Verifikacija DASH podrške u HbbTV okruženju

Nikić, Nemanja

Master's thesis / Diplomski rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:317104

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-26**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

VERIFIKACIJA DASH PODRŠKE U HBBTV OKRUŽENJU

Diplomski rad

Nemanja Nikić

Osijek, 2016.

Sadržaj

1. UVOD *1*

2. TEORIJSKE OSNOVE *2*

2.1 HTTP **2**

2.1.1 TCP/IP i OSI model 3

2.2 HbbTV **4**

2.4 Pregled HbbTV sustava **6**

2.4.1 Arhitektura HbbTV sustava 9

2.5 Sustav za testiranje **10**

2.5.1 Skup testova 11

2.5.2 Testno okruženje 12

3. SPECIFIKACIJE I PROCES STVARANJA TESTNIH SLUČAJEVA *13*

4. TEST API I DEFINICIJA PLAYOUT SET-A *15*

5. MPEG DASH *21*

5.1 DASH manifest – MPD **23**

5.1.1 Period 23

5.1.2 AdaptationSet 23

5.1.3 Representation 23

5.1.4 MediaSegment 24

5.1.5 IndexSegment 25

5.2 MPEG DASH profili, ... **27**

5.2.1 ISOBMFF On Demand profil 28

5.2.2 ISOBMFF Live profil 28

5.2.3 MPEG-2 TS Main profil 29

5.2.4 MPEG-2 TS Simple profil 29

6. DASH KLIJENT *30*

6.1 MSE **32**

7. OPIS I PRIMJER TESTA *35*

7.1 XML struktura DASH testa **35**

7.2 HTML5 DASH test **35**

7.3 T2Unit testovi i testni plan **36**

8. ZAKLJUČAK *39*

LITERATURA *40*

POPIS KRATICA *41*

SAŽETAK *43*

ABSTRACT *43*

ŽIVOTOPIS *44*

PRILOZI *45*

1. UVOD

U ovom radu dan je pregled HbbTV-a (*Hybrid Broadcast Broadband TV*), industrijskog standarda koji omogućuje realizaciju hibridnih interaktivnih DTV aplikacija zasnovanim na web tehnologijama HTML (*Hyper Text Markup Language*), JavaScript i CSS (*Cascading Style Sheets*). Te aplikacije u svom radu koriste emitirani prijenosni tok (*broadcast*) i vezu s internetom (*broadband*). Zahvaljujući obilju web platformi i vezi s internetom, HTTP (*Hypertext Transfer Protocol*) je postao isplativ za bilo kakvu isporuku podataka (multimedije). Zbog takvog trenda od strane MPEG (*Moving Picture Experts Group*) i 3GPP (*3rd Generation Partnership Project*) razvijen je novi standard DASH (*Dynamic Adaptive Streaming over HTTP*). On definira prilagodljive tehnike slanja multimedijskih sadržaja preko HTTP-a, te omogućava interoperabilnost u industriji, i standard je kojeg HbbTV 2.0 certificirani uređaji moraju implementirati. Zbog raznovrsnosti u današnjim komunikacijskim mrežama prilagodljivost predstavlja najvažniji zahtjev za strujajućeg klijenta. Naročito kada se govori o TCP-u (*Transmission Control Protocol*), sloju ispod HTTP-a, koji je poznat po svojim "oscilacijama" u propusnosti. Kod HTTP strujanja, server uglavnom nema puno podataka o statusu klijent/mreža. Obično je klijent taj koji donosi odluke o dostavljanju sadržaja kako bi se zadržala zadovoljavajuća kvaliteta usluge (QoS - *Quality of Service*). Zbog tog razloga klijent treba imati dobre signalizacijske metapodatke koji moraju biti spremni unaprijed ili dobavljeni tijekom izvođenja (*on the fly*) od strane sadržajnog/servisnog poslužitelja. Dan je pregled HbbTV standarda s posebnim osvrtom na DASH funkcionalnost, te detaljno opisan sam DASH standard te njegova MPD (*Media Presentation Description*) struktura i uloga. Realiziran je podskup testnih slučaja kojima se provjerava pravilna implementacija podrške za DASH sadržaje korištenjem HbbTV test API-a (*Application Programming Interface*) u skladu s pravilima testiranja koja definira HbbTV udruženje. Također je, uz pomoć *MediaSource* API-a koji omogućava stvaranje toka medija (*stream*-a) u pretraživaču, proširena postojeću JavaScript biblioteka s DASH funkcionalnostima koje su neophodne za provjeru testnih slučaja u okruženju web pretraživača.

