väljundserver muutunud ning kasutaja saab ülekannet edasi vaadata.

Lisaks on veebiserveris ülekande looja jaoks statistikamoodul, millele ligipääs on parooliga kaitstud. Vaatajate arvu kuvatakse reaalajaliselt uuenevas graafikus, mida uuendatakse iga 10 sekundi järel.

Administraatori tarbeks on veebirakenduses konfiguratsiooni halduse liides, mis võimaldab hallata nii videovoogude kontosid, kui ka kõigi serverite aadresse. Sellesse sisselogimine käib statistikamooduli kaudu, kasutades videovoo "admin" parooli.

Veebiserveri rikke korral juba toimivad videovood vaatajate jaoks ei katke, kuid uued vaatajad ei saa enam vooga liituda, mistõttu on vaja ka veebiserveri käideldavust tõsta. Otstarbekas on kasutada selleks nimeserveriga koormusjaotust. Kui veebiserveri nimele määrata mitu IP aadressi, siis standardne veebisirvik valib sel juhul neist ühe juhuslikult ning, kui sellelt IP-lt ei saa vastust, valitakse järgmine. Seda "vaesemehe koormusjaotiks" kutsutavat meetodit kasutades ei ole vaja tellida kolmandalt osapoolelt koormusjaoti lisateenust (näiteks Amazon ALB [26]) ning saab kasutada olemasolevate serverite resurssi [27]. Oluline on märkida, et see lahendus töötab vaid siis, kui veebiserver rikke korral võrgust eemaldatakse. Juhul, kui server väljastab veateate, siis ka seda loetakse töötavaks vastuseks ning kuvatakse kasutajale.

3.6 Konfiguratsioonihaldus

Omavaheliseks suhtlemiseks peavad serverid teadma teineteise aadresse. Kui võrku lisatakse või eemaldatakse mõni server, siis tuleb omakorda muuta ka kõigi selle serveriga suhtlevate serverite konfiguratsioonifaile.

Kõik serverikomponendid on võimalik tööle seadistada staatilise konfiguratsiooniga. Suurema võrgu mugavamaks haldamiseks on loodud konfiguratsioonifailide automaatne serveritevaheline edastamine.

Kuna konfiguratsioonifailid sisaldavad infot võrgu ülesehituse kohta, on turvalisuse

kaalutlusel need krüpteeritud ning selle funktsionaalsuse kasutamiseks peab staatilises konfiguratsioonifailis olema salvestatud jagatud krüpteerimisvõti.

Konfiguratsiooni edastatakse kahe detailsusastmega. Sisend ja väljundserverid vajavad töötamiseks ainult keskserverite aadresse, mistõttu edastatakse neile ainult osaline konfiguratsioonifail. Kuna keskserverid peavad omavahelises suhtluses vahetama ka videovoogude sisselogimise andmeid, siis turvakaalutlusel kasutatakse nende krüpteerimiseks eraldi võtit. Pidades silmas, et veebiserveris on nii konfiguratsioonihalduse rakendus kui ka väljundserveritega suhtlus, siis peavad seal olema mõlemad võtmed.

Igal konfiguratsioonifaili muudatusel tuleb suurendada selle versiooninumbrit. Failide sünkroniseerimisel valib server endale kõige suurema versiooninumbriga faili. Veebiserveri haldusliidese kasutamisel suurendatakse versiooninumbrit automaatselt.

Konfiguratsioonifailide sünkroniseerimist käivitab perioodiliselt igasse serverisse seadistatud plaanur (ingl scheduler) rakendus. Linux serverite korral on selleks utiliidiks *cron*. Haldusliidesest on administraatoril ka võimalus korraga käivitada kõigi serverite kohene sünkroniseerimine.

Videobondi konfiguratsioonifaili tuleb siiski hallata käsitsi, sest sisendserverite arvu valik sõltub kasutada olevatest internetiühendustest.

4 Rakenduse töökindluse testimine

Videoedastusplatvormi testimisel kasutati kolme internetiühendust, kolme sisendserverit, kolme kekserverit, kolme väljundserverit ning kahte veebiserverit. Testimise tulemused on esitatud järgnevates alampunktides. Iga katse juures on ära toodud selle eesmärk, oodatav tulemus, veaolukorra simuleerimise meetod ning katse tegelik tulemus. Kui on võimalik, siis on testi juures näidatud logifaili asjassepuutuvad read.

4.1 Vead ülekandepaigas

Rakenduse kokkujooksmise tõenäosust võib hinnata väikeseks, kuid seda ei saa mitte kunagi välistada. Otseülekande korral on oluline, et kui kokkujooksmine peaks siiski juhtuma, siis taastuks ülekanne võimalikult kiirelt.

