TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

HENRY HYNYNEN

DYNAMIC ADAPTIVE STREAMING OVER HTTP: KATSAUS TEKNOLOGIAAN JA STANDARDIIN

Diplomityö

Tarkastaja: PhD. Marko Helenius
Tarkastaja ja aihe hyväksytty
tieto- ja sähkötekniikan tiedekuntaneuvoston kokouksessa 30.05.2018

TIIVISTELMÄ

HENRY HYNYNEN: Dynamic Adaptive Streaming over HTTP: katsaus teknologiaan ja standardiin
Tampereen teknillinen yliopisto
Diplomityö, 43 sivua, 4 liitesivua
Joulukuu 2018
Tietotekniikan koulutusohjelma
Pääaine: Communication systems and networks
Tarkastajat: PhD. Marko Helenius
Avainsanat: streaming, dash, hls, smooth streaming, hds

Videon suoratoisto on kehittynyt huimasti eteenpäin viimevuosina. Parantuneen laitetuen ja nopeampien laajakaistayhteyksien ansioista suoratoistopalvelut ovat yhä useampien saatavilla ja ne voivat parhaimmillaan korvata perinteiset antenni- ja kaapelitelevisiolähetykset. Tästä on muodostunut myös tärkeä liiketoimintasegmentti media-alalla.

Tässä työssä luodaan katsaus alalla vallitseviin teknologioihin ja vertaillaan niitä toisiinsa. Erityisesti tarkastellaan DASH-standardia, sen vahvuuksia muihin teknologioihin nähden sekä tarkastellaan sen toimintaperiaatteita. Lisäksi tässä työssä tutkitaan HTTP-protokollan soveltuvuutta, vahvuuksia ja heikkouksia videon suoratoistamisessa. Lopussa esitellään DASHin toimintaa käytännössä.

Verrattaessa HTTP:tä, eli TCP-pohjaista protokollaa, vaihtoehtoiseen UDP-protokollaan havaitaan molemmilla olevan heikkoutensa ja vahvuutensa, eikä suoralta kädeltä voida todeta toista protokollaa paremmin suoratoistoon sopivaksi. HTTP:n eduksi voidaan todeta tilattomuus, valmis HTTP:tä tukeva web-infrastruktuuri ja luotettava toimitus. DASH erottuu kilpailevista teknologioista edukseen ollessaan avoin ISO-standardi, hyvin joustava ja modulaarinen teknologia, ilmainen ja sitä on kehittämässä suuri joukko isoja kansainvälisiä yrityksiä, joilla on tahto saavuttaa mahdollisimman laaja tuki asiakkaiden keskuudessa.

Tämän diplomityön tarkoitus on antaa lukijalle käsitys siitä, miksi HTTP-pohjaiset suoratoistotekniikat ovat suosittuja, sekä esitellä erilaisia teknologioita tähän tarkoitukseen. Työssä esitellään näistä tarkemmin DASH, jolla on potentiaalia saavuttaa vallitseva asema tulevaisuudessa avoimuutensa ja joustavuutensa ansiosta kentällä, jota ovat vallinneet tähän asti suljetut tai osin suljetut teknologiat.

ABSTRACT

HENRY HYNYNEN: Dynamic Adaptive Streaming over HTTP: overview to technology and standard
Tampere University of Technology
Diplomityö, 43 pages, 4 Appendix pages
December 2018
Master's Degree Programme in Information Technology
Major: Communication systems and networks
Examiner: PhD. Marko Helenius
Keywords: streaming, dash, hls, smooth streaming, hds

Video streaming has been developing fast during past few years. Extended support on end-user devices and faster internet connections have brought these services available for more people and they are ready to substitute traditional antenna and cable broadcast television. Video streaming has developed an important business among media companies and broadcasters.

This thesis will take a look at the technologies in this field and compare them against each other. Specially the thesis will introduce DASH standard, its pros and cons and how it works. The thesis will also examine suitability of HTTP protocol in streaming. At the end of the thesis will be presented a practical example of DASH in action.

Comparing TCP-based HTTP protocol against alternative option UDP the thesis will point out that both of these protocols have their pros and cons and one cannot simply determine which one is better streaming protocol in general. For the good opf HTTP we can read statelessness, existing HTTP-supporting web infrastructure and reliable delivery. DASH stands out from its competitors for being open ISO standard, flexible and modular, free and there are several big international companies developing and promoting for it. Their goal is to achieve wide client support across devices.

The purpose of this thesis is to give reader insights and understanding of why HTTP-based technologies are so popular and introduce different products for this purpose. The thesis will examine DASH, which has potential to become dominant technology over its predecessors in a field that has been dominated by proprietary technologies. DASH will triumph for it being open and flexible standard.

ESIPUHE

Tämän työn aiheeseen ja sen ymmärtämiseen on vaikuttanut vahvasti työskentelyni Cybercomilla ja erityisesti projektini asiakkaallamme MTV:llä. Kiitänkin MTV:tä siitä että sain käyttää esimerkkinä työssäni palvelua, jota käyttävää jo yli miljoona suomalaista. Erityinen kiitos video back-end -tiimille, joiden kanssa minulla on ollut ilo työskennellä ja joilta olen oppinut paljon vuosien varrella.

Koko opintoaikani olen saanut jaksamista ja myötätuntoa TeLEn kiltahuoneelta ja paljon iloa on ammennettu killan hienosta toiminnasta. Kiitokset monista korttipöydän ääressä vietetyistä luennoista ja hikisistä hetkistä killan takahuoneessa palvelimia asentaessa.

Lopuksi halua vielä kiittää avovaimoani Maria siitä, että hän on kaikki nämä vuodet uskonut minuun, vaikka välillä vietin enemmän aikaa yliopistolla kuin kotona ja opintovuosiakin vierähti pari ylimääräistä.

Tampereella 6.11.2018

Henry Hynynen

SISÄLTÖ

LYHENTEET JA MERKINNÄT

ABR	Adaptive Bit Rate
AWS	Amazon Web Services
CDN	Content Delivery Network
DASH	Dynamic Adaptive Streaming over HTTP
DRM	Digital Rights Management
DVR	Digital Video Recording
HDS	HTTP Dynamic Streaming
HLS	HTTP Live Streaming
HTTP	Hyper Text Transfer Protocol
MPD	Media Presentation Description
MSE	Media Source Extensions
Origin	Palveluntarjoajan palvelin, josta palvelun sisältö on peräisin.
OTT	Over-The-Top, liiketoimintaa jossa videota striimataan Internetin yli.
SDK	Software Development Kit
SCTE	Society of Cable Telecommunication Engineers
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VOD	Video On-Demand

1. JOHDANTO

Videoiden suoratoistopalvelut ovat nostaneet suosiotaan huimasti viimevuosina. Näin voidaan päätellä esimerkiksi alan yritysten tunnusluvuista ja käyttäjämääristä maailmalla: yhdysvaltalaisen Netflixin liikevaihto on kaksinkertaistunut vuoden 2013 4,37 miljardista dollarista 8,83 miljardiin dollariin vuonna 2016 [1] ja niin ikään yhdysvaltalainen verkkokauppajätti Amazonin vuonna 2005 julkaisema palvelu Amazon Prime on kasvattanut liikevaihtoaan 231%, sen ollen vuonna 2016 6,39 miljardia dollaria [2]. Google raportoi Youtubella olevan yli miljardi käyttäjää ja että palvelussa katsotaan päivittäin yli miljardi tuntia videomateriaalia [3]. Suomessa kaikki merkittävimmät TV-yhtiöt ovat julkaisseet omat suoratoistopalvelunsa: Yleisradiolla Areena, MTV:llä Katsomo ja vastikään lanseerattu Cmore, sekä Nelonen Median Ruutu. Cisco system arvioi, että vuoteen 2021 mennessä 82% Internetin liikenteestä on videota [4].

Palveluiden yleistymistä suosion kasvamista lisää parantuneet Internet-yhteydet. Nopeamman yhteyden kautta voidaan toimittaa katsojille parempilaatuista sisältöä kasvaneen kaistan ansioita, jolloin katsojat voivat hyödyntää teräväpiirto-, tai jopa 4k-laatuun kykeneviä päätelaitteitaan. Mobiiliyhteyksien parantuessa sisältö on saavutettavissa milloin ja mistä vain moderneilla älypuhelimilla [5].
Yhdysvalloissa pääkonttoriaan pitävä Akamai Technologies tarjoaa maailmaanlaajuisia pilvi- ja CDN-palveluita viranomaisille, kansainvälisille suuryrityksille ja useille suurille mediataloille [6]. Akamai julkaisee neljännesvuosittain raportin Internetin tilasta siten, kuin se heille näkyy. Vuoden 2016 ensimmäisen neljänneksen raportissa kerrotaan maailman keskimääräisen yhteysnopeuden kasvaneen edelliseltä vuosineljännekseltä 12 % ollen 6,3 Mbps. Mobiiliyhteyksien tilastoissa oli mukana 74 maata tai aluetta, joista yhdellätoista keskimääräinen yhteysnopeus oli yli 10 Mbps ja 53:lla maalla yli 4 Mbps. [7] Vuotta myöhemmin (Q1/2017) kasvua neljännestä kohden oli 2,3%, keskinopeuden ollen tällöin 7,2 Mbps. Vuotuista kasvua oli tullut 15%. Vuoden 2017 ensimmäisellä neljänneksellä 62 maasta tai alueesta 32:lla maalla keskimääräinen mobiiliyhteyden nopeus oli yli 10 Mbps ja 60:llä yli 4 Mbps.

Tällä hetkellä markkinoilla ei ole olemassa vallitsevaa standardia, vaan teknologiakenttää hallitsevat eri yritysten omat teknologiat. Erityisen laajan suosion ovat

saaneet Microsoftin Smooth Streaming, Applen HTTP Live Streaming (HLS) sekä Adoben HTTP Dynamic Streaming (HDS) [8]. Kukin näistä teknologioista vaatii omat sovelluskehyksensä ja mediaformaattinsa, mikä aiheuttaa ongelmia niin sisällöntuottajille, laitevalmistajille, sovelluskehittäjille kuin kuluttajillekin. Sisällöntuottajat joutuvat tuottamaan materiaalinsa kaikille alustoille sopivaksi, elleivät he halua sulkea pois asiakkaita palvelunsa piiristä. Laitevalmistajien täytyy tukea erilaisia sovelluskehyksiä tai teknologioita, tavallisesti käyttöjärjestelmä- tai jopa rautatasolla, sekä julkaista näihin päivityksiä. Sovelluskehittäjät vastaavasti tekevät eri alustoille (mobiililaitteet, selaimet, televisiot ym.) sovellukset. Lopulta kuluttajan pitää sopeutua vallitseviin olosuhteisiin ja asentaa päätelaitteelleen sopivat ohjelmat ja ajurit. Jos loppukäyttäjällä on käytössään useampi palvelu, täytyy hänen pahimmillaan asentaa ja päivittää useita näistä tarjolla olevista sovelluspaketeista. Kaiken tämän lisäksi jokaisella näistä hallitsevista teknologioista on oma DRM (Digital Rights Management)-tekniikkansa sisällön salaamiseksi.

Dynamic Adaptive Streaming over HTTP, eli tavallisemmin DASH, on teknologia, joka pyrkii ratkaisemaan tämän ongelman. DASH on julkinen ja avoin standardi, jonka kehittämisessä ovat mukana muun muassa edellä mainituista Adobe ja Microsoft, sekä tunnetuista media-alan yrityksistä mm. Netflix [9]. Projektin tavoitteena on luoda yksi joustava ja vakioitu tapa toimittaa sisältöä loppukäyttäjälle ja taata hyvä katselukokemus. Tämän diplomityön tavoitteena on selvittää DASHin tuomia etuja vertailemalla sitä markkinoita hallitseviin teknologioihin, sekä esittelemällä sen muita hyväksi havaittuja ratkaisuja. Työssä myös esitetään esimerkki teknologiasta käytännössä, sekä esitellään yleisesti videon suoratoistoon liittyviä komponentteja ja arkkitehtiratkaisuja. Vertailu tehtiin tutkimalla aiheesta tehtyjä tutkimuksia ja standardeja, sekä vertailemalla yritysten tuotteistaan julkaisemia manuaaleja ja käyttötapausesimerkkejä.

Luvussa kaksi on kerrottu tiedon hakemisesta ja työhön käytetyistä lähteistä. Luvussa kolme punnitaan HTTP:n hyviä ja huonoja puolia videon suoratoistossa, sekä verrataan sitä UDP:hen ja multicastiin pohjautuviin ratkaisuihin. Neljäs luku esittelee itse DASHia ja pureutuu erityisesti sen tapaan esittää media asiakassovelluksille. Viidennessä luvussa vertaillaan keskenään HTTP-pohjaisia toteutuksia ja kuudennessa luvussa on kerrottu miten DASHia on sovellettu käytännössä suomalaisessa MTV Oy:ssä.

Tästä eteenpäin työssä käytetään myös termiä *streamaus* videon tai audion suoratoistosta puhuttaessa. Tätä termiä käytetään yleisesti alalla myös suomen kielellä ja on toisinaan mukavampi käyttää kuin suora suomenkielinen vaihtoehto *suoratoisto*. Suora käännös streamille olisi virta, mikä tavallaan kuvastaakin hyvin sitä, kuin-

ka data virtaa yhteen suuntaan näennäisesti katkeamatta. Vastaavasti yksittäisestä esityksestä, joka streamataan Internetin yli, käytetään termiä *stream*.

2. TUTKIMUSMATERIAALIN KARTOITUS

Tämän työn kannalta oleellisin tietolähde on ollut virallinen ISO-standardi (ISO/IEC 23009-1:2014(E): Information technology Dynamic adaptive streaming over HTTP (DASH) Part 1: Media presentation description and segment formats). Standardi määrittelee oleellisimmat toiminnallisuudet ja ominaisuudet DASHissa. Standardi on kaikille saatavilla ISO:n web-sivuilla [10]. Osat 2-7 käsittelevät DASHin erikoistapauksia ja referenssiohjelmistoja ja arkkitehtuuria. Lisäksi DASHin spesifikaatiota ja yhteensopivuutta on täydennetty ”Interoperability points”-dokumenteilla, jotka on julkaistu DASH Industry Forumin (DASH-IF) sivuilla [11]. Näihin dokumentteihin on viitattu tekstissä tarpeen vaatiessa, kun teknologian erityisominaisuuksia on esitelty. DASH-IF on useiden eri yritysten ja ryhmittymien yhteenliittymä, joka on vastuussa DASHin kehittämisestä. DASH-IF julkaisee dokumenttien lisäksi paljon hyödyllistä informaatiota niin DASHista kuin foorumista itsestään. Google Scholar -työkalu [12] on tarkoitettu tieteellisten julkaisujen hakemiseen. Tätä hakukonetta hyväksikäytten hain julkaisuja aiheesta. Suurin osa hakutuloksista ei kuitenkaan tuonut juuri oleellista uutta tietoa standardin lisäksi, sillä kuten tämä opinnäytetyökin, suuri osa hakutuloksista käsittelivät aihetta melko yleisellä tasolla. Jotkin artikkelit kuitenkin auttoivat paremmin ymmärtämään tiettyjen DASHin ominaisuuksien käyttötapaa ja tarkoitusta paremmin.

Google Scholar oli hyödyllisempi työkalu haettaessa tietoa eri tietoliikenneprotokollista videon suoratoiston suhteen. Tavallisimpia hakusanoja olivat "http", "streaming", "dash", "udp streaming"ja "http streaming". Protokollien vertailussa käytin apuna myös Tampereen Teknillisen yliopiston (TTY) kursseilla käytettyjä kurssimateriaaleja [13]. Kurssien oppimateriaalina oleviin kirjoihin pääsin käsiksi TTY:n kirjaston verkkopalvelun kautta.

Opinnäytetyössä vertailtiin myös DASHia muihin suoratoistoteknologioihin. Vertailukohtina olevat teknologiat ovat kaupallisia tuoetteita, joiden tekijöinä ovat Apple, Microsoft ja Adobe. Tärkein tietolähde näihin tekniikoihin olivat firmojen omat websivut [14][15][16]. Vaikka näiden yritysten tekniikat on tarkoin lisensoitu ja osin maksullisia, julkaisevat ne vapaasti ohjeita ja standardeja kaikille Internetissä. Lisäksi yrityksillä on kattavat tukipalvelut ja esimerkiksi foorumeilla yritysten edustajat

vastaavat ihmisten kysymyksiin ja keskustelut ovat myös kenen tahansa luettavissa. Microsoft on yksi DASH-IF:n perustajajäsenistä, ja siksi tarjoaakin sivuillaan ohjeita myös DASHin käyttöön heidän tuotteillaan. Keskenään kilpailevat yritykset myös usein vertailevat omaa tuotettaan muiden valmistajien tuotteisiin. Vaikka yritykset pyrkivätkin saamaan oman tuotteensa muita parempaan valoon, saa näistä vertailukohdista näkemystä ja uusia kysymyksiä myös tällaiseen opinnäytetyöhön.

Tärkeäksi tietolähteeksi muodostui myös kaupallinen julkaisu Streaming Media. Printatun ja verkkolehden lisäksi yritys järjestää konferensseja ja tekee tutkimusta alan parissa. Lehteen kirjoittavat media-alan ammattilaiset ja tutkijat. Verkkosivuilta [17] pystyy hakemaan lehdessä julkaistuja artikkeleita. Jos tässä työssä ei olekaan useasti viitattu lehden artikkeleihin, on teksteistä ollut hyötyä minun ymmärtämiseni ja tiedon hakemisen kannalta. Olen kyennyt löytämään artikkeleissa mainitut asiat teknologioiden omista dokumentaatioista ja valmistajien web-sivuilta.

Käytännön osuudessa olen hyödyntänyt omaa osaamistani ja työtä, jota olen tehnyt MTV:n tiimin kanssa. MTV:n videopalvelut ovat olleet toiminnassa kauan ennen kuin itse aloitin työt suoratoistopalvelun parissa, mutta olen osallistunut palvelun ja sen infrastruktuurin kehittämiseen useamman vuoden. Kehitystyössä olen joutunut nojaamaan näihin samoihin dokumentteihin mistä olen kirjoittanut tässä kappaleessa. Olen myös päässyt työskentelemään alan huippuosaajien kanssa ja käymään kansainvälisissä alan tapahtumissa.

3. HTTP:N TUOMAT EDUT JA HAITAT SUORATOISTOSSA

Aiemmin johdannossa esitellyt, suurta suosiota nauttivat suoratoistoteknologiat, Smooth Streaming, HDS ja HLS, perustuvat kaikki HTTP-protokollan ja täten siis TCP-protokollan käyttöön. Monissa sovelluksissa käytetään myös UDP-protokollaan pohjautuvia tekniikoita, joilla on omat etunsa TCP-pohjaisiin tekniikoihin nähden. Tässä kappaleessa pyritään valottamaan näiden molempien hyviä ja huonoja puolia, sekä perustelemaan se, miksi HTTP on nyt niin yleisessä käytössä. Tämän diplomityön aiheena oleva DASH perustuu niin ikää HTTP-protokollaan.