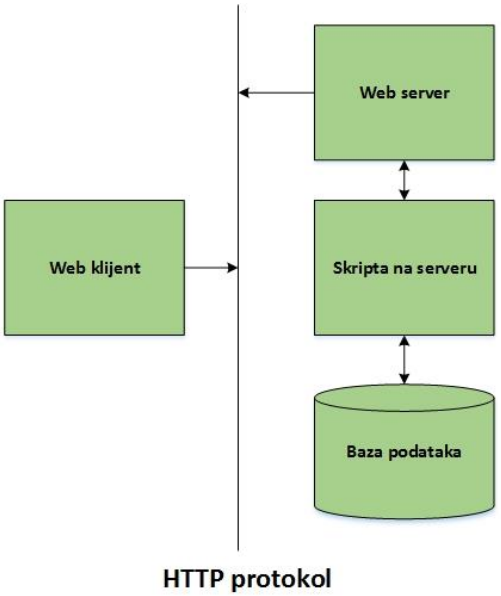
2. TEORIJSKE OSNOVE

U ovom poglavlju su opisane teorijske osnove pojmova na kojima se zasniva rad, kao što su HTTP, HbbTV, DASH, Harness i ostalo.

2.1 HTTP

HTTP je protokol aplikacijske razine za prijenos datoteka koje u sebi sadrže veze na druge dokumente. Ti dokumenti se označavaju kao hipertekst, a veze koje sadrže zovu se hiperveze (*hyperlinks*). Koristi se od 1990. godine pojavom Interneta. HTTP 1.0 definiran je u dokumentu RFC 1945, a HTTP 1.1 u RFC 2068. Temelji se na modelu klijent-poslužitelj, i za prijenos podataka zahtijeva pouzdanu vezu na prijenosnoj razini. Ne vezuje se uz određeni protokol već u budućim primjenama dopušta i neki drugi protokol osim TCP-a. Standardna priključna točka (*port*) na kojoj sluša web poslužitelj je 80, iako se može definirati i drugačije. U HTTP 1.0 protokolu definirano je otvaranje odvojenih TCP veza za prijenos svakog dokumenta. Recimo na primjer da će HTML dokument koji poziva dvije slike unutar stranice izazvati otvaranje dvije nove TCP veze od klijenta ka poslužitelju. U HTTP 1.1 između klijenta i poslužitelja uspostavlja se stalna HTTP veza, koja služi za razmjenu podataka više zahtjeva između klijenta i poslužitelja. Time se štedi procesorsko vrijeme poslužitelja, smanjuje nepotreban promet i umanjuje mogućnost pojave zagušenja. HTTP protokolom definira se i [1]:

1. Forma komunikacije između klijenta i poslužitelja, tj. način postavljanja upita i odgovora i njihov format
2. Kodiranje znakova karakterističnih za brojne jezike (*character set*)
3. Kodiranje sadržaja (*content coding*)
4. Pristup dokumentima za različite tipove protokola
5. Pristup dokumentima uz provjeru identiteta (autorizacija i provjera autentičnosti)
6. Spremanje dokumenata u privremenu memoriju (*caching*)
7. Sigurnosne aspekte -osjetljive točke u komunikaciji između klijenta i poslužitelja (proces dostave podataka korisniku)



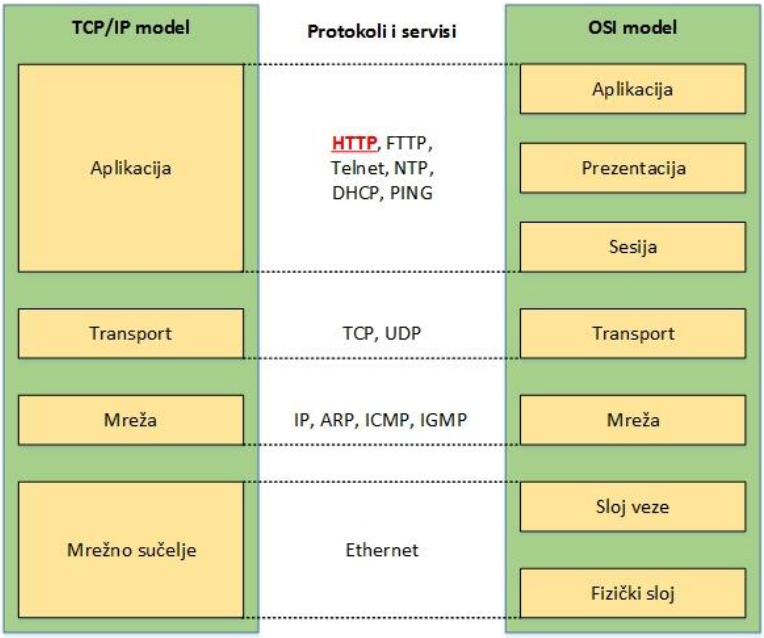
Slika 2.1. Mjesto gdje se nalazi HTTP u osnovnoj arhitekturi web aplikacije[1]

Također je dana struktura TCP/IP i OSI modela i prikazano gdje se nalazi HTTP (aplikacijski sloj).