4.1.1 Ffmpeg protsessi viga

Testi eesmärk: töö jätkamise kontroll ffmpeg protsessi vea korral.
Testi oodatav tulemus: videovoog jätkub võimalikult väikese katkestusega. Videobond suudab jätkata olemasoleva nummeratsiooniga.
Testi meetod: kill -9 'ffmpeg pid' && 'ffmpeg käsurida'.
Testi tulemus: kui järgmises mällu loetud esitlusloendis tuvastatati, et segmendi number on muutunud väiksemaks, lisati esitlusloendile märge "#EXT-X-DISCONTINUITY", mis annab meediamängijale teada, et sellel kohal ei ole videovoog ühtne. Sellisel juhul ei teki meediamängijas esitamisel viga ning video näitamine jätkub katkestusteta.
Osaline väljavõte videobondi logifailist:
2020-05-01 01:34:37: Adding 'carlnet-0000060.ts', server segment number 6897
2020-05-01 01:34:55: WARNING: New segment nr 1, last added segment 169
2020-05-01 01:34:55: Adding discontinuity tag
2020-05-01 01:34:55: Adding 'carlnet-0000001.ts', server segment number 6898

4.1.2 Videobondi protsessi viga

Testi eesmärk: töö jätkamise kontroll videobond protsessi vea korral.
Testi oodatav tulemus: videovoog jätkub katkestusteta, videobond jätkab üleslaadimist samast kohast.
Testi meetod: kill -9 'videobond pid' && ./videobond.
Testi tulemus: videobond tuvastas käivitumisel salvestatud andmefaili järgi, et serveris oleva segmendi number on kokku viidav ffmpeg esitlusloendi segmendinumbritega. Videobond jättis uuesti üleslaadimata vanemad failid ning jätkas tööd esimesest üleslaadimist vajavast segmendist.

4.1.3 Internetiühenduse katkemine

Testi eesmärk: kontrollida töö jätkamist ühe internetiühenduse katkemisel.
Testi oodatav tulemus: ühenduse aegumisel antakse töö üle teisele ühendusele.
Testi meetod: katkestuse emuleerimiseks lülitatakse 4G modemi andmeside välja.
Testi tulemus: ühenduse täielikul puudumisel saatis 4G ruuter ICMP protokolli sõnumiga "mitte kättesaadav"ning töö anti üle koheselt sama sekundi jooksul.

Osaline väljavõte videobond logifailist:
2020-05-01 02:00:54: #1: Thread found job: carlnet-0000143.ts.3.3
2020-05-01 02:00:54: #1: curl_easy_perform() failed 7: Couldn't connect to server
2020-05-01 02:00:54: Putting carlnet-0000143.ts.3.3 back to queue from thread 1
2020-05-01 02:00:54: #0: Thread found job: carlnet-0000143.ts.3.3

4.1.4 Internetiühenduse kiiruse langus

Testi eesmärk: kontrollida töö jätkamist ühe internetiühenduse kiiruse langemisel alla miinimumvajaduse.
Testi oodatav tulemus: ajalimiidi ületamisel antakse sama paketi üleslaadimine ka teisele ühendusele.
Testi meetod: ruuteris 10Kbit/s kiirusepiirangu lisamine.
Testi tulemus: kuna leidus vaba ühendus, tõsteti pakett tagasi laadimisjärjekorda juba 3

sekundi pärast. Kuna video esitamisel oli puhvri ajaline maht kordades suurem (sõltub meediamängijast), siis videovoos katkestust ei tekkinud. Vaba ühenduse puudumisel erines vaid aeg.

Osaline väljavõte videobond logifailist:
2020-05-01 02:14:12: #1: Thread found job: carlnet-0000302.ts.2.3
2020-05-01 02:14:15: #1: Thread uploading too long, we have free threads, doing multiple upload
2020-05-01 02:14:15: Putting carlnet-0000302.ts.2.3 back to queue from thread 1
2020-05-01 02:14:15: #0: Thread found job: carlnet-0000302.ts.2.3

4.2 Sisendserveri rike

Sisendserveri võrguühenduse probleem väljendub videobondi jaoks samamoodi kui eelpool testitud videobondi internetiühenduse katkemine. Seda testi ei korratud.