3.1 Reaaliaikaisuus ja toiston laatu

UDP:ta käyttävät sovellukset toteuttavat paremmin suuret reaaliaikavaatimukset. Reaaliaikaisuus on tärkeä vaatimus esimerkiksi urheilulähetyksissä. Katsoja haluaa elää hetkessä oman suosikkijoukkueensa kanssa. Urheilutapahtuman kokemusta voi myös heikentää ympäristöstä kuuluva juhlinta maalin seurauksena tai sosiaalisesta mediasta ja pikaviestimistä tulevat viestit. Käytännön sovellutuksissa esimerkiksi valvontakameroiden kuvan halutaan olevan mahdollisimman reaaliaikaista, kuten myös mahdollisesti etäohjattavien laitteiden lähettämän kuvan. Nopeat vasteajat korostuvat nimenomaan livelähetysten tapauksessa. Esimerkiksi elokuvaa tai tallennettua ohjelmaa katsottaessa tällaista vaatimusta ei tietenkään ole. Päinvastoin, käyttäjä saattaa haluta nimenomaan pysäyttää toiston hetkellisesti tai kelata tiettyyn kohtaan videossa. Lähtökohtaisesti UDP on yhteydetön, eikä takaa tiedonsiirron onnistumista. Tästä seuraa se, että asiakkaan ja palvelimen välillä on vähemmän liikennettä ja asiakas voi vain vastaanottaa paketteja ilman kuittauksia tai liikenteen kontrollointia, mikä vähentää pakettien käsittelyn viivettä. UDP:n luonteesta ja suunnittelusta johtuen datagrammissa on TCP-segmenttiin verrattuna vähemmän otsikko- ja kontrollitietoja. OSI-mallissa siirtokerroksen datapaketin mahdollisen koon määrittelee verkkokerroksen protokolla, tässä tapauksessa IP-protokolla, joten UDP-datagrammin hyötykuorman teoreettinen maksimikoko on suurempi kuin TCP-segmentin. Tästä syystä UDP:n hyötysuhde on parempi kuin TCP:llä.

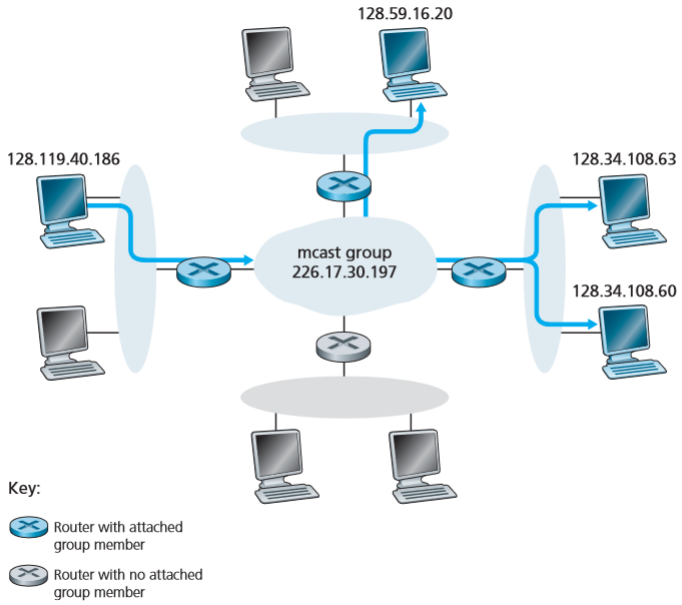
Vastavuoroisesti edellä mainituista ominaisuuksista seuraa epävarmuus. Koska tiedonsiirtoa ei kontrolloida, saattaa dataa hävitä matkalla epäonnistuneen lähetyksen seurauksena tai paketit voivat saapua asiakkaalle väärässä järjestyksessä. Videon ja audion toistossa tämä aiheuttaa häiriöitä. Häiriöiden laatua ei voida yksiselitteisesti selvittää, koska tämä riippuu hävitetyn datan sisällöstä ja määrästä. Kuitenkin pienetkin häiriöt aiheuttavat kuluttajalle huonon katselukokemuksen, varsinkin jos ongelmat toistuvat esimerkiksi huonon yhteyden tai päätelaitteen resurssien puutteen takia. Edellä mainitussa urheiluesimerkissä kriittisellä hetkellä tapahtuva toiston katkeaminen voi aiheuttaa suurtakin närää loppukäyttäjien piirissä. TCP:n eduksi voidaan siis lukea tiedonsiirron luotettavuus [5].

Videotoiston yhtenäisyys on korostunut kun käyttöön on otettu adaptiivisen streamauksen keinot, eli ABR (Adaptive Bit Rate) streaming. Konseptissa on kyse siitä, että olosuhteiden vaihdellessa asiakas voi automaattisesti valita, tai hänelle voidaan tarjota eri laatuja samasta videosta tai audiosta. Kun esimerkiksi verkon suorituskyky laskee tai päätelaite ei kykene käsittelemään sen hetkistä esitystä, pyydetään palvelimelta samaa heikompaa laatua. Esimerkiksi videotoistossa voidaan vaihtaa Full-HD -tasoinen 1920x1080 pikselin kokoinen videokuva HD-Ready -laatuun 1280x720 pikseliä, joka siis vaatii vähemmän tiedonsiirtokaistaa ja prosessointitehoa (riippuen tietysti olennaisesti käytetyistä laatuparametreista videon transkoodauksessa lähdemateriaalista asiakkaille tarjottavaksi materiaaliksi). Verkon suorituskyvyn palautuessa voidaan taas vaihtaa korkeampaan laatuun. Tavoitteena luonnollisesti on suorittaa vaihdos nopeasti ja katkaisematta toistoa.

3.2 Multicastin ongelmat

Multicastin tarkoituksena on lähettää paketti kerralla suurelle joukolle vastaanottajia. Jokaisesta vastaanottajasta ei voida tai kannata pitää erikseen kirjaa, vaan paketit lähetetään erityisiin multicast-ryhmiin. Jokaisella ryhmällä on oma osoite, jolloin verkkolaitteet tietävät mihin dataa pitää lähettää edelleen. Päätelaitteet voivat liittyä näihin ryhmiin käyttämällä IGMP-protokollaa. Liityttyään multicast-ryhmään päätelaitteet vastaanottavat lähimmältä reitittimeltään liikennettä. Reitittimet ovat vastuussa pakettien kopioinnista ja reitityksestä. Kuva 3.1 pyrkii havainnollistamaan tällaista ryhmää ja reititystä: osoitteesta 128.119.40.186 lähetetyt paketit monistetaan reitittimissä, jotka ovat liittyneet ryhmään 226.17.30.197. Reitittimet lähettävät paketit edelleen loppukäyttäjille. Luonteensa takia multicastia hyödyntävät suoratoistopalvelut hyödyntävät tavallisesti UDP-protokollaa: kun viesti lähetetään useaan kohteeseen eikä niistä pidetä verkkotasolla kirjaa, on haasteellista ja jokseenkin tarpeetonta toteuttaa TCP-tyyppinen luotettava tiedonvälitys. [13]

Multicast sopii videon streamaukseen tietyissä tapauksissa hienosti. Lähdepalvelimen kuorma vähenee merkittävästi kun sen ei tarvitse vastata jokaisen clientit pyyntöihin erikseen. Dataa lähetetään parhaassa tapauksissa vain muutamaan (multicast) osoitteeseen ja tällöinkin pakettien duplikoinnin ja reitittämisen hoitaa verkkoinfrastruktuuri. Loppukäyttäjät saavat datansa lähimmältä reitittimeltä, jolloin latenssit ovat pienet ja toimitus luotettavampaa verrattuna Internetin läpi kulkevaan liikenteeseen. Lisäksi palvelimelle ei välttämättä tarvitse lähettää viestejä yhteyden muodostamisen jälkeen, vaan dataa (audiota ja videota) vastaanotetaan sitä mukaa kun sitä verkkoon lähetetään. Live-videoissa saavutetaan näiden ansiosta myös paremmat reaaliaikavaatimukset.



Kuva 3.1 *Kaaviokuva multicastin toiminnasta. Lähde: Kurose & Ross: Computer Networking - A Top-Down Approach[13], sivu 407.*

HTTP:llä saavutetaan tähän verrattuna kuitenkin joitain etuja. Koska asiakas pyytää palvelimelta dataa erikseen tarpeen mukaan, voidaan videota ”kelata” helpommin paikasta toiseen. HTTP byte-range -pyynnöllä saadaan videotiedostosta juuri se osa kun halutaan. Multicastissa kaikille lähetetään samaa dataa, eikä yksittäinen käyttäjä voi käyttää olemassa olevaa yhteyttä haluamansa datan hakemiseen. Jotta videossa päästäisiin taaksepäin, pitää asiakasohjelmiston tallettaa videota paikallisesti. Tämä kuluttaa resursseja päätelaitteelta, joten sovelluksen kehittäjän tulee miettiä, kuinka paljon materiaalia puskurissa säilytetään. Mikäli sovellus käyttää UDP-protokollaan pohjautuvia tekniikoita, pitää sovelluskehittäjän myös ottaa huomioon erilaiset virhetapaukset, kuten hävinneet tai korruptoituneet paketit, sekä

pakettien järjestys. Yksittäisen paketin puuttuminen tai osittainen korruptoituminen harvoin aiheuttaa vakavia ongelmia toistossa, mutta huonoilla tai ruuhkaisilla yhteyksillä tilanne saattaa eskaloitua oikeaksi ongelmaksi. HTTP:llä toteutetussa streamissa TCP pitää huoleen pakettien uudelleenlähetyksistä, ruuhkanhallinnasta ja virheenkorjauksista.

Oman ongelmansa tuo ABR-tekniikka ja toiston laadun vaihtelut. Päätelaitteen tai verkon suorituskyvyn heikentyessä merkittävästi ei välttämättä saada videota toistettua halutulla kuvanlaadulla tarpeeksi nopeasti. HTTP-pohjaisista tekniikoista esimerkiksi DASH taklaa ongelman niin, että asiakas pyytää ja sille vastataan heikommalla laadulla tuotetulla videolla, joka löytyy serveriltä valmiiksi transkoodattuna [18]. Tällöin laadun vaihto on teknisesti ottaen saumatonta, kun HTTP pyynnöt ja vastausten käsittely eivät käytännössä eroa toisistaan. Multicastin kanssa laadun vaihtaminen ei ole yhtä yksinkertaista. Edellisessä kappaleessa kuvattiin multicastin ongelmia virheenkorjaukseen liittyen, ja samasta syystä laadun vaihtaminen on haastavaa multicast-streamissa. Vanha yhteys pitää katkaista, avata uusi uuteen multicast grouppiin jotta saataisiin vastaanotettua eri laatuista videota. Sama prosessi pitäisi toistaa taas verkon tai laitteen suorituskyvyn parantuessa toiseen suuntaan. Tätä ongelmaa on kylläkin saatu helpotettua. Yksi tekniikka on lähettää varsinaisessa streamissa heikkolaatuista kuvaa, ja sitten erillisessä streamissa tai streamin videota sisältävien pakettien välissä metadataa, jonka avulla kuvanlaatua saadaan skaalattua ylöspäin. Menetelmä vaatii kuitenkin paljon algoritmeilta, jotka tietoa pakkaavat ja taas rakentavat uudelleen, sekä päätelaitteilta jotka lopulta joutuvat varsinaisen työn tekemään.

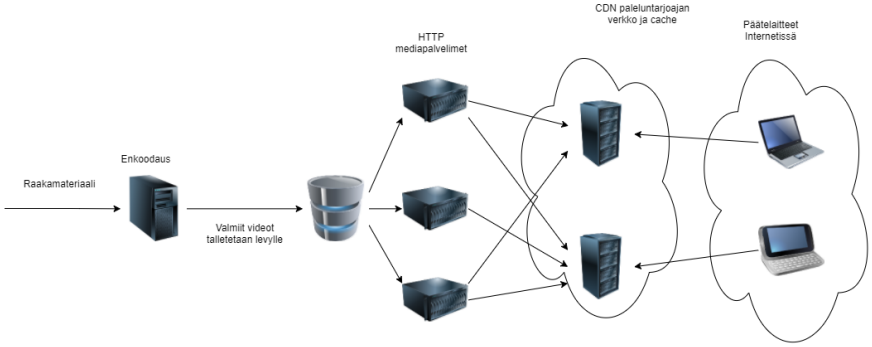
Kuten aiemmin on mainittu, multicastissa on suuri rooli verkkoinfrastruktuurilla, joka on vastuussa liikenteen reitittämisestä ja pakettien monistamisesta, sekä multicast-ryhmien muodostamisesta. Suurimman osan Internetin verkkolaitteista omistavat kuitenkin erilaiset teleoperaattorit, Internet-palveluntarjoajat ja muut suuryritykset. Useimmiten aloittelevalla startup-yrityksellä tai suurellakaan kansallisella toimijalla ole resursseja tai lupia rakentaa kattavaa jakeluverkostoa sisällölleen. Näin ollen monopoli multicast-jakelussa on niillä, ketkä verkkoinfrastruktuurin omistavat, ja muiden toimijoiden laajamittainen multicast-toiminta on joko kiellettyä tai ainakin maksullista. Suomalaisista operaattoreista mm. Telialla, Elisalla ja DNA:lla on omat videopalvelunsa.

3.3 Vaadittava infrastruktuuri

Koska DASH on HTTP-protokollaan perustuva tekniikka, voidaan sisältöä jakaa tavallisilla web-palvelimilla, kuten Apache, IIS tai Nginx. Asiakkaalle, esimerkik-

si selaimelle, on toimitettu tarvittavat tiedot, jotta palvelimelta voidaan pyytää eksplisiittisesti tarvittavat tiedostot, eikä palvelimen tarvitse tällöin tehdä muuta kuin vastata pyyntöön.

Jaeltava materiaali voi olla yksi yhtenäinen tiedosto, tai lähdetiedosto voidaan transkoodata useaan pienempään osaan ja nimetä kukin näistä tiedostoista niin, että asiakasohjelma kykenee näitä pyytämään [19]. Jos jaettava video on yhdessä tiedostossa, tulee palvelimelta pyytää tiettyä osaa tavujärjestyksen perusteella (byte range) [20]. Palvelinohjelmisto voi myös sisältää älykkyyttä, joka esimerkiksi aikakoodien perusteella osaa pilkkoa yhdestä lähdetiedostosta oikeasta kohtaa oikean mittaisia osia.

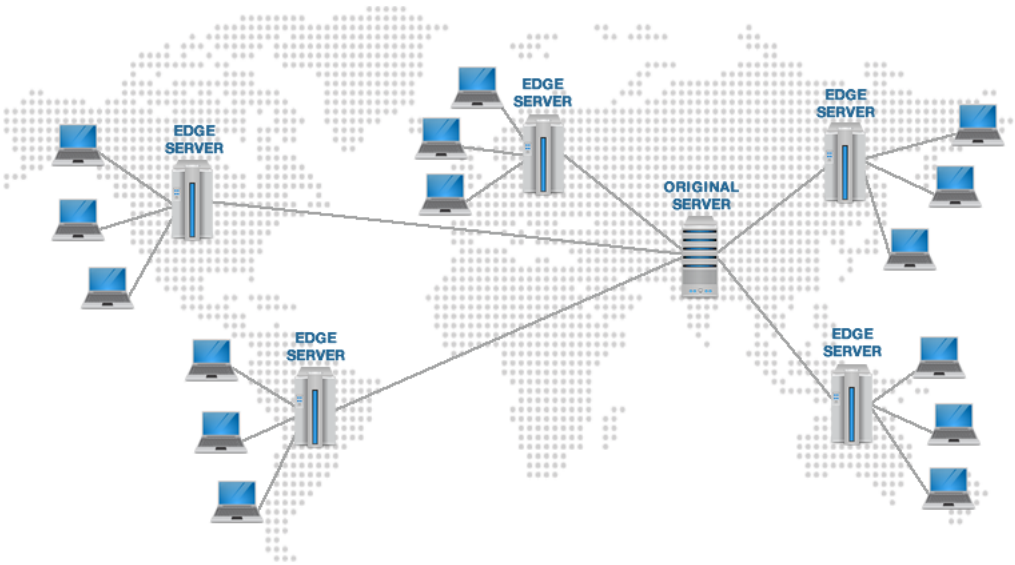


Kuva 3.2 *Sisällöntuottajan jakeluinfrastruktuuri*

Ominaista HTTP-liikenteelle on toistuvien pyyntöjen tallettaminen välimuistiin (cache). Usein toistuviin identtisiin pyyntöihin voidaan vastata palvelimen välimuistista, jolloin palvelimen ei tarvitse erikseen prosessoida jokaista pyyntöä erikseen. Välimuisti sijaitsee palvelimella tavallisesti keskusmuistissa tai tavallista massamuistia nopeammalla kiintolevyllä, jolloin itse datan noutaminen ja verkkoon kirjoittaminen nopeutuu. Internetissä hyödynnetään myös erityisiä jakeluverkkoja (Content Delivery Network) [21]. Usein tällaista palvelua tarjoavat suuret Internet-yritykset ja operaattorit, joilla on tarpeeksi resursseja palvellakseen suuria käyttäjämääriä ja isoja maantieteellisiä alueita. CDN-palveluntarjoaja voi myös omistaa runkoverkkoyhteyksiä tai välittää liikennettä Internetissä muiden toimijoiden välillä. Palveluntarjoajalla on palvelinkeskuksia, eli noodeja, useassa eri paikassa. Asiakkaat pyytävät sisältöä CDN:ltä, jolloin pyyntö pyritään ohjaamaan lähimpään CDN-noodiin, jossa välimuisti sijaitsee [21]. Jos haluttu sisältö löytyy välimuistista, palautetaan tämä asiakkaalle.