2.1.1 TCP/IP i OSI model

OSI (*Open Systems Interconnections model*) je konceptualni model koji standardizira i karakterizira funkcije komunikacijskog ili računalnog sustava gdje pri tome ne gleda strukturu i tehnologiju koju sustav koristi. Cilj OSI modela je interoperabilnost između različitih komunikacijskih sustava korištenjem standardnih protokola. Model je opisan u sedam slojeva. Sloj je posluživan od sloja ispod, te on poslužuje sloj iznad. Npr. sloj ispod koji pruža komunikaciju mreže bez pogreške (*error-free*), osigurava put potreban za aplikaciju koja je iznad njega te poziva idući sloj da šalje i prima pakete koji obuhvaćaju sadržaj tog puta. TCP/IP (*Transmission Control Protocol/Internet Protocol*) je temeljni protokol IPS-a (*Internet Protocol Suite*). Viši sloj, odnosno TCP upravlja rastavljanjem poruke ili dokumenta u manje pakete koji se prijenose preko interneta i primaju od strane TCP sloja koji sastavlja pakete u originalnu poruku. Niži sloj tj. IP rukuje adresnim dijelom kako bi svaki paket završio na pravom odredištu. Svako računalo na mreži provjerava ovu adresu kako bi znalo gdje proslijediti poruku. Iako su neki paketi iz iste poruke preusmjereni različito od drugih, na krajnjoj destinaciji će biti ispravno spojeni. Neke od razlika između OSI i TCP/IP modela su [2]:

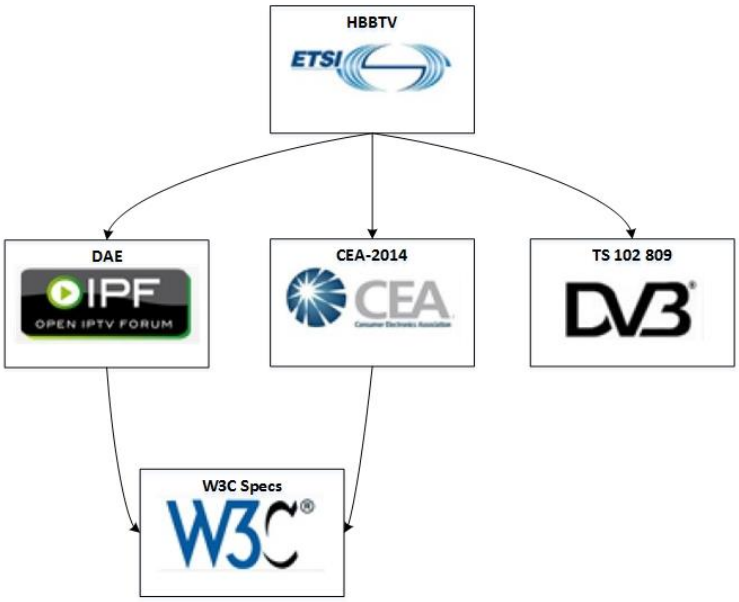
- TCP/IP je jednostavniji model jer se sastoji samo od 4 sloja u odnosu na OSI koji se sastoji od 7
- TCP/IP je pouzdaniji jer je Internet razvijen oko njega
- TCP/IP kombinira OSI podatkovni i fizički sloj u mrežni
- OSI je referentni model dok je TCP/IP njegova implementacija



Slika 2.2. Prikaz OSI i TCP/IP modela[2]