4.2.1 Sisendserveri töövõimetus

Testi eesmärk: kontrollida veahaldust olukorras, kus sisendserveri vastus ei ole ootuspärane.
Testi oodatav tulemus: töö antakse koheselt teisele lõimele üle.
Testi meetod: sisendserveri asemel PHP kood: echo "suvalinevastus";exit;.
Testi tulemus: pakett tagastati koheselt üldjärjekorda, uus lõim võttis töö üle.

Osaline väljavõte videobond logifailist:
2020-05-01 03:37:19: Putting carlnet-0000010.ts.2.3 back to queue from thread 1
2020-05-01 03:37:19: #1: UPLOAD ERROR: UNKNOWN RESPONSE

4.3 Keskserveri rike

4.3.1 Sisendserveri veahaldus keskserveri vea korral

Testi eesmärk: kontrollida sisendserveri käitumist olukorras, kus keskserver ei vasta.
Testi oodatav tulemus: paketid laetakse teistesse keskserveritesse, ühenduse aegumisel tagastatakse videobondile positiivne vastus.
Testi meetod: keskserveri võrgust eemaldamine.
Testi tulemus: videovoo üleslaadimine õnnestus endiselt, suurenes ajaline viide.

Videobondi kasutajaliidesest oli näha, et kolmest seadistatud serverist vastab kaks. Kahe serveri vastuse saamiseks kulus 324 ms, koos võrgust eemaldatud serveriga oodati paketi kättesaamsie kinnitust kokku 2476 ms.
Osaline väljavõte videobondi kasutajaliidesest:
BONDOK: 2/2/2/3 324/2476ms server4: 7690.1.3 OK. server1: 7690.1.3 OK.

4.3.2 Väljundserveri veahaldus keskserveri vea korrall

Testi eesmärk: kontrollida väljundserveri tööd hetkel eelistatud keskserveri rikke korral.
Testi oodatav tulemus: failide laadimiseks valitakse uus keskserver.
Testi meetod: keskserveri võrgust eemaldamine.
Testi tulemus: valiti uus keskserver, videovoog ei katkenud.

Väljavõte väljundserveri logifailist:
2020-05-01 04:18:12 New preferred server: https://videobond1.carlnet.ee/hls

4.3.3 Keskserveris failide taastamine

Testi eesmärk: kontrollida serveri võimet panna segment kokku ka siis kui paketid on üle serverite laiali.
Testi oodatav tulemus: kõik segmendid pannakse kokku ilma vigadeta.
Testi meetod: : videobondile seadistatakse ainult üks väljundserver, mis ignoreerib miinimumi nõuet ning edastab iga paketi ainult ühele serverile.

Testi tulemus: 10 minuti pikkune videovoog esitati ühegi veata.

Osaline väljavõte keskserveri logifailist:
2020-05-02 03:10:49 selfhealed segment: teststream/teststream-000007813.ts
2020-05-02 03:10:50 selfhealed packet: teststream-7814.3.3

4.4 Video edastamine vaatajale

Töötavast serverivõrgustikust pole kasu, kui vaataja ei saa videot vaadata. Seetõttu testiti ka video esitamist vaatajale.

4.4.1 Väljundserveri rike

Testi eesmärk: kontrollida meediamängija väljundserveri automaatse vahetamise võimet.
Testi oodatav tulemus: katkestusteta pidev video esitamine.
Testi meetod: väljundserverite ükshaaval väljalülitamine.
Testi tulemus: test õnnestus osaliselt. Kümnest katsest viiel jätkas video katkestusteta, kolmel kestis katkestus kuni 10 sekundit ning kahel katsel ei jätkanud video edastamist. Veebilehel on kasutusel kaks erinevat video näitamise rakendust, millest mõlemad käitusid sarnaselt. Lähemal uurimisel selgus, et mõlemad kasutavad HLS video näitamiseks sama teeki.

4.4.2 Veebiserveri rike

Testi eesmärk: kontrollida avaliku liidese kättesaadavust ühe veebiserveri rikke korral.
Testi oodatav tulemus: veebilehel oleva videomängija avamine õnnestub ka uuel lehe laadimisel.
Testi meetod: veebiserverite ükshaaval väljalülitamine.
Testi tulemus: kuni üks veebiserver oli töökorras, õnnestus veebilehe laadimine. Kasutaja poolel ei olnud muutus märgatav.

5 Kokkuvõte

Bakalaureusetöö käigus loodi kõrgkäideldav, võrgukoormuse jaotamisega videoülekande voogedastusplatvorm, mille abil on võimalik töökindlalt edastada reaalaja videovoogu, kasutades mitut, ka ebastabiilse või kõikuva kiirusega, internetiühendust.