Kuvassa 3.2 palveluntarjoaja käyttää useata origin-palvelinta, jotka lähettävät dataa CDN-palveluntarjoajan verkkoon niin sanoituille reunapalvelimille, eli edge-

palvelimille. Tätä CDN-verkkoa avaa paremmin kuva 3.3, jossa on nämä edge-palvelimet sijaitsevat eri maanosissa ja palvelevat paikallisia Internetin käyttäjiä. Jos sisältöä ei löydy, CDN hakee sisällön palvelun tuottajalta (origin), tallettaa sen välimuistiinsa ja tarjoilee loppukäyttäjälle. Näin seuraava käyttäjä joka pyytää samaa materiaalia saa vastauksensa taas nopeammin. [5]



Kuva 3.3 *Kaaviokuva CDN:n toimintaperiaatteesta. Origin server sijaitsee Intiassa, CDN-noodit (Edge server) ympäri maailmaa. Lähde: https://f9official.com/wp-content/uploads/2017/08/how-cdn-works.png*

Sen lisäksi että asiakkaita voidaan palvella nopeammin, menetelmä vähentää liikennettä ja kuormitusta origin-palvelimilla. CDN-verkot vähentävät huomattavasti Internetin liikennettä. Erityisesti videostreamauksen kaltaisissa sovelluksissa saavutetaan suuria etuja median luonteen takia. Kokonainen elokuva korkealla laadulla voi olla kooltaan sadoista megatavuista gigatavuihin. Suosittujen tapahtumien, kuten urheilu- tai uutistapahtumien aikaan katsojia voi isoilla palveluilla olla satoja tuhansia, jolloin origin-palvelimilla vaadittaisiin hetkellisesti valtavasti enemmän kapasiteettia kuin mitä normaalisti. Normaalitilanteessakin voidaan vähentää palvelinresursseja, kun pyyntöjä tulee tuhansien yksittäisten IP-osoitteiden sijasta muutamasta CDN-noodista. [8] [5]

3.4 Päätelaitteet ja ohjelmistot

HTTP-protokolla ja Internet-selaimet ovat keskeisessä roolissa nykyaikaisissa mobiililaitteissa ja tietenkin tietokoneissa. Koska infrastruktuuri on kehittynyt pitkälle ja laajalle, on sen käyttöönotto helppoa ja mutkatonta sovelluskehittäjille ja palveluntarjoajille. Erilaisia kirjastoja ja työkaluja sovelluskehityksen tueksi on saataville useimmille ohjelmointikielille ja laitteistotason tuki löytyy lähes kaikista älypuhelimista, tableteista ja uusista televisioista. Luonnollisesti myös henkilökohtaiset tietokoneet ovat panneet alulle koko Webin kehityksen.

HTTP:n tilattomuus on suuri hyöty langattomissa laitteissa, jotka liikkuvat verkosta toiseen [5]. Tilallisilla protokollilla verkon vaihtuessa yhteys saattaa katketa suhteellisen pitkäksikin aikaa, jolloin uuden yhteyden muodostaminen ja tilatiedon hakeminen tai menettäminen voivat aiheuttaa ongelmia sovelluskehittäjille ja antaa huonon käyttökokemuksen loppukäyttäjälle. Erityisesti DASHissa logiikka on päätelaitteen ohjelmistossa ja ilman erityistä tilatietoa sovellus voi palvelimelta pyytää HTTP:llä dataa heti yhteyden palauduttua, eikä palvelimen tarvitse tietää mitään laitteen tilasta [21]. DASHin toimintaa käsitellään tarkemmin myöhemmin tässä työssä. Myös mobiililaitteiden kehitys tukee olemassa olevien ja käytössä olevien teknologioiden hyödyntämistä; mobiililaitteiden suorituskyky ja tallennustila vastaavat jo heikkotehoisia tietokoneita ja mobiiliverkot voivat olla monin paikoin suorituskykyisempiä kuin kiinteät verkot. Tällöin ei ole tarpeen käyttää tai kehittää erityisen kevyitä protokollia tai metodeja videoiden streamaamiseksi.

Julkisia langattomia verkkoja käytettäessä on myös käytännöllistä käyttää tunnettuja protokollia tietoturvan kannalta. HTTP ja salattu HTTPS liikenne on yleisesti sallittu julkisissa verkoissa [21]. Mikäli hyviä tietoturvakäytänteitä on noudatettu, kaikki tietoliikenneportit ovat suljettuja ja tarpeelliset yhteydet erikseen sallittu. Mikäli sovellus käyttää jotakin muuta protokollaa, pitää tietoliikenteen kulkea jonkin toisen portin kautta ja tällöin yhteyden palvelimille voivat olla estettyjä, eikä streamaus tällöin luonnollisesti toimi [8]. [5]

4. DASH

Lyhenne ”DASH” tulee englanninkielisestä termistä ”Dynamic Adaptive Streaming over HTTP”. Nimi kuvaa sen oleellisimpia ominaisuuksia: teknologia perustuu HTTP-protokollaan ja on dynaamisesti adaptoituva. Jälkimmäisellä tarkoitetaan sitä, että DASH toteuttaa ABR-mallin: toiston laatua ja ominaisuuksia voidaan muuttaa saumattomasti ja keskeyttämättä toistoa, ja esityksen sisältöä voidaan muokata toiston aikana.

DASHin kehitystä edistää ja promotoi DASH Industry Forum (DASH-IF), joka on vuonna 2012 perustettu foorumi ja koostuu useista teknologiateollisuuden yrityksistä, kuten Microsoft, Apple ja Netflix. DASH-IF:n jäsenet jakavat näkemyksen siitä, että videon jakeluun tulisi olla yhtenäinen ja helppo tapa yli valmistajien tai laitteiden rajojen. Alun perin teknologian kehitti International Organization for Standardization (ISO), joka hyväksyi DASHin määritelmän Marraskuussa 2011. Standardin työstäminen alkoi jo lokakuussa 2010. Standardi ISO/IEC 23009-1:2012 julkaistiin huhtikuussa 2012. Standardia oli jo alkuvaiheessa kehittämässä DASH-IF:n jäsenyrityksiä. DASH muistuttaakin ominaisuuksiltaan markkinoilla yleisesti käytettyjä Applen, Microsoftin ja Adoben teknologioita.[11][22]

DASH-IF listaa Internet-sivuillaan tärkeimpiä ominaisuuksia, joita teknologia tarjoaa alalle [11]: rippumaton kansainvälinen standardi, kyky toimittaa asiakkaalle usea vaihtoehtoinen video- tai audioraita, mahdollisuus valita ja toistaa näistä mieleiset raidat, yhtenäinen metodi salauksen toteuttamiseen, selkeä malli soittolistatiedostoille, segmentoimattomat lähdetiedostot, tehokas mainosten injektointi, tuki usealle eri CDN-palveluntarjoajalle tai CDN-noodille ja mahdollisuus kuvailla sisältöä soittolistatiedostossa. Lisäksi käyttöönotto on helppoa, sillä sisältöä voidaan jakaa millä tahansa HTTP-palvelimella. Tässä luvussa pyritään selvittämään sitä, miten nämä ominaisuudet saavutetaan. Lisäksi käsitellään tapoja, joilla videota toimitetaan asiakkaalle ja miten asiakkaan ohjelmistot ja mediasoittimet saavat palvelimelta haettua video- ja audiodataa.

4.1 Median esitystapa

DASHin tavoitteena on toteuttaa median suoratoisto niin, että koko tapahtuman kontrollointi on asiakkaan vastuulla [10] [5]. Näin sisällön toimitus voidaan tehdä yksinkertaiseksi ja joustavaksi palvelimen päässä. Tavallinen työnkulku on seuraava: asiakkaalla (esim. mediasoitin ja web-sivu selaimessa) on tiedossa mitä esitystä ollaan aikeissa katsoa. Asiakas pyytää palvelimelta manifestia; manifesti on tiedosto, soittolista, joka sisältää tarvittavat tiedot median hakemiseen ja toistamiseen. Manifestista asiakas saa selville URL:n, josta mediaa voi hakea, sekä esitystapojen tiedot, kuten lähdemateriaalin nimi, tiedostoformaatit ja aikatiedot. Näiden tietojen perusteella soitin voi hakea video- ja audiotiedostoja palvelimelta ja toistaa ne.

Manifestia kutsutaan DASHin yhteydessä nimellä Media Presentation Description (MPD) [18]. MPD:ssä on tavallisesti listattuna vähintään video- ja audioraidat, niiden eri laadut tai mahdolliset kielivaihtoehdot ja URL:t mistä asiakas voi materiaalia ladata. Liite A sisältää esimerkkitiedoston, joka kuvaa videota, jossa on ranskankielinen ja englanninkielinen audioraita, saksankielinen tekstitysraita yksi videoraita. Videosta ja audiosta on useampi laatu tarjolla.

MPD on XML-muotoinen rakenteinen dokumentti, jonka tärkeimmät loogiset osat ovat periodi, esitys ja segmentti [18]. Tiedosto alkaa *MPD*-tagilla, jonka attribuutteina on oleellisia perustietoja protokollan toteutuksesta kyseisessä käyttötapauksessa. Periodit ovat loogisia kokonaisuuksia esityksestä. Ensimmäinen periodi voi sisältää esimerkiksi katsottavan ohjelman ensimmäisen osan, toinen periodi mainoksen tai mainoskatkon sisällön ja kolmas taas loput katsotusta ohjelmasta. Periodin sisällä on yksi tai useampi *AdaptionSet*-lohko. *AdaptionSet*tejä voi olla esimerkiksi ääniraita, ääniraita toisella kielellä, videoraita, tai eri kieliset tekstitysraidat. *AdaptionSet* koostuu erilaisista kyseisen raidan esityksistä, merkittynä *Representation*-tagilla. Eri esitykset ovat sisällöltään samanlaiset, mutta ne ovat enkoodattu eri resoluutioilla tai bittinopeuksilla.

4.1.1 Segmentointi, aikaleimat ja ABR

Yksi olennainen osa DASH-manifestia on streamin aikatiedot, eli segmentit. Koska tiedon siirto tapahtuu HTTP:n yli ja yhden paketin hyötykuorman koko on rajallinen, pitää lähdetiedostosta palauttaa asiakkaalle sopivan pieni osa. Materiaali voidaan transkoodata etukäteen sopivan mittaisiksi segmenteiksi, mutta DASHissa on mahdollista myös pyytää kokonaisesta lähdetiedostosta vain osa HTTP byte

range -pyynnöllä. MPD:stä käy ilmi esityksen pituus ja alkamisajankohta, sekä kunkin video- tai audiosegmentin pituus ja lukumäärä. Näin asiakas voi laskea, missä kohtaa esitystä toisto on sillä hetkellä ja mitä segmenttejä tulee pyytää palvelimelta seuraavaksi. Esityksen ajallisen kohdan lisäksi voidaan pyytää tiettyä osaa lähdemateriaalin tavujärjestyksen (eng. byte-range) perusteella. Aikaleimoihin perustuvien pyyntöjen käsittelyyn vaaditaan palvelimelta logiikkaa, joka osaa hakea ajan perusteella sopivat tavut segmentoimattomasta lähdetiedostosta. Liitteen A manifestin perusteella voitaisiin pyytää 640x480 pikselin resoluutioisesta videosta ensimmäiset 64 kilotavua seuraavanlaisella kyselyllä:

```
http://cdn1.example.com/41325645.mp4?range=0-64000
```

Olennaista tässä esimerkissä on URL:n muodostaminen MPD:n informaatiosta. XML-tiedoston MPD-lohkossa on listattu kaksi kappaletta *<BaseURL>* -tageja, joista asiakas valitsee haluamallaan logiikalla toisen. Sen jälkeen etsitään tiedostosta haluttu esitys, eli tässä tapauksessa mainittu video (*AdaptionSet mimeType="video/mp4"*), siitä haluttu resoluutio (esityksen WIDTH- ja *height*-arvot) ja poimitaan *<Representation>*-lohkosta jälleen *<BaseURL>*-lohkon arvo. Nämä tiedot yhdistämällä saadaan yksilöityä yksi tietty esitys halutusta mediasta. HTTP-pyynnön GET-metodin parametrina annetaan haluttu osa videosta. Asiakkaalla on jälleen vastuu siitä, että palvelimelta haetaan tarpeeksi nopeasti dataa ja oikeassa järjestyksessä, jotta median esittäminen on mahdollista. Manifestissa voidaan myös eksplisiittisesti kertoa esityksen kulku suhteessa aikaan:

```
<SegmentTemplate
  timescale="50"
  initialization="863536-$RepresentationID$.dash"
  media="863536-$RepresentationID$-$Time$.dash">
  <SegmentTimeline>
    <S t="0" d="100" r="2282" />
  </SegmentTimeline>
</SegmentTemplate>
```

Ohjelma 4.1 *DASH-manifestin segment template.*

Ohjelman 4.1 *SegmentTimeLine* kertoo, että esitys alkaa nollasta (t=0), videosegmentin pituus on 100 aikayksikköä ja tällainen segmentti toistuu striimissä 2282 kertaa. Tässä esimerkissä *863536* on katsotun esityksen nimi. Attribuutti *media* kertoo formaatin, miten parametrit välitetään palvelimelle HTTP-kutsussa *BaseURL*-kenttien perusteella koostetun URL:n jälkeen. Edellisen esimerkin esitystä voitaisiin siis hakea parametreilla *562465736.mp4-8-3400.dash*. Kyseessä olisi siis esityksen

34. videosegmentti (3400 jaettuna sadalla = 34). *initialization* segmentti palauttaa asiakkaalle metadataa, jota vaaditaan jotta mediasoitin pystyy toistamaan videosegmenttien sisältöä.

Edellä kuvatuilla mekanismeilla voidaan toteuttaa ABR-toiminnallisuus saumattomasti katkaisematta toistoa. Mikäli asiakassovellukseen rakennettu logiikka huomaa verkkoyhteyden tai laitteen suorituskyvyn heikentyneen niin paljon että toistoa ei voida jatkaa sen hetkisellä laadulla, pyydetään palvelimelta seuraavaa videosegmenttiä heikommalla laadulla [18]. Asiakkaalla on tiedossa joko ajankohta tai byte-range, jossa ollaan menossa sillä hetkellä, seuraavaan pyyntöön vain vaihdetaan parametri $RepresentatitionID$. Esitettävästä videosta on palvelimella valmiiksi transkoodattuna eri laatuisia versioita [21]. Edellä käsitellyssä esimerkissä voitaisiin pyytää MPD:ssä listattua seuraavaksi heikointa esitystä ohjelman 4.2 kuvauksen mukaan.

```
<Representation id="7" bandwidth="512000" width="320" height="240">
  <BaseURL>56363634.mp4</BaseURL>
</Representation>
```

Ohjelma 4.2 *Yksi MPD:n esitys. XML kuvaa esityksen kaistanleveyden, kuvan koon ja järjestyksen manifestissa.*

Esitys siis sisältää saman videon mutta pienemmällä resoluutiolla ja bittinopeudella. Koska pienempi kuva kuluttaa vähemmän kaistaa, on toistolla paremmat edellytykset jatkua katkotta. Kun verkko-olosuhteet taas paranevat, voidaan siirtyä samalla menetelmällä takaisin parempiin laatuihin.

4.1.2 Digitaalinen käyttöoikeuksien hallinta

DASH on DRM-agnostinen teknologia, mikä tarkoittaa sitä, että palvelun- tai sisällöntarjoaja voi käyttää mitä tahansa menetelmää materiaalin salaukseen. Salausmekanismi tulee kertoa manifestissa *ContentProtection*-lohkossa [21]. Mikäli kyseisiä lohkoja ei MPD:stä löydy, ei sisältö standardin mukaan saa olla salattua. Eri esitykset mediasta voivat olla eri tavoin salattu, mutta manifestin tulee kuitenkin tarjota tarvittavat tiedot, jotta asiakas voi salauksen purkaa (olettaen että tällä on siihen tarvittavat oikeudet).

```
<Period>
<!-- Audio protected with a specified license -->
<AdaptationSet
```

```
        mimeType="audio/mp4"
        codecs="mp4a.40"
        lang="en"
        subsegmentStartsWithSAP="1"
        subsegmentAlignment="true">
  <ContentProtection schemeIdUri="http://example.net/052011/drm">
    <drm:License>
      http://MoviesSP.example.com/protect?license=kljklsdfiowek
    </drm:License>
    <drm:Content>
      http://MoviesSP.example.com/protect?content=oyfYvpo8yFyvyo8f
    </drm:Content>
  </ContentProtection>
  <Representation id="1" bandwidth="64000">
    <BaseURL>audio/en/64.mp4</BaseURL>
  </Representation>
</AdaptationSet>
<!-- Audio protected with embedded information defined by 'ZZZZ' -->
<AdaptationSet
        mimeType="audio/mp4"
        codecs="mp4a.40"
        lang="fr"
        subsegmentStartsWithSAP="1"
        subsegmentAlignment="true">
 <ContentProtection schemeIdUri="urn:mpeg:dash:mp4protection:2011"
                    value="ZZZZ"/>
   <Representation id="3" bandwidth="64000">
     <BaseURL>audio/fr/64.mp4</BaseURL>
   </Representation>
</AdaptationSet>
```

Ohjelma 4.3 *Ote MPD:n periodista, jossa on listattu audioraidat.*

Ohjelman 4.3 esityksessä [10] audioraita on suojattu kahdella eri menetelmällä. Yritys nimeltä *MoviesSP* on suojannut ja lisensoinut englanninkielisen ääniraidan. *SchemeIdUri*-ominaisuuden osoittamasta URL:stä haetaan säännöt salauksen purkamisella. Säännöissä on mainittu, että MPD:ssä tulee olla kaksi URL:ää, joiden avulla voidaan hakea tarvittavat avaimet sisällön purkamiseksi. Ranskankielisen ääniraidan salaus on toteutettu menetelmällä *zzzz*, jonka jokin organisaatio on rekisteröinyt ja dokumentoinut MP4REG-järjestelmään. Kaikki tarvittavat tiedot tämän salauksen purkamiseksi voidaan hakea sieltä.

4.1.3 Live

Edellä on kuvattu videostreamin toimitusta ja esitystapaa tilanteissa, joissa videotiedostot on valmiiksi transkoodattuja, kokonaisia ja niistä on kaikki tarvittavat tiedot jo olemassa. Tällaista esitystä kutsutaan termillä Video On-Demand, eli VOD. Tilanne kuitenkin muuttuu kun streamataan reaaliaikaista kuvaa, kuten urheilutapahtumaa. Live-tapahtumasta ei tietenkään ole saatavilla kuin se osa esitystä, joka on jo oikeassa elämässä tapahtunut. Täten siis videon pituus kasvaa tapahtuman edetessä. Tapahtumasta voidaan live-esityksen jälkeen julkaista tallenne VOD-esityksenä. Liven tapauksessa video pitää myös lennosta segmentoida oikean mittaisiksi pätkiksi ja oikeaan formaattiin, jotta se on saatavilla asiakkaille mahdollisimman nopeasti. Käynnissä olevasta streamista ei voida tehdä byte-range -hakua, koska tiedoston koko kasvaa, ja tiedoston lukeminen ja kirjoittaminen samaan aikaan voi aiheuttaa ongelmia tiedon eheydessä. Videon enkoodaus ja segementointi aiheuttavat striimiin viivettä. Viivettä aiheuttaa myös HTTP-streamauksen luonne, jossa asiakkaat hakevat streamia palvelimelta (niin sanottu ”pull”-menetelmä), verrattuna esimerkiksi perinteiseen televisiolähetykseen, jossa uutta materiaalia syötetään asiakkaalle jatkuvasti (”push”-menetelmä).