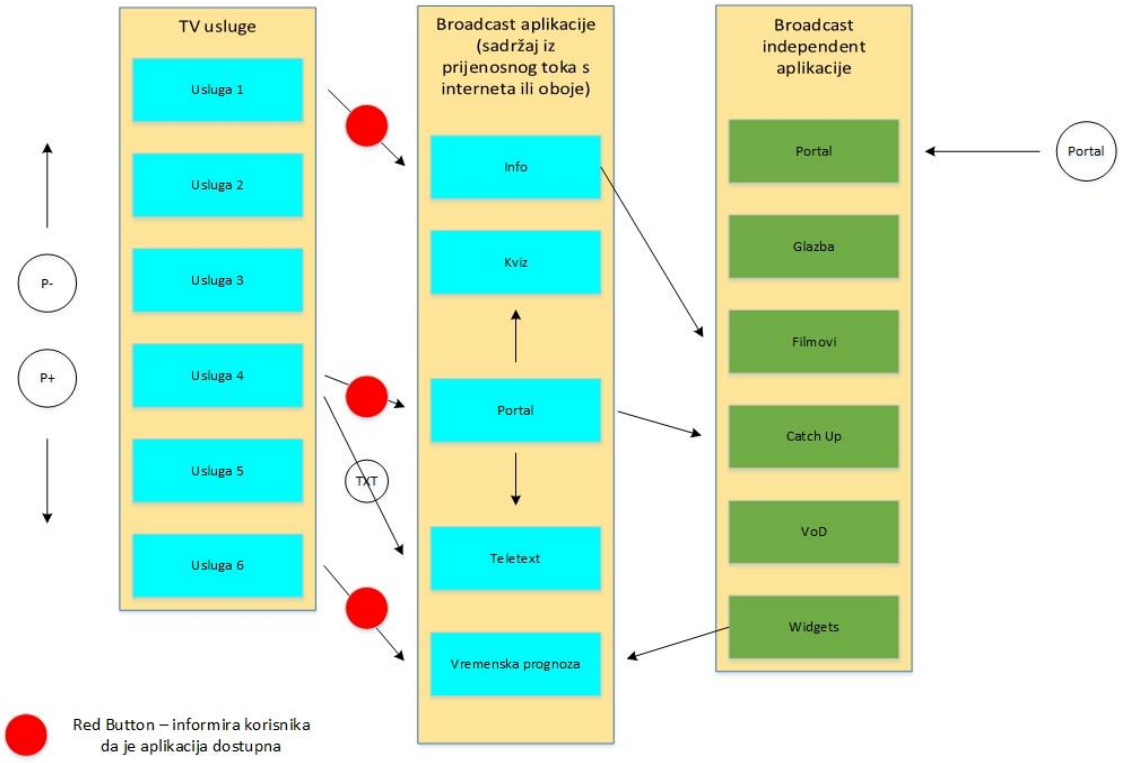
2.2 HbbTV

HbbTV je industrijski standard (ETSI TS 102 796) i inicijativa kako bi se omogućila realizacija hibridnih, interaktivnih DTV aplikacija koje za svoj rad koriste emitirani prijenosni tok (*broadcast*) i vezu sa internetom (*broadband*). Jedna od osnovnih karakteristika HbbTV-a je da je zasnovan na postojećim Web standardima i tehnologijama.



Slika 2.3 HbbTV s ostalim standardima i tehnologijama[3]

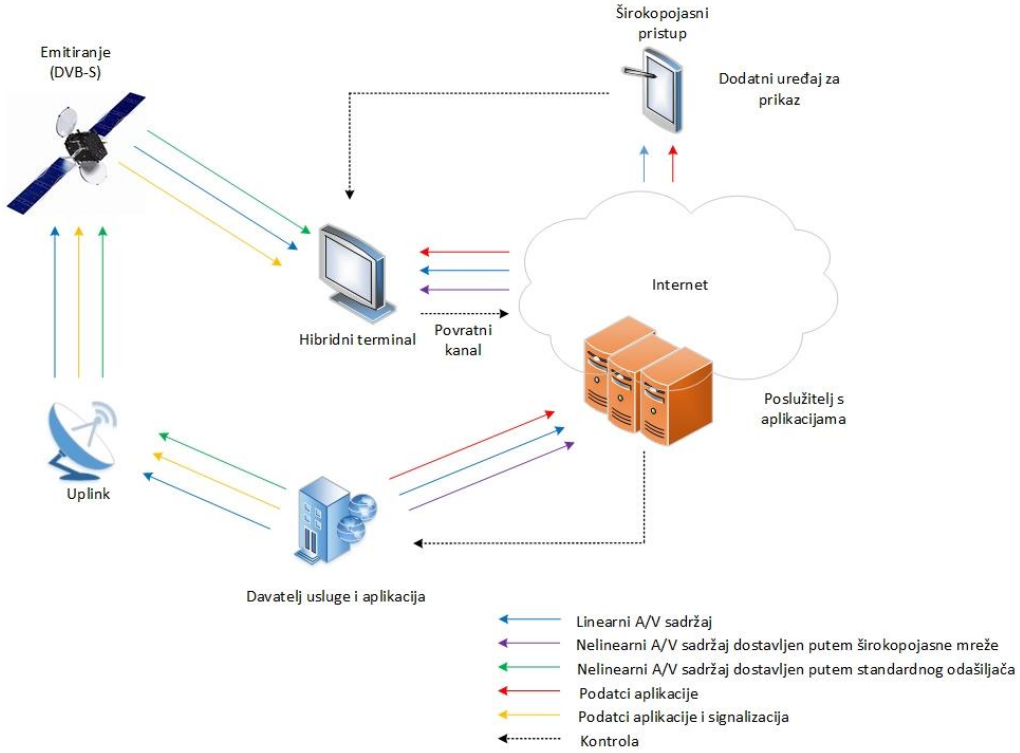
Za razliku od MHEG-a (*Multimedia and Hypermedia Experts Group*) koji je jedan od najčešće korištenih jezika za opis interaktivnih TV usluga (posjeduje svoj vlastiti jezik te ograničenu grupu ciljanih programera), HbbTV za svoje korištenje tipično traži DTV (*Digital television*) prijemnik s IP vezom, te se oslanja na tehnologije koje su vrlo poznate programerima za web (HTML, CSS, JavaScript). Korištenje tih tehnologija poboljšava brzinu razvoja novih HbbTV aplikacija i lakše povezivanje s postojećim sadržajima. HbbTV obuhvaća usluge kao što su poboljšani teletext, *catch-up* usluge, VOD (*Video on Demand*), EPG (*Electronic Program Guide*), interaktivni oglasi, društvene mreže, glasanja, igrice i druge multimedijske aplikacije. Na slici 2.4 možemo vidjeti neke od primjera HbbTV aplikacija [3].