Töö autorile teadaolevalt ei ole varem loodud sellist rakendust, mis samaaegselt kasutab üleslaadimiseks mitut internetiühendust ning dubleerib voo veakindlalt serverite vahel. Loodud rakendus aitab lahendada probleemi, mis sageli esineb reaalajas videoülekannete tegemisel maapiirkondadest või väliüritustelt, kus on võimalik küll kasutada mitut erinevat internetiühendust, millest aga ükski eraldivõetuna ei ole videoülekande tegemiseks piisavalt kiire ja stabiilne.

Minimaalse viivitusega edastamine saavutati kasutades paralleelselt mitut internetiühendust. Üleslaadimise andmemahu kokkuhoid saavutati video dubleerimisega keskserverite vahel. Rakendus jätkab tööd ka internetiühenduses esinevate probleemide korral. Ühenduse katkemisel jätkatakse tööd ülejäänud toimivate ühenduste abil ning kiiruse langemisel jätkatakse tööd, kasutades madalama kvaliteediastmega videovoogu.

Loodud rakendus on töökindel tarkvara töös esineda võivate tõrgete suhtes. Kodeerimistarkvara (ffmpeg) vea korral jätkatakse minimaalse kaoga olemasolevat HLS videovoogu. Videobond tarkvara rikke korral jätkatakse taaskäivitamisel ülekannet samast kohast katkestuseta. Sisend-, kesk- ja väljundserverid on dubleeritud ja rikke korral lülitatakse ülekanne automaatselt ümber töökorras serverite peale.

Väljatöötatud rakendust testiti võimalike rikete imiteerimise teel. Testide tulemusel osutus rakenduse töökindlus väga heaks ja puuduseid ei leitud.

Valminud rakenduse baasil on võimalik teha täiendavat arendustööd: ülekandekvaliteedi automaatne reguleerimine vastavalt ühenduse kiirusele, automaatselt teise kooderarvuti käivitamine esimese rikke korral ja rikkega serveri(te) automaatne võrgust eemaldamine veahalduse kiirendamiseks.

Viidatud kirjandus

[1] Karlheinz Brandenburg. Mp3 and aac explained, 1999. https://web.archive.org/web/20170213191747/https://graphics.ethz.ch/teaching/mmcom12/slides/mp3_and_aac_brandenburg.pdf.

[2] Faq - via corp. http://www.via-corp.com/licensing/aac/faq.html (02.12.2019).

[3] Bitmovin video developer report, 2019. http://cdn2.hubspot.net/hubfs/3411032/Bitmovin%20Magazine/Video%20Developer%20Report%202019/bitmovin-video-developer-report-2019.pdf (02.12.2019).

[4] Choosing video codecs. https://flowplayer.com/blog/choosing-video-codecs (02.12.2019).

[5] Onur Uzun. I-p-b frames, 2017. https://medium.com/@nonuruzun/i-p-b-frames-b6782bcd1460 (06.05.2020).

[6] Brian Carle. How does h.264 work?, 2012. https://www.salientsys.com/assets/uploads/docs/Understanding_H_264.pdf (02.12.2019).

[7] Dan Schonfeld. Learn more about transport stream. handbook of image and video processing (second edition), 2005.

[8] Http live streaming (hls) - apple developer documentation. https://developer.apple.com/streaming (20.11.2019).

[9] Ffmpeg formats documentation. https://ffmpeg.org/ffmpeg-formats.html#hls-1 (06.05.2020).

[10] Live playlist (sliding window) construction | apple developer documentation. https://developer.apple.com/documentation/http_live_streaming/example_playlists_for_http_live_streaming/live_playlist_sliding_window_construction (20.11.2019).

[11] Creating a master playlist | apple developer documentation. https://developer.apple.com/documentation/http_live_streaming/example_

playlists_for_http_live_streaming/creating_a_master_playlist (06.05.2020).

[12] Using http live streaming. https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html (06.05.2020).

[13] About the ext-x-version tag | apple developer documentation. https://developer.apple.com/documentation/http_live_streaming/about_the_ext-x-version_tag (20.11.2019).

[14] 1.0 introduction. https://developer.apple.com/library/archive/documentation/AudioVideo/Conceptual/HLS_Sample_Encryption/Intro/Intro.html (20.11.2019).

[15] Traci Ruether. 15.11.2019. what is hls (http live streaming)? https://www.wowza.com/blog/hls-streaming-protocol (20.11.2019).

[16] About ffmpeg. http://www.ffmpeg.org/about.html (20.11.2019).

[17] ffmpeg documentation. http://www.ffmpeg.org/ffmpeg.html (20.11.2019).