Koska sisältö päivittyy ja esityksen pituus kasvaa jatkuvasti, täytyy MPD:tä päivittää ja asiakkaan tulee hakea päivitetty MPD säännöllisesti. Live-esityksen kannalta olennaista tietoa MPD:ssä on kentät *availabilityStartTime*, *availabilityEndTime*, sekä *minimumUpdatePeriod*. Nämä on kuvattu ohjelmassa 4.4. Ensimmäinen kertoo UTC-ajassa milloin striimi alkaa ja toinen kenttä milloin se loppuu. *minimumUpdatePeriod* ilmaisee intervallin, jonka jälkeen asiakkaan tulee päivittää MPD:nsä. *minimumUpdatePeriod* voi olla minimissään yhden segmentin kesto, muutoin asiakas saattaisi hakea saman version soittolistasta ennen kuin se on ehtinyt päivittyä. Soittolista haetaan aina samasta URL:stä.

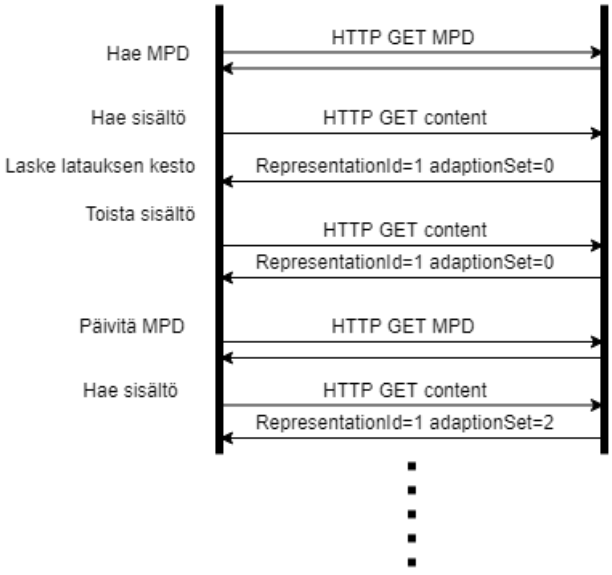
```
<MPD
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:mpeg:dash:schema:mpd:2011"
xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011
http://standards.iso.org/ittf/PubliclyAvailableStandards/\
MPEG-DASH_schema_files/DASH-MPD.xsd"
xmlns:cenc="urn:mpeg:cenc:2013"
xmlns:mas="urn:marlin:mas:1-0:services:schemas:mpd"
type="dynamic"
availabilityStartTime="2018-02-25T14:00:07Z"
publishTime="2018-02-25T16:52:05.313227Z"
minimumUpdatePeriod="PT2S"
timeShiftBufferDepth="PT2H51M32S"
```

```
maxSegmentDuration="PT3S"
minBufferTime="PT10S">
```

Ohjelma 4.4 *Perustietoja esityksestä.*

Asiakkaan vastuulle jää päätellä, missä kohtaa streamia on nykyhetki. Sen pystyy laskemaan, kun tiedossa on alkamisaika ja loppumisaika (esityksen kesto), sekä keskimääräinen segmentin pituus. DASHin standardin mukaan MPD:ssä voidaan listata myös ne segmentit, jotka eivät ole vielä saatavilla; joko seuraavaan MPD:n päivitykseen asti tai koko esityksen loppuun asti. Tästä kertoo MPD:n attribuutti *type*, jonka arvo ylläolevassa esimerkissä on ”dynamic”. *timeShiftBufferDepth* määritellään kun *type* on ”dynamic”, ja sen arvo kertoo ajanjakson jolta sisältöä on saatavissa. Asiakas voi siis toisin sanoen ”kelata” taaksepäin, tai aloittaa katselun esityksen alusta. Liven aikaikkuna voi olla yhtä pitkä esityksen keston kanssa, tai se voi jostain syystä olla lyhyempi. Ikkunaa voi olla syytä lyhentää esimerkiksi todella pitkissä lähetyksissä, jolloin tallenteen kesto on pitkä ja videosegmentit alkavat viedä huomattavasti tilaa palvelimella.

Asiakas siis hakee serveriltä manifestin sekä saatavilla olevat segmentit palvelimelta, toistaa median, päivittää manifestin ja hakee taas uudet saatavilla olevat mediasegmentit [18]. Tätä toistetaan esityksen loppuun.



Kuva 4.1 *Sekvenssikaavio live-esityksen dataliikenteestä asiakkaan ja palvelimen välillä.*

Kuva 4.1 kuvaa edellä esiteltyä tapahtumaketjua. Jotta toisto saataisiin aloitettua mahdollisimman nopeasti, ladataan ensimmäiset segmentit heikolla laadulla, jolloin

sisällön hakemiseen menee mahdollisimman vähän aikaa. Mikäli ympäristön olosuhteen sen sallivat, voidaan seuraavat segmentit hakea paremmalla laadulla. Satunnaisesti esityksen keskeltä valittu versio MPD:stä voisi segmentoinnin osalta näyttää seuraavalta:

```
<SegmentTimeline>
   <S t="0" d="96256" r="5132" />
</SegmentTimeline>
```

Ohjelma 4.5 *Manifestin SegmentTimeline ajanhetkellä t.*

Hetkeä myöhemmin päivitetyssä manifestissa *SegmentTimeline* on päivittynyt:

```
<SegmentTimeline>
   <S t="0" d="96256" r="5136" />
</SegmentTimeline>
```

Ohjelma 4.6 *Manifestin SegmentTimeline ajanhetkellä t+1.*

Esitys alkaa edelleen nollasta (*t="0"*) ja jokaisen segmentin kesto on sama (*d="96256"*), mutta segmenttien määrä on päivittynyt (attribuutti *r*). Ensin segmenttejä oli 5132, sitten 5136 kappaletta. Alkutilanne on kuvattu ohjelmassa 4.5 ja lopputilanne ohjelmassa 4.6.

4.1.4 Mainosten injektointi

Mediayhtiöille mainokset ovat äärimmäisen tärkeä osa liiketoimintaa. Mainosten avulla tv-kanavat ja sisällöntuottajat voivat rahoittaa liiketoimintaansa, etenkin jos ne tuottavat merkittävän määrän ilmaista sisältöä loppukäyttäjille. Mainosten avulla siis voidaan tarjota ihmisille viihdettä huokeampaan hintaan tai jopa ilmaiseksi. Mainoksia näytetään myös maksullisen sisällön kanssa, ja etenkin paljon katsojia vetävät tapahtumat tuottavat huomattavia summia rahaa. Mainoskatkot ovatkin tuttuja jo perinteisistä televisiolähetyksistä. Tällöin signaalin lähettäjä vaihtaa esimerkiksi urheilutapahtuman kameroiden lähettämän signaalin mainoksiin ja näin erilainen sisältö lomitetaan lähetykseen. Samaa menetelmää voidaan käyttää myös videon streamauksessa: lähetystä tekevän enkooderin sisääntuloon tuleva signaali vaihdetaan toiseen ajoittain. Tietokoneet, Internet ja asiakas-palvelin -tyyppinen malli kuitenkin mahdollistaa tehokkaamman mainonnan: asiakas ja asiakkaan päätelaite ja sijainti voidaan tunnistaa yksilöllisten HTTP-pyyntöjen, IP-osoitteen tai sovelluksen lähettämän datan perusteella, jolloin käyttäjälle voidaan näyttää kohdennettua mainontaa hänen ympäristönsä ja oletettujen mielenkiinnonkohteiden mukaan.

DASHin standardi ja sen mukainen MPD:n rakenne mahdollistavat tehokkaan mainosten injektoinnin niin palvelimen kuin asiakkaankin päässä.

Mainosten injektoinnissa pyritään näyttämään mainokset, eli varsinaisesta käyttäjän hakemasta sisällöstä poikkeava sisältö, saumattomasti osana alkuperäistä esitystä. Tähän päästään MPD:tä manipuloimalla. Hierarkkinen XML-muotoinen MPD koostuu korkealla tasolla periodeista, jotka merkitään *<period>* -tageilla. Mainos voidaan esittää esimerkiksi niin, että ensimmäinen periodi sisältää tiedot siitä sisällöstä, jota käyttäjä hakee palvelusta, toinen periodi sisältää mainoskatkon tiedot ja kolmas taas jatkaa varsinaista sisältöä. Jokaisella periodilla on alku- ja loppuaika (tai vaihtoehtoisesti kesto) ilmoitettuna. Jokainen periodi on itsenäinen kokonaisuus, jolloin mainoksella voi olla muusta sisällöstä poikkeava URL mistä mainos ladataan. Tällöin siis mainos voidaan ladata erilliseltä mainospalvelimelta tai suoraan mainostoimiston palvelimelta. Mainoksen sisältävä manifesti voisi olla ohjelman 4.7 kaltainen.

```
<period id="1" start="PT0.00S" duration="PT600.6S">
  <AdaptionSet>
    <BaseUrl>www.videoservice.com</BaseUrl>
        ...
  </AdaptionSet>
</period>
<period id="2" duration="PT5.00S">
  <AdaptionSet>
    <BaseUrl>www.adservice.com</BaseUrl>
        ...
  </AdaptionSet>
</Period>
<period id="3" duration="PT550.0S">
 <AdaptionSet>
    <BaseUrl>www.videoservice.com</BaseUrl>
        ...
  </AdaptionSet>
</period>
```

Ohjelma 4.7 *Useasta periodista koostuva manifesti, joka sisältää mainoksen.*

Apuna mainosten injektoinnissa ja personoinnissa voidaan käyttää erillistä mainospalvelinta. Kun asiakas pyytää palvelimelta MPD:tä, lähetetään asiakkaan tiedot mainospalvelimelle, joka generoi sopivan periodin liitettäväksi lopulliseen manifestiin. MPD serverin kuorman helpottamiseksi voidaan myös käyttää erillistä XML resolveria ja XML Linking Language (XLink) elementtejä MPD:ssä. XLink-elementit on esitelty ohjelmassa 4.8. Tällöin mainoksen sisältävässä periodissa on kaksi attri-

buuttia: *xlink:href*, joka sisältää URL:n ja parametrit mainoksen hakemiseksi, sekä *xlink:actuate* -parametrin, joka kertoo milloin sisältö haetaan.

```
<Period
xlink:href="http://adserver.com/period_timescapes_komp?country=fi&zip=33200"
xlink:actuate="onLoad">
</Period>
```

Ohjelma 4.8 *XLink-elementillä kerrotaan asiakkaalle mistä mainos ladataan.*

Kun mainoksen aika koittaa, tehdään *xlink:href* pyyntö xml-resolverille, joka on yhteydessä mainospalvelimelle ja hakee tarvittavat tiedot. Tämän jälkeen resolveri muodostaa sopivan MPD:n ja palauttaa sen asiakkaalle. Tuki XLinkille pitää löytyä playeristä, ja sellainen löytyy DASH-IF:n tarjoamasta referenssiplayeristä.

Live-lähetyksessä ei välttämättä voida yksiselitteisesti määrittää kohtaa, jossa mainoskatko pidetään. Esimerkiksi jääkiekko-ottelussa tauot määräytyvät ottelun tapahtumien mukaan, eikä mainoksia haluta näyttää kesken pelitilanteen. Mainoskatko voidaan signaloida asiakkaalle käyttämällä SCTE 35 -standardin mukaisia viestejä. Tällainen viesti tulkitaan asiakkaan toimesta merkiksi siitä, että nykyinen MPD ei ole enää validi (MPD Validity Expiration). Tämän aiheuttaa sen, että asiakas pyytää palvelimelta uutta MPD:tä, jolloin asiakkaalle palautetaan muunneltu manifesti, joka sisältää mainokset.

Mainosten hakeminen ja näyttäminen voidaan myös hoitaa täysin irrallaan varsinaisen palveluntarjoajan infrastruktuurista playerin tai sitä näyttävän verkkosivun tai applikaation toimesta. Tällöin logiikka on vapaasti sovelluskehittäjien toteutettavissa. Näillä menetelmillä ei kuitenkaan ole suoraa yhteyttä DASHin standardiin.

4.2 HTML5 ja player-toteutukset

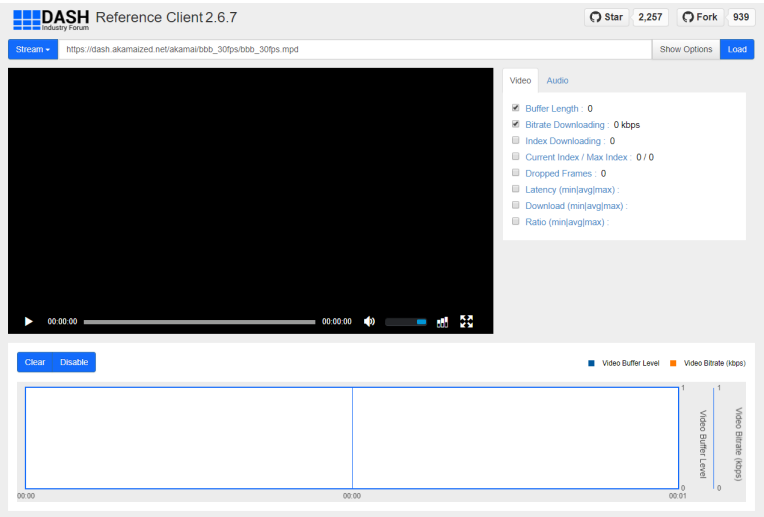
Alun perin HTML:n standardissa ei oltu huomioitu tarvetta videon ja äänen esittämiselle Webissä. Tilanne kuitenkin muuttui ajan saatossa kun kaistanleveydet kasvoivat ja tekniikat kehittyivät. Koska suoraa tukea videolle ei ollut, oli sisällön esittäminen riippuvaista kolmansien osapuolien sovelluksista ja selainten lisäosista. Tällaisia ovat muun muassa Adoben Flash ja Microsoftin Silverlight -sovelluskehykset. Tällaisten lisäosien ongelmia ovat esimerkiksi mahdolliset tietoturva-aukot ja suuri resurssien tarve päätelaitteelta. Web-kehittäjät eivät myöskään pysty vaikuttamaan näiden ohjelmien sisäiseen toimintaan ja ominaisuuksiin, jolloin oman Web-sivun kehittäminen oli haasteellisempaa ja kehittäessä tuli huomioida kolmannen osapuolen

sovelluskehyksen vaatimukset. Tarpeeseen vastattiin kun W3C julkaisi ehdotuksensa Media Source Extension (MSE) -tekniikasta. [23]

Suosituimmista selaimista Google Chrome otti MSE:n käyttöön 2013 [24] ja Mozilla 2015 [25]. MSE on tekniikka, joka sallii JavaScriptin luoda data streameja ja täten käyttää hyväksi laitteistoa ja käyttöjärjestelmää videon pakkauksen purkamiseksi. Kun JavaScript ja web-sivu osaavat nyt natiivisti hyödyntää selaimen rajapintoja, on video purkaminen tehokkaampaa. Koska kolmannen osapuolen sovelluksia ei välttämättä tarvita, voidaan soitin ladata osana sivua ja se on vapaasti sivun kehittäjien muokattavissa.

Koska DASH on alusta alkaen pyrkinyt olemaan alustariippumaton avoin standardi, on edellä mainittu muutos mahdollistanut DASHin käytön ja leviämisen laajoille yleisöille. DASH-IF on julkaissut myös oman referenssitoteutuksensa mediasoittimesta nimellä dash.js. GitHub-palvelussaan DASH-IF kirjoittaa (englanniksi): "Dash.js on DASH Industry Forumin aloite pystyttää tuotantovalmis kehys video- ja audiosoittimien rakentamiseksi MPEG-DASH -sisällön toistamiseen. Tässä käytetään JavaScript -kirjastoja asiakkaan päätelaitteessa sekä hyödynnetään W3C:n määrittelemää MediaSource Extensions API:a". Sivulla mainitaan projektin kontribuoijiksi myös Microsoft, Akamai, Ericsson, Qualcomm, Adobe, Sony ja Cisco. Lähdekoodi on julkaistu BSD-3 -lisenssin alla, mikä mahdollistaa koodin ja ohjelman binäärien käytön niin yksityisessä kuin kaupallisessa käytössä. Näin DASH-IF haluaa promotoida standardiaan ja helpottaa tekniikan käyttöönottoa.[26]

DASH-IF ylläpitää myös palvelua, jossa on dash.js-pohjainen player jota voi käyttää omien striimien testaamiseen [27]. Tarjolla on myös eri versioita soittimesta testausta varten.



Kuva 4.2 *Näkymä dash.js referenssiplayeristä*

Työkalulle voi syöttää oman MPD:n URL:n, jolloin player pyrkii lataamaan ja toistamaan käyttäjän tarjoamaa sisältöä. Tarjolla on myös laaja valikoima erilaista testimateriaalia, kuten DASH-IF:n julkisesti tarjoamat testivektorit [28]. Työkalulla voi tarkkailla striimin latausnopeuksia, latensseja, sekä muita toistoon liittyviä statistiikkoja. Soittimeen voi myös antaa joitain parametreja, kuten haluttu DRM-tekniikka ja ABR:n laatu. Ennen kaikkea testisoitin on omiaan testaamaan JavaScript-kirjastoihin tehtyjä muutoksia ja niiden yhteensopivuutta oman infrastruktuurin kanssa. Kuva 4.2 sisältää kuvankaappauksen palvelusta ja joistakin sen valinnaisista ominaisuuksista.

5. VERTAILUA MUIHIN STREAMAUSTEKNOLOGIOIHIN

Adaptiivinen striimaus HTTP-protokollaa hyödyntäen ei itsessään ole uusi keksintö. Kuten aiemmin tässä opinnäytetyössä on mainittu, suosituimpia teknologioita ovat Microsoft SmoothStreaming, Adobe HTTP Dynamic Streaming ja Applen HTTP Live Streaming. Tässä kappaleessa pyritään esittämään yhtäläisyydet ja erot edellä mainittujen teknologioiden ja DASHin välillä. Vertailussa on mukana ainoastaan nämä HTTP-teknologiat. Kuten kappaleessa kaksi on esitetty, UDP eroaa toteutustavallaan huomattavasti HTTP-streamauksesta, eikä näiden kahden protokollan vertailu tässä kontekstissa ole mielekästä. Molemmilla on kuitenkin omat heikkoutensa ja vahvuutensa.