Slika 2.4 Primjeri HbbTV aplikacija[3]

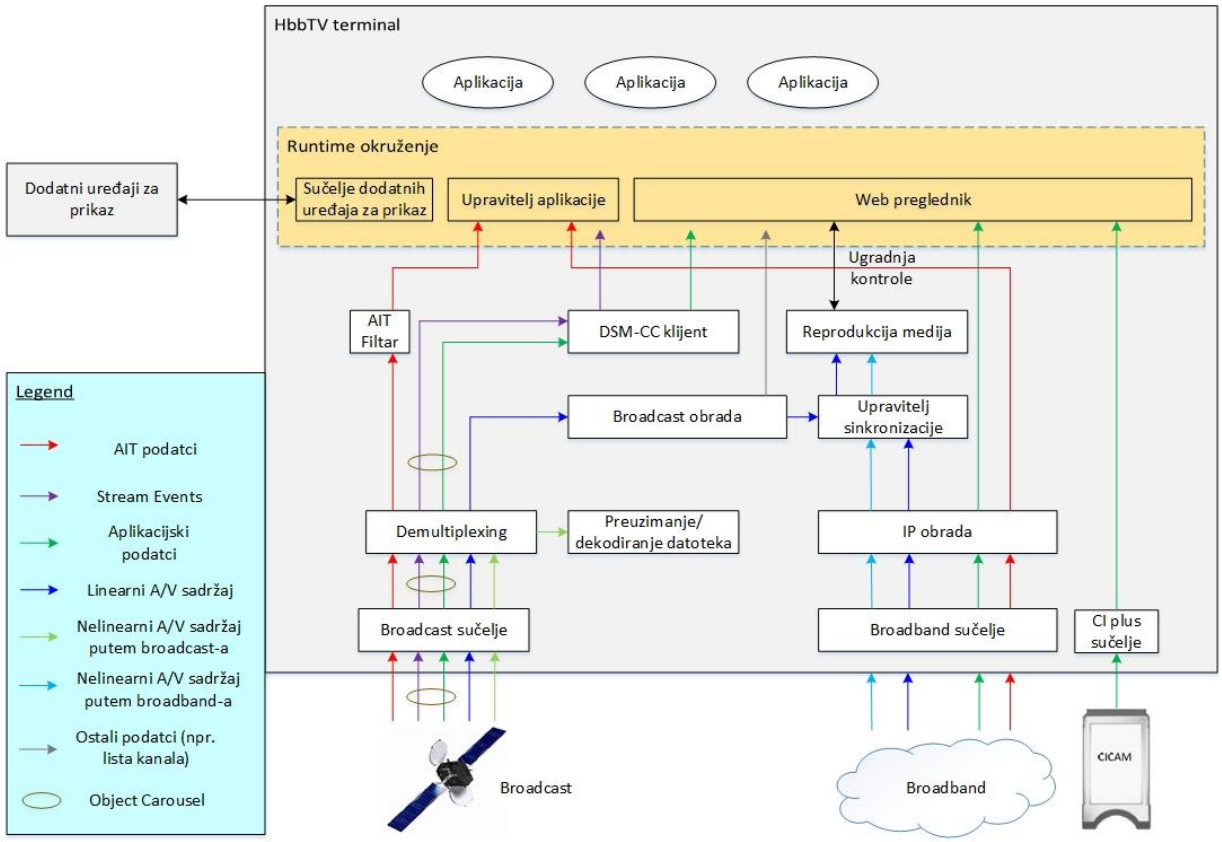
2.4 Pregled HbbTV sustava

Hibridni terminal ima mogućnost da bude spojen na dvije mreže u paraleli. Na jednoj strani može biti spojen na *broadcast* DVB (*Digital Video Broadcasting*). Kroz ovu *broadcast* konekciju, hibridni terminal može primiti standardni *broadcast* A/V sadržaj, A/V sadržaj koji nije u stvarnom vremenu (*real-time*), podatke od aplikaciji i signalne informacije o aplikaciji. Čak i ako terminal nije spojen na *broadband*, njegova konekcija s *broadcast* mrežom mu omogućava da primi aplikacije vezane uz *broadcast*. Hibridni terminal također može biti spojen na Internet putem *broadband* sučelja. To nam omogućuje dvosmjernu komunikaciju s davateljem usluga. Preko ovog sučelja terminal može primati podatke o aplikacijama, i nelinearni A/V sadržaj. Hibridni terminal također podržava prijenos datoteka (*download*) A/V sadržaja koji nije u stvarnom vremenu preko ovog sučelja. *Broadband* sučelje se može povezivati i sa dodatnim/pomoćnim uređajima za prikaz (*Companion Screen Device*) i drugim HbbTV terminalima na istoj lokalnoj mreži. Na slici 2.5 dan je pregled HbbTV sustava gdje je korišten DVB-S kao primjer *broadcast* konekcije [3].



Slika 2.5 Pregled HbbTV sustava

Slika 2.6 prikazuje glavne funkcionalne komponenti unutar hibridnog terminala gdje je svaka dalje u radu detaljnije opisana.