[18] Live video streaming software | wirecast. https://www.telestream.net/wirecast/ (06.05.2020).

[19] Multistream to 30+ platforms simultaneously | restream. https://restream.io (06.05.2020).

[20] Karin Schade and Jason Lane. Embrace failure: Build a high-availability streaming architecture. https://www.akamai.com/us/en/multimedia/documents/white-paper/embrace-failure-build-a-high-availability-streaming-architecture-whitepaper.pdf (06.05.2020).

[21] Bond - mission critical live video - teradek. https://teradek.com/collections/bond-family (06.05.2020).

[22] Network communication requirements for using self-hosted sputnik servers with core. https://support.teradek.com/hc/en-us/articles/360001735493-Network-communication-requirements-for-using-self-hosted-Sputnik-se (06.05.2020).

[23] Brochure of streamer pro | mushroom networks. https://www.mushroomnetworks.com/brochure-streamer-pro/ (06.05.2020).

[24] Broadband bonding service | mushroom networks. https://www.mushroomnetworks.com/broadband-bonding-service/ (06.05.2020).

[25] H. Parmar and M. Thornburgh. Adobe's real time messaging protocol, 2012. https://wwwimages2.adobe.com/content/dam/acom/en/devnet/rtmp/pdf/rtmp_specification_1.0.pdf (04.05.2020).

[26] What is an application load balancer? - elastic load balancing. https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html (06.05.2020).

[27] Sandra Henry-Stocker. Poor man's load balancing, 2012. https://www.networkworld.com/article/2722741/poor-man-s-load-balancing.html (05.05.2020).

Lisad

I. Paigaldus ja kasutusjuhend

Videot kodeerivasse arvutisse tuleb laadida kataloog "videobond" ning kompileerimiseks käivitada käsk *make*. Vajalik on C kompilaator ja *curli* teek. Faili videobond.conf_sample põhjal tuleb luua konfiguratsioonifail "videobond.conf", milles on näidatud sisendserverite aadressid.

Sisendserveri, keskserveri, väljundserveri ja veebiserveri programmi kood asuvad vastavalt kataloogides "gw", "server", "edge"ja "www". Kõigis neljas kataloogis leiab ka näidiskonfiguratsiooni failist "config_local.php_sample", mis tuleb peale modifitseerimist ümber nimetada "config_local.php". Logifaili kirjutamiseks ning automaatse konfiguratsiooni kasutamiseks peavad olema kataloogid php protsessile kirjutatavad.

Kuigi veebiserveri ja keskserveri "$servers" massivis on rohkem seadistusi kui seda nõuavad sisend ja väljundserver, võib selle siiski samal kujul kopeerida ning viimased ignoreerivad üleliigseid välju.

Videovoo vaatajate statistika mooduli töötamiseks vajab väljundserver *MySQL* või *MariaDB* andmebaasi. Vajaliku andmetabeli loomise käsk on näidiskonfiguratsioonifailis. Andmebaasi puudumine videoedastusplatvormi põhitööd ei sega.

Keskserveri ja veebiserveri näidiskonfiguratsioonis on näidatud ka kahe videovoo seadistus koos paroolidega, millest "admin" nimelise konto abil pääseb veebiserveri haldusliidesesse, mille kaudu on soovitatav koheselt muuta ära admin parool.

Konfiguratsiooni automaatuuenduste seadistamiseks tuleb iga serveri kohta lisada operatsioonissüteemi plaanurisse järgmised käivituskäsud:

```
curl –X POST https://server/gw/post.php?cron
curl –X POST https://server/server/post.php?cron
curl –X POST https://server/edge/?cron
curl –X POST https://server/www/?cron
```

Ilma videosisendkaardita kasutamise testimiseks on lisatud videobond töökataloogi skript, mille abil saab luua HLS voo videofailist. Enne kasutamist tuleb seadistada fail "videobond.conf"mille näidise leiab failist "videobond.conf_sample". Esmalt tuleb käivitada olles videobond kataloogis: "./teststreamfailist.sh /path/to/video.mp4". Peale seda saab käivitada videobondi: "./videobond". Navigeerides veebisirvikuga veebiserveri

aadressile, sisestada loodud voo nimi ning peale seda on näha edastatav video.

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, **Karl Õmblus**,
*(*autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   Kõrgkäideldav võrgukoormuse jaotamisega reaalaja videoülekande lahendus,
   *(*lõputöö pealkiri)

   mille juhendaja on Märt Bakhoff,
   *(*juhendaja nimi)

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Karl Õmblus
08.05.2020