5.1 Perusominaisuudet

Kaikista vertailun tekniikoista tukevat seuraavia toiston ominaisuuksia: usean eri laadun toistaminen ja adaptiivisuus (multi bitrate), usean eri ääniraidan tarjoaminen, tekstitykset (usealla eri kielellä), videon haku ja kelaus (lisäksi ns. ”trick play”) ja live-sisällön streamaus. Näiden perusominaisuuksien valossa kuluttajalle ei lopulta ole merkitystä palveluntarjoajan käyttämällä tekniikalla, ominaisuudet ovat jokseenkin tuttuja jo DVD-tallenteista. Englanninkielinen termi ”trick play” tarkoittaa sellaista ominaisuutta, missä sisältö voidaan toistaa hitaammin tai nopeammin kuin mitä esitys on tarkoitettu, esimerkiksi kelatessa videota. Osa ominaisuuksista on lisätty standardeihin pikku hiljaa vuosien varrella teknologioiden kehittyessä.

Nämä ominaisuudet on rakennettu ainoastaan DASHin standardiin: usea vaihtoehtoinen CDN, natiivi mainosten injektointi ja koodekkiagnostisuus. Useamman CDN-noodin, eli käytännössä videolähteen, tarjoaminen parantaa toiston luotettavuutta ja saatavuutta. Jos ensisijainen CDN on jostain syystä saavuttamattomissa, asiakas voi valita jonkin toisen CDN:n URL:n manifestista. Vastaavan kaltaisia ominaisuuksia voidaan myös toteuttaa muissa teknologioissa, mutta DASHissa se kuuluu standardiin ja vaihtoehtoiset lähteet esitetään suoraan manifestissa. Sama pätee myös

mainosten injektointiin; siihen on kehitetty keinoja, jotka usein nojaavat videosegmenttien korvaamiseen tai manifestin manipulointiin. DASH kuitenkin tarjoaa natiivin ratkaisun esittelemällä manifestissa useampia periodeja ja mahdollistamalla erillisten mainospalvelinten käytön. Mainosten injektointia käsiteltiin luvussa 3.1.4. DASH ei ota kantaa siihen, millä koodekilla videomateriaali on enkoodattu. Applen HLS tukee ainoastaan H.264 enkoodausta, Microsoftin SmoothStreaming H.264 sekä VC-1 enkoodausta (VC-1 on Microsoftin kehittämä) ja Adoben HDS vaatii VP-6 tai muiden tapaan H.264 enkoodausta. Vaikka DASH onkin koodekkiagnostinen tekniikka, DASH-IF suosittelee silti käyttämään H.264 enkoodausta, jotta saavutettaisiin mahdollisimman laaja yhteensopivuus laitteiden välillä.

Kaikki vertailun teknologiat vaativat videon pakkaamista ISO Base Media File Format -muotoon. Tämä tunnetaan yleisemmin nimellä MP4. Apple tosin lisäsin tämän HLS:ään vasta vuonna 2016. Tämä oli helpotus alan toimijoille, sillä aiemmin HLS-sisältö piti olla pakattuna MPEG2-TS -formaattiin. Muutoksen jälkeen samaa valmiiksi fragmentoitua sisältöä voidaan jaella myös DASHia tukeville laitteille ilman uutta enkoodausta [29]. Adoben HDS tukee videon resoluutiota ainoastaan 1920x720 pikseliin asti (720p)[30], kun muut formaatit tukevat jopa 4k-tarkkuutta [30][31].

DASHin käyttäminen on kaikille ilmaista. Se toimii ilmaisten Web-palvelimien, kuten Apache tai NGinX, avulla. Referenssisoitin ja JavaScript-kirjasto (dash.js) ovat myös vapaasti saatavilla. Myös Microsoftin IIS-ohjelmisto tulee ilmaiseksi Windows-käyttöjärjestelmän mukana ja siihen saa ladattua SmoothStreaming-lisäosan ilmaiseksi. Windows itsessään ei tosin ole ilmainen, vaan lisenssi pitää hankkia. Myös SmoothStreamingin toistamiseen web-sivulla vaadittava Silverlight-ohjelmisto on ilmaiseksi käytettävissä. HLS videota voi myös toimittaa asiakkaille tavallisilla web-palvelimilla. Video pitää kuitenkin valmiiksi segmentoida ennen jakelua, mihin tarvitaan erillinen ohjelmisto. Applen mukaan tähän voi käyttää kolmannen osapuolen työkaluja, mutta he tarjoavat myös omaa Media Segmenter -työkalua. Työkalu on näennäisesti ilmainen, se on saatavilla rekisteröidyille Apple-kehittäjille. Tuon statuksen saa rekisteröitymällä ja maksamalla 100 USA:n dollaria vuodessa [32]. Adobe erottuu joukosta asettamalla kunnon hinnan tuotteelleen. Striimaamiseen tarvittava Adobe Media Server maksaa perusversiona 995 dollaria. Standard-versio on kuitenkin ominaisuuksiltaan sen verran karsittu, että useimmat isot yritykset joutunevat hankkimaan Pro-version 4500 dollarilla [33]. Adobe ei web-sivuillaan määrittele onko hinta kertaluontoinen korvaus, vai saako tällä lisenssin käyttöönsä vain tietyksi aikaa.

Adobe on ilmoittanut lopettavansa Flashin kehittämisen ja jakelun vuoteen 2020 mennessä [34]. Tämä tarkoittaa todennäköisesti HDS:n loppua nykymuodossaan.

Myös serverin pään ohjelmiston Media Serverin tuki loppuu vuonna 2018 [35].

5.2 Digitaalisten käyttöoikeuksien hallinnan tekniikat

Vertailun tekniikoista kolme käyttää omaa DRM-tekniikkaansa sisällön suojaamiseen ja avainten hallintaan. SmoothStreaming tukeutuu Playreadyyn, HLS FairPlayhyn ja HDS Adobe Access -tekniikkaan. DASH ei ota kantaa tapaan, jolla avaimia hallitaan tai miten niitä jaetaan asiakkaille. DASH-asiakasohjelmille viestitään käytetty DRM-tekniikka manifestissa. Tästä on kirjoitettu tarkemmin luvussa 3.1.2.

Applen FairPlay on tuettuna ainoastaan Applen omissa tuotteissa (OS X, iOS, Applte TV, AirPlay). Sisällön haltija hallitsee myös avaimia omalla avainpalvelimellaan. Avainpalvelimelle tulee asentaa FairPlay-moduuli, vastaavasti asiakassovellukseen pitää asentaa oma moduulinsa. FairPlayn koodi on saatavilla kehittäjille ja on muokattavissa sovelluksen tarpeisiin. Käyttö sinänsä ei maksa mitään, rekisteröidyiltä kehittäjiltä vaaditaan 100 USA:n dollarin vuotuinen maksu [32].

Adoben Primetime -niminen tuote käyttää Access-teknologiaa ja integroituu myöskin Adoben oman tuoteperheen kanssa. Lisenssipalvelimelle asennetaan ohjelmisto, jonka kanssa asiakkaat palveluntarjoajan muut järjestelmät keskustelevat. Adobe tarjoa SDK:n, jonka avulla asiakkaan päässä voidaan sisällön salaus purkaa saadulla avaimella. Järjestelmä vaatii toimiakseen Flash playerin tai AIR-ajoympäristön asiakaslaitteelta. Adoben web-sivuilla ei anneta suoraan hintaa tuotteelle, vaan asiakasta kehotetaan ottamaan yhteyttä ja neuvottelemaan tarpeistaan Adoben myyjien kanssa. [36] [37]

Microsoftin Playready noudattaa samaa kaavaa. Lisenssipalvelimelle pitää asentaa Microsoftin ohjelmisto, joka lisäksi toimii ainoastaan Windows-käyttöjärjestelmillä. Asiakkaan puolelle pitää joko kirjoittaa ohjelmisto Playready SDK:lla tai käyttää Silverlight-ohjelmistoa, joka valmiiksi tukee Playready-salauksen purkua. Lisenssipalvelimen ohjelmisto ei maksa mitään, mutta palveluntrajoajan pitää hakea lisenssiä Microsoftilta. [38]

5.3 Vaadittavat ohjelmistot ja infrastruktuuri

Kun sisältö on sopivasti pakattu ja mahdollisesti pilkottu valmiiksi, riippumatta käytetystä tekniikasta sisältöä voidaan jaella tavallisilla web-palvelimilla. Koska liikenne palvelimen ja asiakkaan välillä on toteutettu HTTP-protokollalla, voidaan hyödyntää olemassa olevaa Web-infrastruktuuria. HTTP toimii kuitenkin vain kuljetusprotokollana, ennen toimitusta ja sen jälkeen tarvitaan kuitenkin erinäisiä ohjelmistoja,

jotta videon toimitus ja toisto onnistuu.

DASH ei tarvitse origin-palvelimelle periaatteessa mitään ylimääräistä tekniikkaa. Video enkoodataan MP4-formaattiin ja siitä voidaan pyytää osia byte range -tyyppisillä HTTP-kutsuilla. Monet enkoodaustuotteet osaavat myös valmistella manifestin. Halutessaan tähän voi käyttää jotain erillistä ohjelmistoa. Origin-palvelimelle voidaan kuitenkin lisätä älykkyyttä ja ominaisuuksia käyttämällä erinäisiä laajennoksia ja erillisiä ohjelmistoja videon jakelussa [39][40]. Asiakkaan puolelta vaaditaan HTML5:ttä ja MSE:tä tukeva selain. Vaihtoehtoisesti tarvitaan jokin muu sovellus (kuten mobiiliapplikaatio), joka kykenee toistamaan streamattua mediaa. DASH-IF on kehittänyt JavaScript-kirjaston Dash.js, joka on vapaasti käytettävissä.

SmoothStreaming tuotetaan Microsoftin IIS-palvelinohjelmistoon asennetulla Smooth-Streaming -lisäosalla. Lisäosa muun muassa fragmentoi videota sopiviksi paloiksi toimitusta varten, tarjoaa DVR-tallennusominaisuuksia ja metriikkadataa asiakkailta ja osaa vastaanottaa streamia enkooderilta tai ingest-palvelimelta jakaakseen sitä edelleen. Koska kyseessä on IIS:n lisäosa, voidaan näitä tuotteita käyttää ainoastaan Windows-käyttöjärjestelmää ajavilla palvelimilla. Asiakkaan selain vaatii Silverlight-lisäosan selaimeen. Silverlight asennetaan erikseen ja se osaa purkaa PlayReady-salauksen ja toistaa SmoothStreaming videota. Lisäksi se tarjoaa soittimeen ominaisuuksia, kuten pause ja kelaus. Microsoft tarjoaa sovelluskehittäjille myös Silverlight SDK:n, jolla voidaan rakentaa omia applikaatioita videon toistamiseksi.

HLS on natiivisti tuettu uusimmilla selaimilla, jotka hyödyntävät MSE- ja EME-laajennoksia [41][42]. Applen valmistamat laitteet tukevat muuten natiivisti HLS-formaattia. Myös Applen QuickTime-soitin pystyy toistamaan HLS-streamia. Palvelimen puolella ei itse streamaamiseen tarvita erityisiä ohjelmistoja, sisältöä voi jaella tavallisella web-palvelimella. Vuonna 2016 Apple ilmoitti, että HLS tukee nyt myös mp4-tiedostomuotoa (kts. kappale 4.1.). Tämän myöstä sisältööäkään ei tarvitse välttämättä valmistella erityisesti. Vanhemman käytännön mukaan sisällön pitää olla mpeg2-ts -muodossa ja valmiiksi palasteltu sopivan pituisiksi segmenteiksi. Apple tarjoaa kehittäjille työkalut streamin segmentointiin ja manifestien tekemiseen tutulla 100 dollarin vuosimaksulla [43]. Adoben mukaan myös heidän Media Server -tuotteensa kykenee myös tuottamaan HLS-streamia [44]. [32]

Kuten kappaleessa 4.1. jo mainittiin, Adoben HDS vaatii toimiakseen serveriltä ohjelmiston Adobe Media Server. Versiosta riippuen Media Server kykenee tuottamaan myös UDP multicast, HLS- ja RTMP streamia (RTMP on myös Adoben kehittämä protokolla Flash-soittimille, joka on tulossa tiensä päähän [45]). Asiakkaan puolella

vaaditaan puolestaan Flash Player HDS-sisällön toistamiseen. Adobe tarjoaa ilmaista Flash Playeria käyttäjille. Kyseessä on selaimeen asennettava laajennos. Vaihtoehtoisesti kehittäjille on käytössä Air-ajoympäristö omien sovelluksien luomiseen.

6. DASH KÄYTÄNNÖSSÄ

Tässä osiossa kuvaillaan todellista järjestelmää, jossa DASHia hyödyntämällä toimitetaan sisältöä asiakkaille. Keskiössä ovat omat kokemukseni työelämästä, sekä näkemäni ratkaisut ja järjestelmät. Työnantajani Cybercom Finlandin toimeksiannosta olen päässyt työskentelemään MTV Oy:lle ja saanut näköalapaikan viihdeteollisuuteen ja erityisesti videon striimaamiseen suomalaisille asiakkaille. Työnkuvaani on kuulunut koko julkaisuketjun suunnittelu, toteutus, vianselvitys ja automatisointi. Tämä kattaa lähdemateriaalin vastaanottamisen ja enkoodauksen/transkoodauksen, sisällön suojaamisen, varastoinnin ja jakelun. Olen työssäni myös automatisoinut ympäristöjä joista jaellaan sekä videota, että webbisivuja jotka lopulta toistavat sisältöä. Striimaustekniikat eivät rajoitu MTV:llä ainoastaan DASHiin, vaan käytössä on myös Microsoftin SmoothStreaming ja Applen HLS.

6.1 MTV Oy ja taustoja

MTV Oy on suomalainen mediayhtiö, joka on perustettu television alkuaikoina vuonna 1957 nimellä Oy Mainos-TV-Reklam Ab. Yhtiön nimi vaihtui vuonna 1982 ja vuonna 2007 yhtiön osti ruotsalainen Bonnies AB. MTV:n tarjontaan kuuluu muun muassa televisiokanavat MTV3, Sub ja AVA, sekä verkossa suoratoistopalvelut Katsomo ja CMore. [46] Bonnier ei julkaise tarkkoja talouslukuja omistamistaan yrityksistä, mutta vuoden 2016 vuosikatsauksessa mainitaan liiketoiminta-alueen käyttökatteeksi 373 miljoonaa Ruotsin kruunua, eli noin 36,2 miljoonaa euroa. Bonnier omistaa myös TV4:n Ruotsissa, CMore Entertainmentin joka toimii pohjoismaissa, sekä Nyhetsbolaget Sverige AB:n [47]. Kantar TNS -yhtiön tilastojen mukaan mtv.fi oli vuonna 2017 keskimäärin Suomen viidenneksi suosituin verkkosivusto, kävijöitä ollen 1,2-1,5 miljoonaa viikoittain [48].

MTV on julkaissut videota verkossa jo vuodesta 1995 [49], mutta nykyinen Katsomopalvelu lanseerattiin ensimmäisen kerran vuonna 2009. Katsomo on tarjonnut sekä ilmaista että maksullista sisältöä. Katsottavissa on muun muassa urheilua, uutisia, televisiossa näytettyjä ohjelmia, elokuvia ja sarjoja. Vuonna 2016 julkaistiin uusi suoratoistopalvelu CMore Katsomon rinnalle. Sisällöt jaettiin karkeasti kah-

teen osaan: maksullinen sisältö on katsottavissa CMoressa ja ilmainen sisältö Katsomossa. CMoren sisältö profiloituu niin sanotuksi "premium contentiksi", eli asiakkaille pyritään tarjoamaan suosittuja kotimaisia ja ulkomaisia elokuvia ja sarjoja, ensi-iltoja ja huippu-urheilua. Katsomon ja CMoren taustalla olevat tekniikat ja infrastruktuuri ovat oleellisilta osilta yhteisiä tai yhdenmukaisia, eikä työn aiheen kannalta ole oleellista jaotella näitä palveluita. Yleisesti ottaen emme halua sekoittaa näitä kahta palvelua toisiinsa, mutta tässä kontekstissa tekstin sujuvuuden kannalta valitsemme toisen näistä kahdesta. Tästä eteenpäin käytetään palvelusta nimeä Katsomo, riippumatta siitä kummasta palvelusta oikeasti on juuri siinä yhteydessä kysymys.

Ennen DASHin käyttöönottoa Katsomon videojakelu perustui suurimmaksi osaksi Microsoftin SmoothStreaming -teknologiaan ja niin ikää Microsoftin Silverlight -selainlaajennokseen. Selainlaajennokseen liittyvät hankaluudet ja uudet mahdollisuudet HTML5-teknologian tehdessä tuloaan kuitenkin pistivät miettimään vaihtoehtoja, joita oli valitettavan heikosti tarjolla aikanaan. DASHin ja HTML5:een perustuvia ratkaisuja alettiin suunnitella jo hyvissä ajoin, mutta vasta laaja selainvalmistajien tuki teki näiden käytön todella mahdolliseksi. Kuten aiemmin kappaleessa ”HTML5 ja player-toteutukset” on mainittu, esimerkiksi Firefox Mozilla lisäsi tuen vasta vuonna 2015. DASHin laajempi jakelu alkoi vuosien 2015 ja 2016 aikana. [49] SmoothStreaming on edelleen käytössä tietyille päätelaitteille. Se toimii myös väliformaattina, kun streameja tai videotallenteita siirretään jakeluketjun sisällä. Lisäksi käytössä on vielä kolmas formaatti, Applen HLS, sillä Apple vaatii sitä tietyille päätelaitteillensa (ja tarjoaa sille myös natiivin tuen). SmoothStreaming ja siihen liittyvät palvelimet ja ohjelmistot ovat historiassa osoittaneet luotettavuutensa ja laatunsa, eikä siitä sen vuoksi ole aktiivisesti pyritty pääsemään eroon väliformaattina.

DASH otettiin käyttöön Katsomossa jo ennen kuin itse aloitin aktiivisen työskentelyn palvelun parissa. Omissa työtehtävissäni olen kuitenkin selvittänyt toistoon ja sisällöntuotantoon liittyviä ongelmia, mikä on vaatinut DASHin ja Katsomon infrastruktuurin tuntemista. Koska väliformaattina käytetään SmoothStreamia, on sen toiminta syytä tuntea. Ajan kuluessa käytettyihin ohjelmistoihin ja itse palveluun on tullut päivityksiä, joita olen ollut mukana tekemässä ja testaamassa.

6.2 Yleiskuvaus infrastruktuurista

Katsomosta esitetään sekä video on-demand (VOD) -sisältöä, että livesisältöä (live). Näiden julkaisuketju poikkeaa toisistaan ja niitä on syytä käsitellä erikseen. Sisällön hankinta ja raakamateriaalin tuotanto on MTV:n muiden osien ja tiimien vastuulla,

eikä se kuulu tämän opinnäytetyön aihepiiriin. Samaten osittain aiheesta ohi menee raakamateriaalin transkoodaus ja enkoodaus tuotantovalmiiksi sisällöksi, mutta tätä asiaa sivutaan tarpeen mukaan.