Slika 2.6 Funkcionalne komponente unutar hibridnog terminala

Preko *broadcast* sučelja, terminal prima AIT (*Application Information Table*) podatke, linearni A/V sadržaj, A/V sadržaj (koji nije u stvarnom vremenu), podatke od aplikaciji te neke događaje medijskog toka (*stream event*). Zadnja dva podatkovna toka se prijenose koristeći DSM-CC (*Digital Storage Media Command and Control*) *object carousel*. Sadržaj koji nije u stvarnom vremenu se prijenosi pomoću FDP protokola (*File Download Protocol*). Imajući to na umu, potrebna su dva dekodera kako bi se dobili podatci iz *object carousela*. Dobiveni podaci se dalje prosljeđuju *runtime* okruženju-u (sastoji se od web preglednika, upravitelja aplikacijama i sučelja dodatnih uređaja za prikaz). Upravitelj aplikacija ocjenjuje AIT da bi kontrolirao životni ciklus interaktivne aplikacije. Web preglednik je odgovoran za prikaz i izvršavanje interaktivnih aplikacija.

Linearni A/V sadržaj je obrađen na isti način kao i kod ne-hibridnih DVB terminala. Ovo je uključeno u funkcionalnu komponentu *broadcast* obrade koja obuhvaća sve DVB funkcionalnosti obuhvaćene na ne-hibridnom DVB terminalu. Dodatno neke informacije i funkcije iz *broadcast* obrade mogu biti pristupne od strane *runtime* okruženja (npr. informacije o

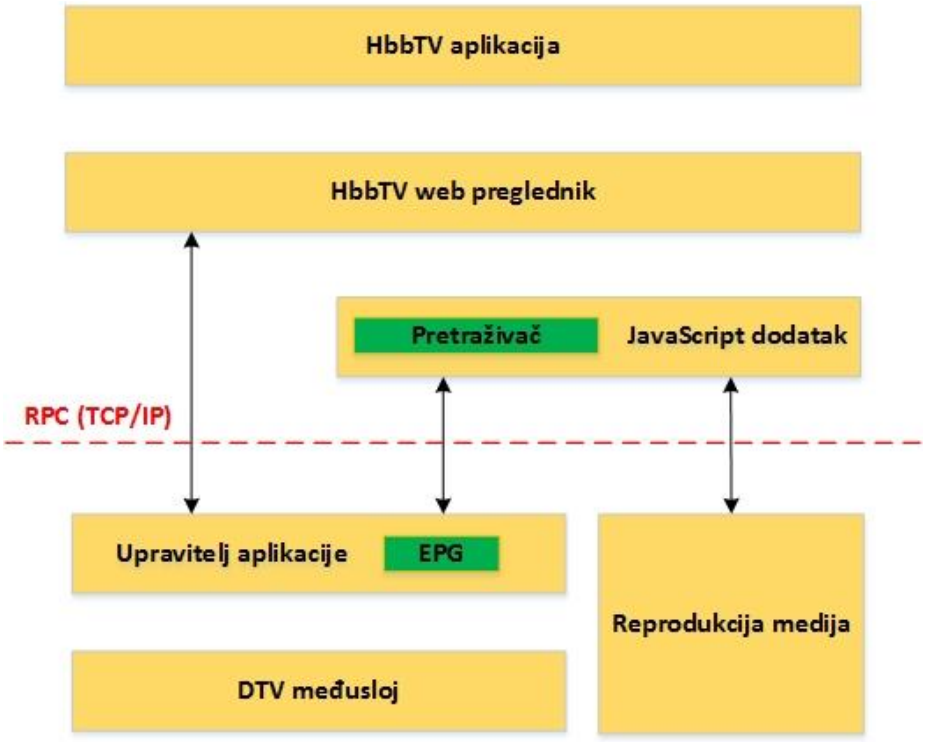
listi kanala ...). One su uključene u ostale podatke na slici 2.6. Štoviše, aplikacija može skalirati i ugraditi linearan A/V sadržaj u korisničko sučelje dobiveno od aplikacije. Ove funkcionalnosti su dane od strane reprodukcije medija. Na slici 2.6 to uključuje sve funkcionalnosti vezane uz obradu A/V sadržaja.

Pomoću *broadband* sučelja hibridni terminal ima konekciju s Internetom kojim dobiva drugi način da traži podatke sa servera ili od poslužitelja. Ta konekcija se također koristi za primanje A/V sadržaja (npr. sadržaj na zahtjev). Komponenta IP obrade sadrži sve komponente dobivene od terminala kako bi rukovala podacima koje dobiva s Interneta. Dalje se kroz ovu komponentu podaci aplikacije prijenose *runtime* okruženju. A/V sadržaj se prosljeđuje reprodukciji medija koji može biti kontroliran od strane *runtime* okruženja i time biti ugrađen u korisničko sučelje dobiveno od aplikacije. Zajedno s reprodukcijom medija, upravitelj sinkronizacije može sinkronizirati sadržaj dostavljen hibridnom terminalu pomoću *broadband* sučelja.

Sučelje dodatnih uređaja za prikaz omogućuje hibridnom terminalu da pronađe dodatne uređaje za prikaz i obrnuto. Kroz njega, interaktivne aplikacije koje se izvršavaju u pregledniku mogu zatražiti da aplikacija bude instalirana ili pokrenuta na dodatnom uređaju za prikaz ili obrnuto (aplikacija pokrenuta na dodatnom uređaju za prikaz može zatražiti od preglednika da pokrene neku aplikaciju). U kombinaciji, sučelje dodatnih uređaja za prikaz i reprodukcija medija zajedno omogućuju sinkronizaciju sadržaja dostavljenog na hibridni terminal kroz sučelje s sadržajem dostavljenim na dodatni uređaj za prikaz ili neki drugi hibridni terminal.