OTT-tiimin (Over-The-Top, videon jakelu internetin kautta) vastuulle kuuluu videon enkoodaus ja transkoodaus, sisällön varastointi, salaus ja sisällönhallintajärjestelmän ylläpito, materiaalin jakelu origin-servereille sekä lopulta videon striimaus web-palvelimien kautta CDN-palveluntarjoajille. Tiimi myös konfiguroi CDN-verkkoa yhteistyössä palveluntarjoajan kanssa. Palvelun web-sivustosta ja mediasoittimesta on vastuussa oma tiiminsä, jonka kanssa tehdään kuitenkin tiivistä yhteistyötä. Enkoodauksesta ja transkoodauksesta OTT-tiimin kanssa on myös vastuussa ulkopuolinen palveluntarjoaja, joka toimii MTV:n tiloissa. Sisällönhallintajärjestelmä, sen yksityinen ja julkinen ohjelmointirajapinta (API), sekä tietokanta on myös ostettu kolmannelta osapuolelta palveluna ja on pääosin palvelun toimittajan ylläpidossa. Tämä API on olennaisessa osassa videon toiston kannalta.

Osa infrastruktuurista sijaitsee MTV:n tiloissa ja osa Amazonin pilvipalvelussa (AWS, Amazon Web Services). MTV:n tiloissa oleva laitteisto kattaa palvelun osalta palvelimet, levypalvelimet ja verkkoinfrastruktuurin. Ympäristö on osa MTV:n laajempaa infrastruktuuria. AWS tarjoaa kattavan valikoiman PaaS (Platform-as-a-Service) -tyyppisiä, sekä IaaS (Infrastructure-as-a-Service)-ratkaisuja, tietovarastoja, automatisointityökaluja, sekä esimerkiksi oman CDN-palvelun [50]. Loppukäyttäjät ja heidän päätelaitteensa eivät koskaan ole suoraan yhteydessä MTV:n palvelimiin, vaan kaikki sisältöä tarjoillaan CDN-palvelusta. Kuten luvussa 2.3 on todettu, tämä on olennaista palvelun suorituskyvyn ja käyttökokemuksen kannalta. Tällainen ratkaisu myös piilottaa origin-palvelimet CDN:n taakse, mikä parantaa tietoturvaa pienentämällä hyökkäyspinta-alaa. CDN hakee tarvitsemansa sisällön ja sivustot Katsomon palvelimilta. Katsomon CDN-palveluntarjoajana toimii Akamai Technologies. Akamain verkon kautta kulkee 15-30% maailman Internet-liikenteestä [51]. He toimittavat ratkaisuja median alalla muun muassa musiikkitelevisio MTV:lle, Yhdysvaltain koripalloliigalle NBA:lle sekä pelistudio Ubisoftille [6].

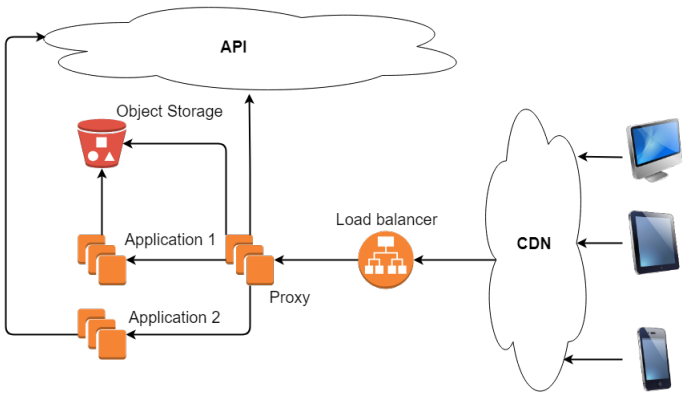
Katsomon origin-servereillä web-jakelusta vastaa Apache -palvelinohjelma, joka on saavuttanut maailmalla lähes standardinomaisen aseman web-palvelimena [52]. Apachen rinnalle on asennettu Unified Streaming -nimisen yhtiön tekemä lisäosa, josta käytetään yleisemmin nimitystä Unified Streaming Platfrom (USP) [39]. USP osaa käsitellä ja luoda manifesteja ja paketoida mediatiedostoja niin, että asiakas voi ne toistaa. Ohjelma toimii ”just-in-time”-periaatteella, tai dynaamisesti, eli manifestit luodaan samalla kun asiakkaat niitä pyytävät. Näin manifestia voidaan räätälöidä asiakaskohtaisesti. Lisäksi live-ohjelman manifesti muuttuu jatkuvasti, eikä näi-

tä voida tallentaa välimuistiin. Pohjana manifesteissa käytetään SmoothStreaming manifesteja. Videon formaatti on fragmentoitu mp4, joten HLS:ää varten formaatti muutetaan transport streamiksi.

6.2.1 Web front-end

Web-edustan tehtävänä on palvelun käyttöliittymän ja videokirjaston esittäminen loppukäyttäjälle, sekä itse videon toistaminen. Kuten aiemmin on mainittu, loppukäyttäjät ovat yhteydessä CDN-palveluun, joka jakaa staattista sisältöä välimuistista. Mikäli asiakkaan haluamaa sisältöä ei löydy välimuistista, haetaan se Katsomon palvelimilta. Ympäristö on rakennettu AWS:n pilvialustan toimittamien komponenttien päälle. Selaimessa näytettävä applikaatio koostuu useista eri osista ja toiminnallisuutta on hajautettu useille palvelimille. Staattista sisältöä, kuten staattiset html-dokumentit ja videosoittimen koodi, tarjoillaan suoraan tietovarastosta.

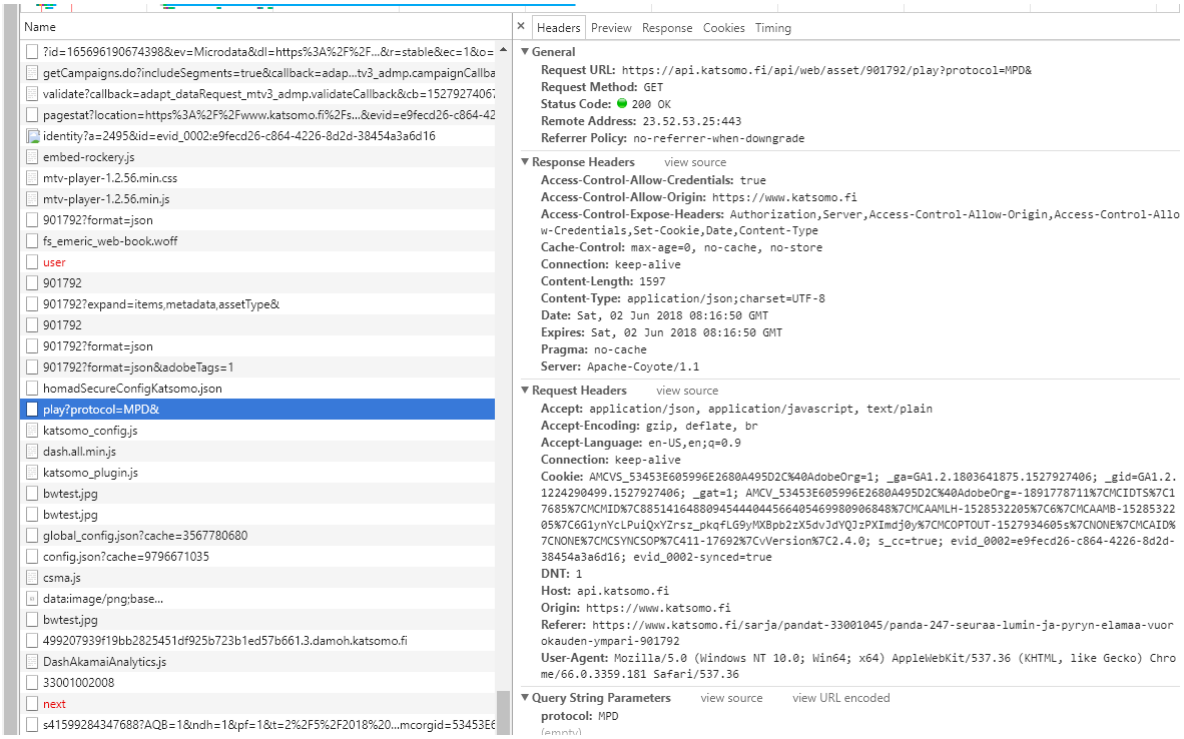
Kolmannen osapuolen tarjoama API päivittää tietokantaa ja hakee sieltä sovellusten tarvitsemat tiedot. APIsta haetaan tiedot esimerkiksi tarjolla olevista nimikkeistä, esitysajoista, kategorioista, käyttäjistä, käyttöoikeuksista ja videolähteistä. APIn kautta sisällöntuotannossa tietokantaan kirjoitetaan tiedot uusista nimikkeistä ja sitä käytetään automatiikan ohjaukseen. Tämä rajapinta ja tietokanta ovat korvaamaton osa infrastruktuuria. Infrastruktuuri on olennaisilta osiltaan yhteinen kaikille päätelaitteille.



Kuva 6.1 *Yksinkertaistettu kaavio edustapalvelimien infrastruktuurista.*

Kuvassa 6.1 on esitetty yksinkertaistettu malli web-palvelun infrastruktuurista. CDN:n kyselyt ohjataan välityspalvelimien kautta joko oikealle sovellukselle tai suoraan API:iin. Staattiset sivut haetaan suoraan tietovarastosta. Sovelluspalvelimet hyödyntävät monia eri resursseja, mutta tämä on piilotettu asiakkailta.

Kun asiakas haluaa toistaa videon, hän navigoi ensin mielenkiintoisen nimikkeen sivulle käyttöliittymässä. Painaessaan ”toista”-nappia, käyttäjä ohjataan sivulle, jolle ladataan palvelimelta mediasoitin. Soitin perustuu DASH-IF:n referenssisoittimeen ja dash.js -kirjastoon [27], jota on muokattu Katsomon tarpeisiin. Seuraavaksi pyydetään palvelimelta manifestia, jotta soitin voi toistaa sisältöä. Soitin pyytää manifestia APIlta, joka tietokannan tietojen mukaan koostaa URL:n. Erilaisilla sisällöillä on erilaisia konfiguraatioita ja ne voivat sijaita eri palvelimilla, joten manifestikin voidaan tarjoilla useammasta eri lähteestä tai muodostaa erilaiseksi eri parametreilla. APIn tarjoamasta URL:stä soitin saa manifestin ladattua. Kuka tahansa voi seurata tapahtumaketjua selaimen kehittäjätyökaluilla, jotka näyttävät HTTP-liikenteen asiakkaan ja palvelimen välillä.



Kuva 6.2 *Kuvankaappaus selaimen kehittäjätyökalusta. Korostettuna on palvelimelle lähetetty kutsu manifestin URL:n selvittämiseksi.*

Kuvasta 6.2 nähdään että sivulle ladataan myös paljon muuta soittimen ja manifestin lisäksi: itse web-sivu, kuvia joita sivulla näytetään, tyylitiedostoja, analytiikkaan liittyviä scriptejä ynnä muuta palvelun sisältöä. Kuvassa on kuitenkin korostettu pyyntö, jonka asiakas tekee palvelimelle. HTTP GET-metodin *Request URL* on koostettu dynaamisesti. Osa ’/web/’ ilmaisee palvelimelle, että sisältöä katsellaan selaimella. Seuraavaksi ’/asset/901792/’ ilmaisee että haluamme katsoa juuri tätä nimikettä, jonka id tietokannassa on 901792. Haettaessa resurssia ’play’ palautetaan XML-dokumentti, jossa kerrotaan manifestin sijainti. Parametri ’protocol=MPD’

viittaa siihen, että käytämme DASHia suoratoistoon (manifestin muoto on standardin mukainen mpd-tiedosto).

Serverin lähettämää vastausta on mielekkäintä tarkastella syöttämällä pyynnön kohde-URL *Request URL* selaimeen, tai lataamaalla se jollain muulla sopivalla ohjelmalla. Vastaus on XML-muotoinen dokumentti, josta selviää manifestin sijainnin lisäksi muuta informaatiota. Dokumentissa kerrotaan esimerkiksi, että käytämme salattua HTTPS-protkollaa, ohjelma on 'live', eli sisältö on suora lähetys, ja että streamia ei ole salattu. Vaikka sisältö on salaamatonta, kerrotaan kuitenkin mitä tekniikkaa käytettäisiin, jos tarvetta olisi. Koska esimerkissä on käytetty Googlen Chrome-selainta, on DRM-tekniikkana Googlen kehittämä Widevine [53] (jota siis käytettäisiin jos sisältö olisi salattu). APIn palauttamat arvot on kuvattu ohjelmassa 6.1 ja arvoja *base* ja *server* käytetään pohjana streamin lataamisessa. Vaikka DASHissa voidaan standardin mukaan käyttää useita eri lähdepalvimia, on Katsomossa käytössä kerrallaan vain yksi lähde.

```
<server>mtvdashliveusp4-a.akamaized.net</server>
<base>https://mtvdashliveusp4-a.akamaized.net</base>
```

Ohjelma 6.1 *Katsomon API kertoo mistä osoitteesta videota ja manifestia voi kysyä.*

Lopullinen sisällön latausosoite muodostetaan yhdistämällä näitä tietoja manifestin sisältöön. Edellä käsitellyn sisällön manifesti on liitteessä B. Manifestissa on kerrottu *BaseURL*-parametri, joka liitetään osaksi sisällön URL:ää. Manifestissa on määritelty esitykselle kaksi *AdaptionSet*-lohkoa, joista kokonainen esitys koostuu: video ja audio. Kuvassa 6.3 korostettuna oleva ladattava tiedosto on nimeltään *PANDA-video=2499968-38198262573.dash*. Tämä vastaa manifestissa määriteltyä templatea (ohjelma 6.2):

```
<SegmentTemplate
  timescale="25"
  initialization="PANDA-$RepresentationID$.dash"
  media="PANDA-$RepresentationID$-$Time$.dash">
```

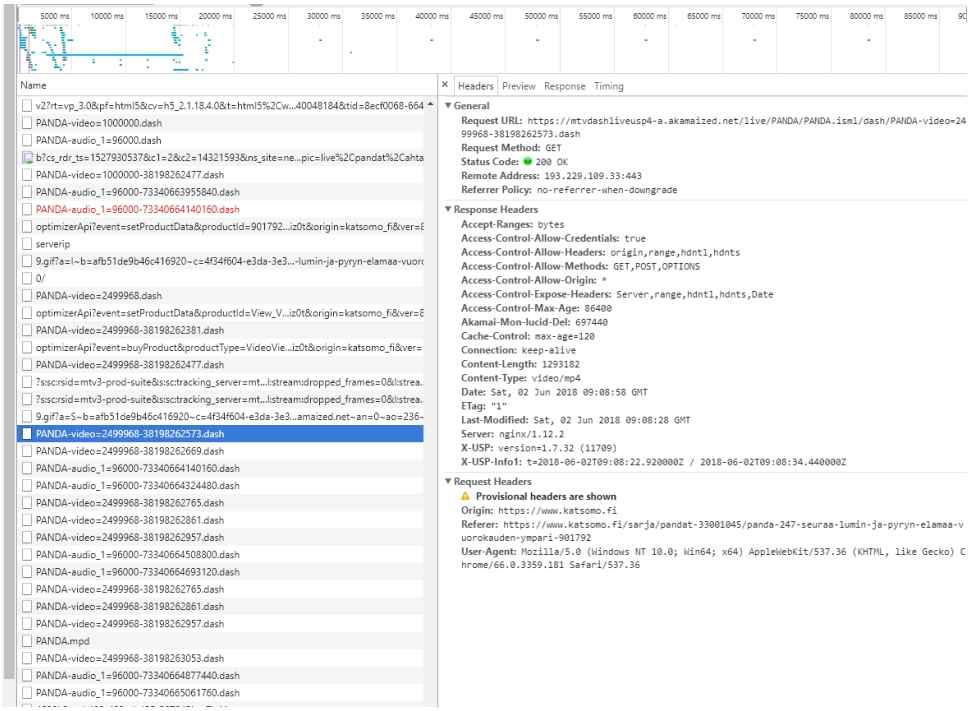
Ohjelma 6.2 *Oikeanlaisen HTTP-kyselyn muodostamiseen tarvittavia tietoja.*

Tässä siis *video=2499968* on RepresentationId ja *38198262573* on aikakoodi. Esimerkissä on videolle määritelty viisi erilaista versiota, jotka eroavat laadullisesti toisistaan. RepresentationId:llä *video=2499968* on ohjalmassa 6.3 määritelty esitys, jonka kaistanleveys on sama kuin sen id, eli 2499968.

```
<Representation
    id="video=2499968"
    bandwidth="2499968"
    width="1280"
    height="720"
    codecs="avc1.4D4020"
    scanType="progressive">
</Representation>
```

Ohjelma 6.3 *Yksittäisen videolaadun tiedot manifestissa.*

Id viitaa esityksen kaistanleveyteen. Kyseinen video on siis pakattu niin, että sen bittitiheys noin 2,5 megabittiä sekunnissa, se on korkeudeltaan 720 pikseliä ja leveydeltään 1280 pikseliä ja videokoodekkina on käytetty AVC1-koodekkia. *scanType="progressive"* viittaa videon transkoodauksen pakkaustapaan. Kuvasta 6.3 voidaan nähdä, että soitin on ladannut aikaisemmin heikompaa laatua, nimittäin *video=1000000*. Esimerkin kuvakaappaus on otettu aivan toiston alusta; suosittu käytäntö on aloittaa striimin lataus matalammalla laadulla, jotta toisto voitaisiin aloittaa mahdollisimman nopeasti ja luotettavasti. Myöhemmin nostetaan videon laatua, kun voidaan varmistua siitä että laitteisto ja verkko mahdollistavat tämän. Kuten aiemmin on todettu, voidaan samoin myös toiston laatua laskea olosuhteiden heikentyessä. Samoja periaatteita voidaan tietysti soveltaa myös muuhun sisältöön, kuten audioon, mutta tässä Katsomon esimerkkitapauksessa on käytössä vain yksi audioesitys.



Kuva 6.3 *Selain lataa video- ja audiopaketteja palvelimelta*

Kuvassa 6.3 näkyy, kuinka palvelimelta ladataan useita peräkkäisiä video- ja audiosegmenttejä. Perättäisiä video- ja audiosegmenttejä haettaessa soitin laskee missä kohtaa esitystä ollaan ajallisesti, ja kasvattaa sen mukaan aikaparametria HTTP-pyynnöissään. Manifestissa on arvoilla *minimumUpdatePeriod*, *maxSegmentDuration* ja *minBufferTime* viestitty soittimelle, kuinka usein videosegmenttejä pitää hakea. Asiakkaalla pitää siis olla aina katsottavissa 10 sekuntia materiaalia, yksi segmentti ei voi kestää yli neljää sekuntia ja manifesti pitää hakea uudelleen vähintään kahden sekunnin välein. Live-tilanteessa materiaalia ei välttämättä ole kymmentä sekuntia saatavilla, jos latenssit ovat pienet. Tällöin uutta sisältöä haetaan käytännössä aina kun sitä on saatavilla (tai kahden sekunnin välein, kun manifesti päivittyy).

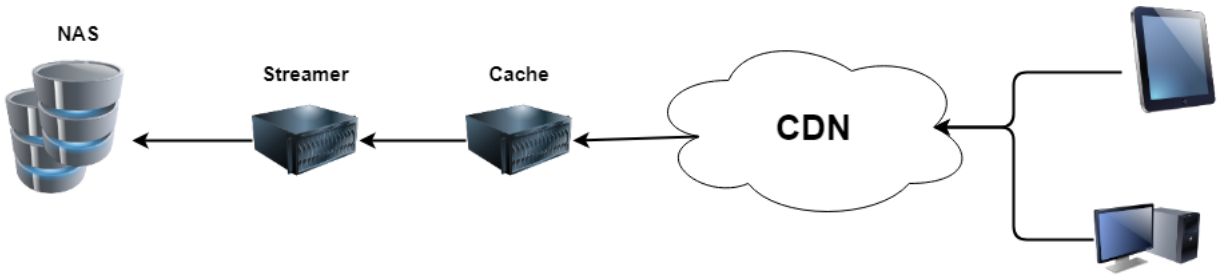
6.2.2 On-Demand video

Katsomo ja CMore tarjoavat asiakkaille katsottavaksi muun muassa uutisia, elokuvia, sarjoja ja urheilua. Elokuvat ja sarjat ovat luonnollisesti valmiiksi videotallenteina. Osa sisällöstä, kuten edellä mainituista urheilu ja uutiset esitetään ensin livenä. Livelähetyksestä tehdään tallenne, joka on jälkeenpäin katsottavissa palvelussa. Tallenteiden tuottaminen ei kuulu tämän diplomityön piiriin, tässä kohtaa meille riittää tieto siitä, että tallenteet ovat olemassa jossakin tietovarastossa ja Katsomon tietokannassa on tarvittavat tiedot ohjelmista.

Osa Katsomon videotallenteista sijaitsee ja jaellaan MTV:n tiloissa sijaitsevilta palvelimilta, osa taas AWS:n päälle rakennetusta infrastruktuurista. Katsomossa aloitettiin pilvipalveluiden hyödyntäminen vuonna 2015 ja tavoitteena on siirtää jatkuvasti suurempi osa infrastruktuurista ja jakelusta AWS:ään. MTV:n tiloissa olevan VOD-jakelun infrastruktuuri muodostuu levypalvelimesta, striimauspalvelimesta ja välimuistipalvelimista. Asiakkaan soitin pyrkii ensin lataamaan sisältöä manifestin osoittamasta URL:stä, eli CDN-noodilta. CDN itsessään tallentaa välimuistiin videon ja ääniraidan paloja ja palauttaa asiakkaalle halutun sisällön, mikäli se löytyy muistista. Tilan rajallisuuden ja kustannusten vuoksi välimuistissa on kuitenkin vain materiaalia, jota on katsottu viime aikoina. Kun sisältö on tarpeeksi vanhaa, tai kun tila välimuistista loppuu tila, aletaan sisältöä poistaa jotta saadaan tilaa uusille video- ja äänipaketeille. Jos asiakkaan hakemaa sisältöä ei löydy CDN:stä, haetaan se Katsomon palvelimilta.

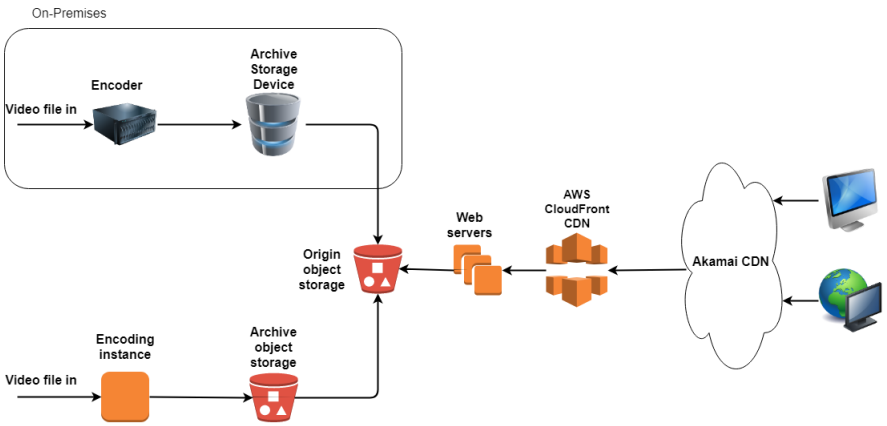
Katsomolla on myös oma välimuistipalvelin, joka toimii samalla periaatteella kuin CDN:n välimuistikin. Kuitenkin koska kyseessä on itse hallittu palvelin omissa tiloissa, voidaan kustannustehokkaasti käyttää huomattavasti suurempaa levykapasiteettia. Osin tämän ansiosta voidaan sisältöä myös säilyttää muistissa pidempiä aikoja,

jolloin saadaan nostettua edelleen todennäköisyyttä sille, että asiakkaan pyytämä sisältö löytyy välimuistista. Mikäli haluttu sisältö puuttuu tästäkin välimuistikerroksesta, haetaan se kovalevyltä. Streamauspalvelin on web-palvelin, jolla on pääsy verkkolevypalvelimelle. Välimuistipalvelin pyytää puuttuvan sisällön streamauspalvelimelta, joka hakee sisällön levyltä, muokkaa videon sopivaan muotoon jotta se sopii DASHin määritelmään, luo manifestin ja palauttaa nämä lopulta välimuistipalvelimelle. Tämä taas toimittaa sen edelleen CDN-noodille, mistä se lopulta päätyy asiakkaalle. Ketjua on havainnollistettu kuvassa 6.4. Tehokas välimuistin käyttö on tärkeätä viiveiden minimoimiseksi. Materiaalin hakeminen levyjärjestelmä voi olla hyvinkin hidasta verrattuna siihen, että data sijaitsee nopealla paikallisella kovalevyllä tai jopa palvelimen keskusmuistissa. Ilman tehokasta välimuistia on vaarana kapasiteetin loppuminen joko levyjärjestelmästä, verkosta tai streamauspalvelimista käyttäjämäärien kasvaessa.



Kuva 6.4 *Yksinkertaistettu arkkitehtuurikuva on-premises -videojakelusta.*

Jakeluketju toimii samalla periaatteella myös AWS:n päälle rakennetussa järjestelmässä. Fyysisten laitteiden sijasta käytetään kuitenkin virtuaalipalvelimia ja AWS:n tarjoamia palveluita.



Kuva 6.5 *Yksinkertaistettu arkkitehtuurikuva AWS-jakelusta. Amazonin CDN:n käyttö riippuu jaeltavasta sisällöstä.*

Sisältöä transkoodataan sekä MTV:n tiloissa, että AWS:n päällä. Jakelua varten tiedostot kuitenkin siirrettään AWS:n *Simple Storage Service (S3)* -tietovarastoon. Streamauspalvelimet hakevat sisällön S3:sta pyydettäessä. Mikäli sisältökohtainen konfiguraatio niin sanoo, sisältö tarjotaan AWS:n CDN-palvelulle *CloudFront*ille. CloudFront on täysiverinen CDN-palvelu, joka jakelun lisäksi myös tallettaa sisältö välimuistiinsa. CloudFrontista sisältö päätyy edelleen Katsomon pääasialliseen CDN-palveluun Akamaille. DASH-formaatissa olevan sisällön Akamai hakee suoraan origin-palvelimilta. AWS-videojakelu on kuvattu kuvassa 6.5. Kaikki liikenne tapahtuu salatulla HTTP-protokollalla. Katsomon videotallenteiden ja siirtostriimien formaatti perustuu historiallisista syistä Microsoftin SmoothStreaming-formaattiin, mutta streamauspalvelin paketoi datan uudelleen DASH-formaattiin asiakkaan käyttäessä tätä teknologiaa. Myös SmoothStreaming ja HLS ovat tuettuja teknologioita loppukäyttäjille asti.

Liite B sisältää esimerkin Katsomon live-sisällön manifestista. On-demand -sisältö noudattaa oleellisilta osilta samaa kaavaa. Merkittäviä eroja on manifestin alussa määritelty *type="static"*, joka kertoo että manifestin ei odoteta muuttuvan esityksen aikana. Lisäksi lohkon *SegmentTimeline* sisällä aikajana alkaa ohjelman 6.4 mukaan nollasta, eli esityksen alusta. Koska manifesti ei päivity, tarvitsee se hakea palvelimelta vain kerran.

```
<SegmentTimeline>
  <S t="0" d="96256" r="1309" />
  <S d="70656" />
</SegmentTimeline>
```

Ohjelma 6.4 *Noudattamalla kuvattua mallia voidaan hakea esityksen jokainen segmentti.*

Lisäksi suurin osa on-demand -sisällöstä on salattua. Tällöin manifestissa pitää kertoa tarvittavat tiedot, jotta asiakas voi hankkia itselleen avaimen ja purkaa salauksen.

```
<!-- Common Encryption -->
<ContentProtection
      schemeIdUri="urn:mpeg:dash:mp4protection:2011"
      value="cenc"
      cenc:default_KID="C92B906A-6C05-4890-AA81-2FAC7EA33105">
</ContentProtection>
<!-- PlayReady -->
<ContentProtection
      schemeIdUri="urn:uuid:9A04F079-9840-4286-AB92-E65BE0885F95"
      value="MSPR␣2.0">
</ContentProtection>
<!-- Widevine -->
<ContentProtection
```

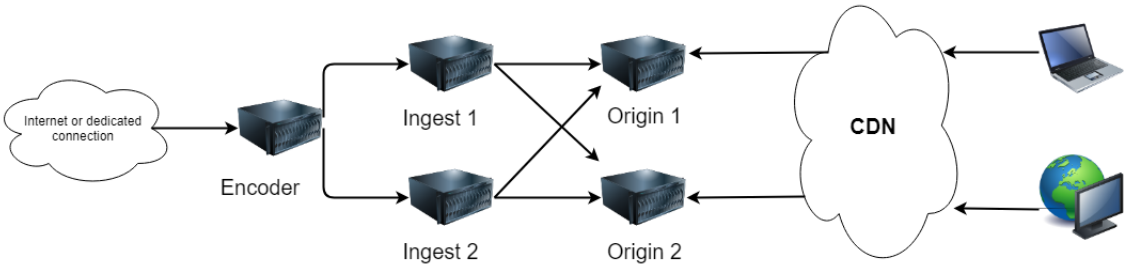
```
      schemeIdUri="urn:uuid:EDEF8BA9-79D6-4ACE-A3C8-27DCD51D21ED">
</ContentProtection>
<!-- Marlin -->
<ContentProtection
      schemeIdUri="urn:uuid:5E629AF5-38DA-4063-8977-97FFBD9902D4">
  <mas:MarlinContentIds>
    <mas:MarlinContentId>urn:marlin:kid:c92b906a6c054890aa812fac7ea33105</mas:M
  </mas:MarlinContentIds>
</ContentProtection>
```

Ohjelma 6.5 *Manifesti sisältöö kaiken tarvittavan tiedon siihen, että jokainen käytetty salaus voidaan purkaa.*

Ohjelman 6.5 esimerkissä on määritelty neljä erilaista salausta. Asiakkaan sovellus valitsee itse sopivan ja huolehtii salauksen purusta.

6.2.3 Live video

Live-sisältö jaellaan yksinomaan MTV:n konesalista. Näin saadaan minimoitua viiveet lähetyksissä. Live-lähetykset ovat usein myös liiketoiminnan kannalta kriittisempiä ja olemassa oleva infrastruktuuri on osoittautunut luotettavaksi ja toimivaksi. Iso osa sisällöstä myös tuotetaan MTV:n tiloissa (esimerkiksi uutiset ja moni kotimainen viihdeohjelma). Merkittävä osa livesisällöstä lähetetään samanaikaisesti myös televisiokanaville, jolloin kuva saadaan ”kaapattua” ja lähetettyä samalla myös Internetin yli. Muu sisältö tuodaan Internetin yli Katsomon enkoodereille.



Kuva 6.6 *Arkkitehtuurikuvaus Katsomon live-streamuksesta*

Enkooderi tuottaa sisään tulevasta signaalista kaikki halutut laadut, joita jaellaan loppukäyttäjille. Tässäkin tapauksessa väliformaattina on SmoothStreaming. Enkooderi syöttää streamia ingest-palvelimille, jotka puolestaan syöttävät streamia origin-palvelimille. Lopulta CDN hakee striimin originilta ja jakelee edelleen asiakkaille. Ketju on mallinnettu kuvassa 6.6. Ingest-palvelimet ovat Windows-palvelimia,

joihin on asennettu Microsoftin IIS-palvelinohjelmisto ja IIS:iin SmoothStreaming-lisäosa. Ingest- ja origin-kerrokset on kahdennettu laiterikkojen varalta. Origin paketoi streamin pyyntöjen mukaan DASH- tai HLS-formaattiin. Origin-kerros myös tallettaa levyllensä niin sanottua DVR (Digital Video Recording)-tallennetta. Tämä on tietyn mittainen osa lähetyksestä nykyhetkestä taaksepäin. DVR mahdollistaa liven kelaamisen taaksepäin enne kuin tapahtumasta on saatu julkaistua kokonainen tallenne. Enkooderi tuottaa lähetyksestä on-demand -tallennetta saman aikaisesti kun se tuottaa live-lähetystä.

Livessä manifestin tyyppi on *dynamic*, mikä kertoo soittimelle että manifesti päivittyy esityksen aikana. Manifestin *minimumUpdatePeriod* kertoo kuinka usein manifesti tulee hakea palvelimelta uudelleen. Liite B sisältää esimerkin Katsomon live-lähetyksestä. Kyseinen lähetys on erityinen, koska sillä ei varsinaisesti ole alkua eikä loppua, vaan lähetys on jatkuvasti käynnissä. Näitä kutsutaan ”Simulcast”-lähetyksiksi. Tällaisia lähetyksiä ovat esimerkiksi perinteiset televisiokanavat. Kaikki MTV:n tuottamat kanavat ovat katsottavissa myös Katsomossa. Jos kyseessä olisi esimerkiksi urheilulähetys, *SegmentTimeLine*n arvo t (lähetyksen "nykyhetki") olisi nolla, eli esitys alkaa alusta. Simulcast-lähetyksiä ei luonnollisesti voida tallettaa levylle, koska ne ovat käytännössä äärettömän pitkiä. Siksi asiakas ei myöskään voi pyytää katsottavaksi striimin alkua. Katsottavissa on nykyhetkestä taaksepäin DVR:n verran materiaali. DVR-ikkunan muodostaa d- ja r-arvot. D kuvaa videosegmentin pituutta (suhteessa timescale-arvoon), ja r kertoo monestiko tällainen segmentti toistuu esityksessä. Kun lähetys käynnistetään, r on nolla. Liitteen B esimerkissä se on saavuttanut määritellyn maksimiarvonsa 233.

7. YHTEENVETO

Videon suoratoisto on iso bisnes niin Suomessa kuin muuallakin maailmassa. Ala on kasvanut räjähdysmäisesti viime vuosien aikana ja teknologiat niin streamauksessa kuin ICT-alalla yleensä ovat kehittyneet huimasti, mahdollistaen sisällön streamauksen korkealla laadulla ja pienillä viiveillä niin kotiin kuin mobiililaitteisiin. Alalla isoja pelaajia ovat sisällöntuottajien lisäksi muun muassa operaattorit, ohjelmistotalot ja laitevalmistajat.

Erilaisten protokollien joukosta HTTP on muodostunut suosituksi tavaksi toimittaa myös audiota ja videota loppukäyttäjille. Protokolla on ollut pitkään käytössä webissä ja Internetin infrastruktuuri on kasvanut tukemaan tätä protokollaa. HTTP:tä käytettäessä voidaan esimerkiksi hyödyntää olemassa olevia CDN-noodeja ja perinteisiä web-palvelimia sisällön toimituksessa. Tuki HTTP:lle löytyy myös käytännössä katsoen kaikista laitteista, ja HTTP:tä tukevia sovelluskehitys työkaluja ja kirjastoja on runsaasti saatavilla. Koska HTTP on tilaton protokolla, ei asiakkaan ja palvelimen tarvitse pitää jatkuvaa yhteyttä auki, jolloin järjestelmä on robustimpi. Protokolla mahdollistaa myös erilaisten asetusten ja lippujen toimittamisen joko osana hyötykuormaa tai HTTP-paketin header-osiossa. TCP:hen perustuen toimitus on luotettavaa.

Kilpaillulla alalla on myös useita kilpailevia teknologioita, joista suurimmat ovat työn aiheena oleva DASH, sekä Microsoftin Smooth Streaming, Adoben HDS ja Applen HLS. Kaikki edellä mainitut perustuvat HTTP-protokollaan, ja ovat adaptiivisia streamaustekniikoita. Adobe nojaa Flash-teknologiaan, jonka tuki on päättymässä ja täten myös HDS ei tule jatkumaan. Microsoft tukee tuotteissaan myös DASHia ja onkin yksi DASH Industry Forumin perustajajäsenistä. Applen voidaan katsoa myös lisänneen tukeaan DASHille. DASH erottuu joukosta ainoana täysin avoimena ja vapaana standardina. Sen standardin virallinen kehitys on alkanut vuonna 2012 ja jatkuu edelleen. Sen takana on useita multikansallisia yrityksiä ja tuotantoyhtiöitä. Tavoitteena on luoda streamauksen standardi, joka toimisi kaikilla alustoilla ja olisi mahdollisimman joustava, muokattavissa ja joka vastaisi nykymaailman tarpeisiin. DASH ei tarvitse palvelimen puolelta mitään erityistä ohjelmistoa, vaan sisältöä ja manifesteja voidaan jakaa tavallisilla web-palvelimilla. DASH-IF

tarjoaa vapaasti käytettäväksi dash.js-kirjaston ja referenssisoittimen, jolla sisältöä voidaan katsoa asiakkaan laitteissa. Asiakkaalta vaaditaan tuki HTML5:lle ja selaimen MSE- ja EME-laajennokset. DASH on koodekkiagnostinen, eikä se ota suoraan kantaa käytettyyn DRM-ratkaisuun. Siihen on sisäänrakennettu ratkaisu mainosten toistamiselle ja tuki usealle eri ääni-, tekstitys- ja videoraidalle. Sisältöä ei tarvitse segmentoida valmiiksi palvelimelle, vaan kokonaisesta esityksestä voidaan hakea tarvittavat osat HTTP byte range -kutsulla. DASH tukee useata vaihtoehtoista CDN-palvelua (originia), tehden järjestelmästä vikasietoisemman. Kaikista uusimmista selaimista löytyy natiivisti tuki DASHille. DASH on virallinen ISO standardi: ISO/IEC 23009-1.