2.4.1 Arhitektura HbbTV sustava

Na slici 2.7 je prikazan HbbTV softverski stog. DTV međusloj parsira DVB tablice (uključujući i AIT) i obavještava upravitelja aplikacija.



Slika 2.7 Jednostavna arhitektura HbbTV sustava

HbbTV aplikacije su prikazane koristeći HbbTV web preglednik i one komuniciraju s DTV softverskim stogom putem JavaScript dodataka. JavaScript dodaci su razvijeni kao dio HbbTV stoga te podržavaju ugrađene HbbTV objekte : *broadcast*, audio/video, upravitelj aplikacija, roditeljska zaštita ...

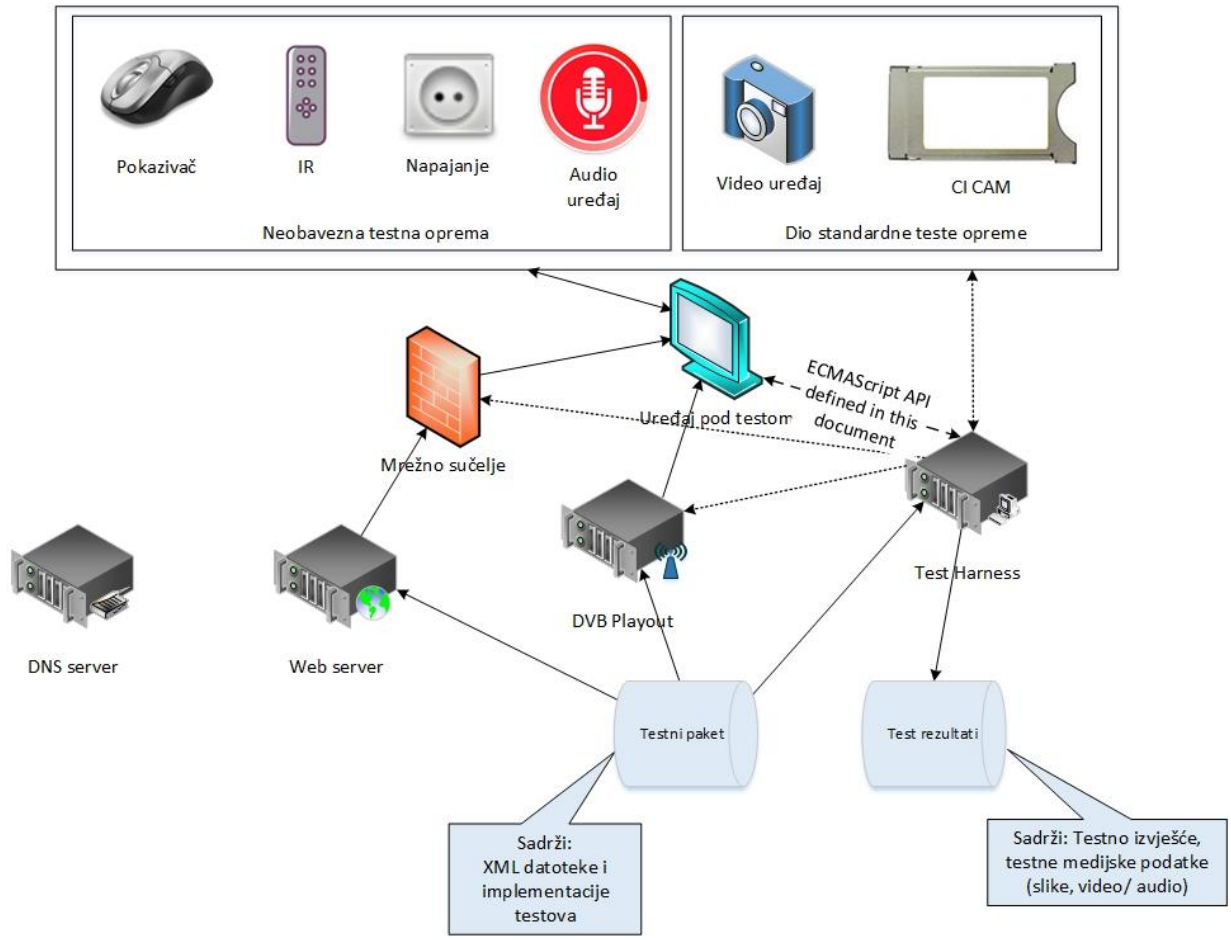
2.5 Sustav za testiranje

HbbTV testni sustav sastoji se od :

1. Skupa odobrenih testnih slučaja, dobivenih i održavanih od strane HbbTV zajednice.
2. Testnog okruženja korištenog da se izvrše testni slučajevi i zabilježe rezultati testiranja sadrži se od :
    a. Uređaja pod testom (*DUT - Device Under Test*).
    b. Test Harness-a koji služi kao testni operator te upravlja testnim slučajevima i rezultatima izvršavanja.
    c. Standardne testne opreme (*Standard Test Equipment*).
    d. Opcionalne testne opreme (*Optional Test Equipment*).

e. RF i IP konekcije između uređaja pod testom i ostalih elemenata testnog okruženja.