Kasvavan tuen, aktiivisen kehityksen ja vaikuttavien taustajoukkojen myötä DASHin voidaan odottaa yleistyvän. Sille on jo nyt laaja tuki, mutta kaikki palveluntarjoajat eivät vielä ole siirtyneet DASHin käyttöön vanhoista järjestelmistään. Joustavuuden ja avoimen standardin vuoksi se on helposti käyttöönotettavissa ja muokattavissa moniin tarpeisiin. Useista eri teknologioista yhteen standardiin siirtyminen helpottaa niin sisällöntuottajia, sovelluskehittäjiä kuin kuluttajiakin. Sisällöntuottajat voivat toimittaa jakeluun yhden version sisällöstä, joka sitten toimii kaikkialla. Tällä säästetään varasto- ja tiedonsiirtokuluissa sekä aikaa ja laskentatehoa enkoodereilta. Sovelluskehittäjien ei tarvitse tehdä kaikille alustoille erikseen soitinta ja sen ympärille sovellusta, sillä yksi sovellus toimisi useammalla alustalla. Loppukäyttäjien ei tarvitse huolehtia erinäisistä lisäosista ja ohjelmista, joita vaaditaan erilaisten sisältöjen katseluun.

LÄHTEET

[1] United States Securities and Exchange Commission, "Annual report pursuant to section 13 or 15(d) of the securities exchange act of 1934." report [online]. Saatavilla: http://files.shareholder.com/downloads/NFLX/5217561654x0x938338/FB0485BA-48EF-4457-ABED-CF26A5B21523/10K_Final.PDF, Viitattu: 01/2017.

[2] United States Securities and Exchange Commission, "Annual report pursuant to section 13 or 15(d) of the securities exchange act of 1934, amazon.com inc.." report [online]. Saatavilla: https://www.sec.gov/Archives/edgar/data/1018724/000101872417000011/amzn-20161231x10k.htm, Viitattu: 02/2017.

[3] Youtube, "Youtube in numbers." report [online], Toukokuu 2017. Saatavilla: https://www.youtube.com/intl/en-GB/yt/about/press/, Viitattu 17.9.2017.

[4] Cisco Systems, "Cisco visual networking index: Forecast and methodology." whitepaper [online]. Saatavilla: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html, Päivitetty: 09/2017.

[5] T. Stockhammer, "Dynamic adaptive streaming over http design principles and standards," *MMSys '11 Proceedings of the second annual ACM conference on Multimedia systems*, pp. 133–144, February 2011.

[6] Akamai Technologies, "Our customers." website [online]. Saatavilla: https://www.akamai.com/us/en/our-customers.jsp, Viitattu: 10/2017.

[7] Akamai Technologies, "State of the internet q4 2016." report [online]. Saatavilla: https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q4-2016-state-of-the-internet-connectivity-report.pdf, Viitattu: 03/2017.

[8] C. Timmer and C. Müller, "Http streaming of mpeg media." Article, Streaming Day, 2010. Saatavilla: https://svn-itec.uni-klu.ac.at/trac2/dash/export/32/trunk/documentation/01%20STDay-2010-v1.1.pdf.

[9] DASH Industry Forum, "Members." website [online]. Saatavilla: http://dashif.org/members/, Viitattu 21.10.2017.

[10] ISO/IEC, "Information technology - dynamic adaptive streaming over http (dash), part 1: Media presentation description and segmen formats," *International standard*, 2014. Saatavilla: `http://standards.iso.org/ittf/PubliclyAvailableStandards/c065274_ISO_IEC_23009-1_2014.zip`.

[11] DASH Industry Forum, "About." website [online]. Saatavilla: `http://dashif.org/about/`, Viitattu 7.1.2018.

[12] Google, "Google scholar." search engine. `https://scholar.google.fi/`.

[13] J. Kurose and K. Ross, *Computer Networking - A Top-Down Approach*. Pearson, sixth ed., 2012.

[14] Apple Inc., "Http live streaming." website. Saatavilla: `https://developer.apple.com/streaming/`.

[15] Adobe Inc., "Adobe media server technical overview guide." website [online]. Saatavilla: `https://helpx.adobe.com/adobe-media-server/tech-overview/topics.html`.

[16] Microsoft, "Smooth streaming deployment guide." website [online]. Saatavilla: `https://docs.microsoft.com/en-us/iis/media/smooth-streaming`.

[17] StreamingMedia.com, "Official website." website [online]. Saatavilla: `http://www.streamingmedia.com/`.

[18] T. C. Thang and Q.-D. Ho, "Adaptive streaming of audiovisual content using mpeg dash," *IEEE Transactions on Consumer Electronics*, vol. 58, pp. 78–85, February 2012.

[19] Unified Steraming, "Adaptive bitrate (abr) streaming." website [online], http://docs.unified-streaming.com/documentation/vod/streaming.html.

[20] IETF, "Hypertext transfer protocol (http/1.1): Range requests." RFC [online], Huh 2012. Saatavilla: `https://tools.ietf.org/html/rfc7233#section-2.1`.

[21] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the internet," *IEEE Multimedia*, pp. 62–67, October-December 2011.

[22] ISO, "Iso/iec 23009-1:2012." standard [online], https://www.iso.org/standard/57623.html, Jun 2014.

[23] mozilla.org, "Audio and video on the web." [online]. Saatavilla: `https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Video_and_audio_content`,Viitattu: 3.4.2018.

[24] Google Inc., "Chrome 30 beta: A richer web on android." blog [online], Elo 2013. Saatavilla: https://blog.chromium.org/2013/08/chrome-30-beta-richer-web-on-android.html,Viitattu: 3.4.2018.

[25] mozilla.org, "42.0 firefox release." website [online], Mar 2015. Saatavilla: https://www.mozilla.org/en-US/firefox/42.0/releasenotes/,Viitattu: 3.4.2018.

[26] DASH Industry Forum, "Dash.js." website [online]. Saatavilla: https://github.com/Dash-Industry-Forum/dash.js/wiki,{Viitattu:}4.4.2018.

[27] DASH Industry Forum, "dash.js javascript reference client." website [online]. Saatavilla: http://reference.dashif.org/dash.js/,Viitattu: 4.4.2018.

[28] DASH Industry Forum, "Dash if test assets database." website [online]. Saatavilla: http://testassets.dashif.org/#testvector/list,Viitattu: 4.4.2018.

[29] R. Grandl, "Wwdc16: Hls supports fragmented mp4 and becomes mpeg-dash compatible!." online, Kes 2016. Saatavilla: https://bitmovin.com/hls-news-wwdc-2016/,Viitattu: 4.8.2018.

[30] Adobe Inc., "Product information." website [online]. Saatavilla: https://www.adobe.com/fi/products/adobe-media-server-standard/faq.html,Viitattu: 21.8.2018.

[31] Microsoft, "Mpeg-dash and smooth streaming test content." website [online]. Saatavilla: http://playready.azurewebsites.net/Home/AdaptiveTests,Viitattu: 21.8.2018.

[32] Apple Inc., "Using http live streaming." website. Saatavilla: https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html,Viitattu: 4.8.2018.

[33] Adobe Inc., "Buying guide." website [online]. Saatavilla: https://www.adobe.com/products/adobe-media-server-standard/buying-guide-pricing.html,Viitattu: 4.8.2018.

[34] Adobe Corporate Communications, "Flash & the future of interactive content." blog [online], Hei 2017. Saatavilla: https://theblog.adobe.com/adobe-flash-update/,Viitattu: 21.8.2018.

[35] Adobe Inc., "Products and technical support periods." website [online]. Saatavilla: https://helpx.adobe.com/support/programs/eol-matrix.html#KM,Viitattu: 21.8.2018.

[36] Adobe Inc., "Adobe primetime pricing." website [online]. Saatavilla: https://www.adobe.com/fi/marketing-cloud/primetime/pricing.html, Viitattu: 15.8.2018.

[37] Adobe Inc., "Adobe access overview." website [online], Huhtikuu 2014. Saatavilla: https://www.adobe.com/support/adobeaccess/pdfs/server/AdobeAccess_4_Overview.pdf, Viitattu: 15.8.2018.

[38] Microsoft, "Frequently asked questions." website [online]. Saatavilla: https://www.microsoft.com/playready/licensing/faq/, Viitattu: 15.8.2018.

[39] Unified Steraming, "Unified streaming platform." website [online]. Saatavilla: https://www.unified-streaming.com/, Viitattu 30.6.2018.

[40] Wowza media systems. Official website [online]. Saatavilla: https://www.wowza.com/,Viitattu: 17.8.2018.

[41] Microsoft Edge Team, "Simplified adaptive video streaming: Announcing support for hls and dash in windows 10." blog [online], Tam 2015. Saatavilla: https://blogs.windows.com/msedgedev/2015/01/29/simplified-adaptive-video-streaming-announcing-support-for-hls-and-dash-in-w 21.8.2018.

[42] Mozilla and individual contributors, "Live streaming web audio and video." online, Elo 2018. Saatavilla: https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Audio_and_video_delivery/Live_streaming_web_audio_and_video,Viitattu: 21.8.2018.

[43] Apple Inc., "About apple's http live streaming tools." website. Saatavilla: https://developer.apple.com/documentation/http_live_streaming/about_apple_s_http_live_streaming_tools,Viitattu: 21.8.2018.

[44] Adobe Inc., "Video streaming made simple." online. Saatavilla: https://www.adobe.com/fi/products/adobe-media-server-standard.html,Viitattu: 21.8.2018.

[45] R. Reinhardt, "Gone in a flash: Migrating videos to a flash-less world." Streaming Media Magazine, January/February 2017. Saatavilla: http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/

Gone-in-a-Flash-Migrating-Videos-to-a-Flash-Less-World-115771.aspx,Viitattu: 21.8.2018.

[46] MTV, "Mtv lyhyesti." blog [online]. Saatavilla: https://www.mtv.fi/yritys,Viitattu: 4.4.2018.

[47] Bonnier AB, "Annual report 2016." report [online], Huh 2017. Saatavilla: https://www.bonnier.com/globalassets/about-us/annual-reviews/2016/bonnier-annual-report-2016-final.pdf, Viitattu 4.4.2018.

[48] TNS-Gallup, "Suomen web-sivustojen viikkoluvut." report [online]. Saatavilla: http://tnsmetrix.tns-gallup.fi/public/,Viitattu: 4.4.2018.

[49] K. Sinkko, "Mtv, miksi silverlight? ilmalan videojakelun lyhyt historia." blog [online], Huhtikuu 2017. Saatavilla: http://sinkko.org/blogs/sinkko/2017/04/14/mtv-miksi-silverlight/,Viitattu: 4.4.2018.

[50] Amazon Web Services, "Our customers." website, [online], https://aws.amazon.com/.

[51] A. G. Tharakan and S. Patnaik, "Strong dollar hurts akamai's profit forecast, shares fall." Streaming Media Magazine, January/February 2017. Saatavilla: https://www.reuters.com/article/us-akamai-tech-results/-strong-dollar-hurts-akamais-profit-forecast-shares-fall-idUSKBN0NJ2IV20150
5.4.2018.

[52] The Apache software foundation, "Apache http server project." website, [online], https://httpd.apache.org/.

[53] Google Inc., "Widevine technologies." website [online], Marraskuu 2018. Saatavilla: http://www.widevine.com/,Viitattu: 2.5.2018.

LIITE A. ESIMERKKI MPD:STÄ

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="urn:mpeg:dash:schema:mpd:2011"
    xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011␣DASH-MPD.xsd"
    type="static"
    mediaPresentationDuration="PT3256S"
    minBufferTime="PT1.2S"
    profiles="urn:mpeg:dash:profile:isoff-on-demand:2011">

   <BaseURL>http://cdn1.example.com/</BaseURL>
   <BaseURL>http://cdn2.example.com/</BaseURL>

 <Period>
        <!-- English Audio -->
  <AdaptationSet mimeType="audio/mp4" codecs="mp4a.40"
    lang="en"subsegmentAlignment="true" subsegmentStartsWithSAP="1">
    <ContentProtection
      schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-656164657221"/>
    <Representation id="1" bandwidth="64000">
          <BaseURL>7657412348.mp4</BaseURL>
    </Representation>
    <Representation id="2" bandwidth="32000">
        <BaseURL>3463646346.mp4</BaseURL>
    </Representation>
  </AdaptationSet>
  <!-- French Audio -->
  <AdaptationSet mimeType="audio/mp4" codecs="mp4a.40.2" lang="fr"
    subsegmentAlignment="true" subsegmentStartsWithSAP="1">
    <ContentProtection
      schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-656164657221"/>
      <Role schemeIdUri="urn:mpeg:dash:role:2011" value="dub"/>
      <Representation id="3" bandwidth="64000">
        <BaseURL>3463275477.mp4</BaseURL>
      </Representation>
      <Representation id="4" bandwidth="32000">
        <BaseURL>5685763463.mp4</BaseURL>
      </Representation>
  </AdaptationSet>
  <!-- Timed text -->
  <AdaptationSet mimeType="application/ttml+xml" lang="de">
        <Role schemeIdUri="urn:mpeg:dash:role" value="subtitle"/>
      <Representation id="5" bandwidth="256">
        <BaseURL>796735657.xml</BaseURL>
```

```
      </Representation>
  </AdaptationSet>
  <!-- Video -->
  <AdaptationSet mimeType="video/mp4" codecs="avc1.4d0228"
    subsegmentAlignment="true" subsegmentStartsWithSAP="2">
    <ContentProtection
      schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-656164657221"/>
    <Representation id="6" bandwidth="256000" width="320" height="240">
          <BaseURL>8563456473.mp4</BaseURL>
    </Representation>
    <Representation id="7" bandwidth="512000" width="320" height="240">
          <BaseURL>56363634.mp4</BaseURL>
    </Representation>
    <Representation id="8" bandwidth="1024000" width="640" height="480">
          <BaseURL>562465736.mp4</BaseURL>
    </Representation>
    <Representation id="9" bandwidth="1384000" width="640" height="480">
          <BaseURL>41325645.mp4</BaseURL>
    </Representation>
    <Representation id="A" bandwidth="1536000" width="1280" height="720">
          <BaseURL>89045625.mp4</BaseURL>
    </Representation>
    <Representation id="B" bandwidth="2048000" width="1280" height="720">
          <BaseURL>23536745734.mp4</BaseURL>
    </Representation>
  </AdaptationSet>
  </Period>
</MPD>
```

Lähde: ISO/IEC 23009-1:2014(E): DASH International Standard: Part 1: Media presentation description and segment formats, Annex G. 15.5.2017

LIITE B. ESIMERKKIMANIFESTI KATSOMON LIVESISÄLLÖSTÄ

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Created with Unified Streaming Platform(version=1.7.32) -->
<MPD
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:mpeg:dash:schema:mpd:2011"
  xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011␣\
␣␣␣␣http://standards.iso.org/ittf/PubliclyAvailableStandards/␣\
␣␣␣␣MPEG-DASH_schema_files/DASH-MPD.xsd"
  type="dynamic"
  availabilityStartTime="2018-04-26T07:58:11Z"
  publishTime="2018-06-02T09:04:51.093220Z"
  minimumUpdatePeriod="PT2S"
  timeShiftBufferDepth="PT14M58.560S"
  maxSegmentDuration="PT4S"
  minBufferTime="PT10S"
  profiles="urn:mpeg:dash:profile:isoff-live:2011,urn:com:dashif:dash264">
  <Period
    id="1"
    start="PT0S">
    <BaseURL>dash/</BaseURL>
    <AdaptationSet
      group="1"
      contentType="audio"
      lang="en"
      segmentAlignment="true"
      audioSamplingRate="48000"
      mimeType="audio/mp4"
      codecs="mp4a.40.2"
      startWithSAP="1">
      <AudioChannelConfiguration
        schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011"
        value="2">
      </AudioChannelConfiguration>
      <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main" />
      <SegmentTemplate
        timescale="48000"
        initialization="PANDA-$RepresentationID$.dash"
        media="PANDA-$RepresentationID$-$Time$.dash">
        <SegmentTimeline>
          <S t="73340610134400" d="184320" r="233" />
        </SegmentTimeline>
      </SegmentTemplate>
```

```
    <Representation
      id="audio_1=96000"
      bandwidth="96000">
    </Representation>
  </AdaptationSet>
  <AdaptationSet
    group="2"
    contentType="video"
    par="16:9"
    minBandwidth="499968"
    maxBandwidth="2499968"
    maxWidth="1280"
    maxHeight="720"
    segmentAlignment="true"
    sar="1:1"
    frameRate="25"
    mimeType="video/mp4"
    startWithSAP="1">
    <Role schemeIdUri="urn:mpeg:dash:role:2011" value="main" />
    <SegmentTemplate
      timescale="25"
      initialization="PANDA-$RepresentationID$.dash"
      media="PANDA-$RepresentationID$-$Time$.dash">
      <SegmentTimeline>
        <S t="38198234445" d="96" r="233" />
      </SegmentTimeline>
    </SegmentTemplate>
    <Representation
      id="video=499968"
      bandwidth="499968"
      width="512"
      height="288"
      codecs="avc1.4D401E"
      scanType="progressive">
    </Representation>
    <Representation
      id="video=749952"
      bandwidth="749952"
      width="640"
      height="360"
      codecs="avc1.4D401E"
      scanType="progressive">
    </Representation>
    <Representation
      id="video=1000000"
      bandwidth="1000000"
      width="768"
```

```
        height="432"
        codecs="avc1.4D401E"
        scanType="progressive">
      </Representation>
      <Representation
        id="video=1499968"
        bandwidth="1499968"
        width="864"
        height="486"
        codecs="avc1.4D401F"
        scanType="progressive">
      </Representation>
      <Representation
        id="video=2499968"
        bandwidth="2499968"
        width="1280"
        height="720"
        codecs="avc1.4D4020"
        scanType="progressive">
      </Representation>
    </AdaptationSet>
  </Period>
  <UTCTiming
    schemeIdUri="urn:mpeg:dash:utc:http-iso:2014"
    value="https://time.akamai.com/?iso" />
</MPD>
```