Na slici 2.7 vidimo međusobnu povezanost između ovih potrebnih elemenata.



Slika 2.8 Pregled jednog testnog sustava

2.5.1 Skup testova

Skup testova u HbbTV-u sastoji se od testnih slučajeva. Svaki test je jedinstveno označen i namijenjen da provjerava specifičan zahtjev iz HbbTV specifikacije. Testni slučaj se sastoji od :

- XML-a (*eXtended Markup Language*) testnog slučaja - to je višenamjenski dokument koji sadrži :

    - Specifikacijske reference
    - Proceduru i očekivane rezultate
    - Povijest izmjena (razvojne verzije) testnih slučaja
- Konfiguracijskih datoteka koje su potrebne za izvršavanje testnog slučaja na Test Harnessu
- Testnih implementacijskih dokumenata (gdje više testnih slučaja može koristiti iste dokumente koji su dijeljeni)
- Informacije o licencama

Testni slučajevi u setu testova mogu imati jedan od dva statusa :

- Testni slučajevi odobreni od strane HbbTV testirajuće grupe formiraju dio zahtjeva za HbbTV usklađenost
- Testni slučajevi koji nisu odobreni od strane HbbTV testirajuće grupe. Oni su tada dani na korištenje razvojnom timu (*developer*-ima), te mogu biti odobreni ukoliko u budućnosti zadovolje HbbTV zahtjeve i razmotreni su od strane HbbTV testirajuće grupe.

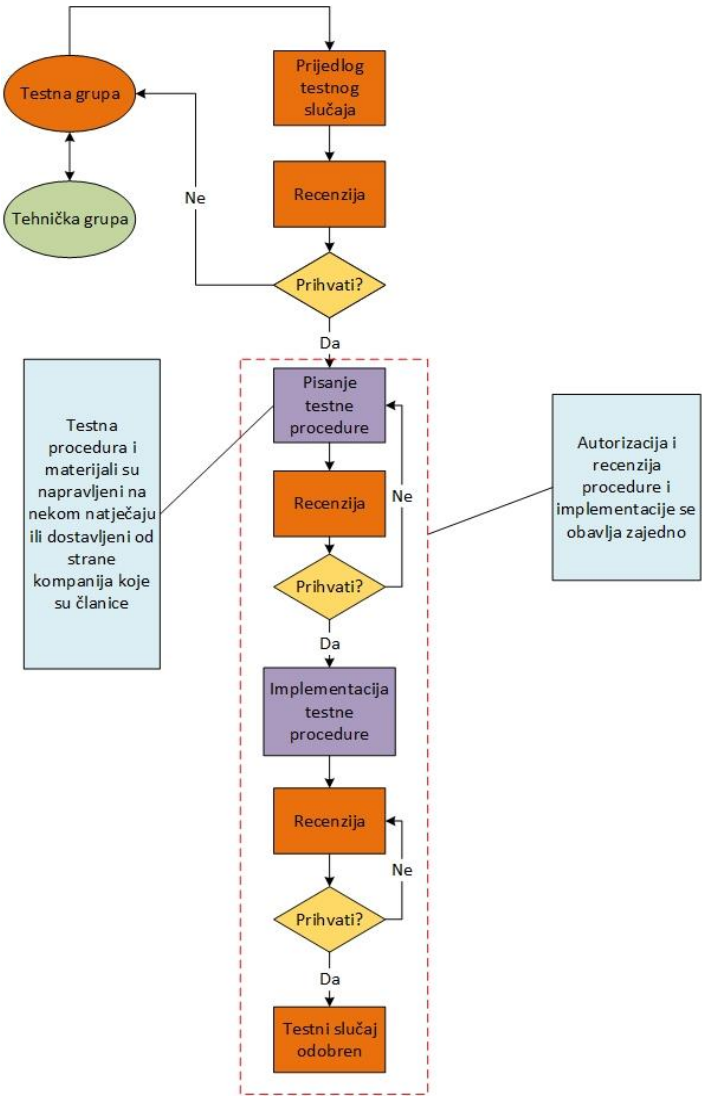
2.5.2 Testno okruženje

Testno okruženje za izvršavanje HbbTV seta testova na DUT-u sastoji se od dvije glavne komponente :

- Standardne testne opreme - to je skup svih alata koji su potrebni za pohranu, posluživanje, generiranje i puštanje testnih slučajeva na DUT-u. U to spadaju i web serveri i DVB *playout* medijski tok. Komponente standardnog testnog okruženja nisu uključene s setom testova, ali se mogu koristiti komercijalno dostupni alati.
- Test Harness - je sustav koji upravlja selekcijom i izvršenjem testnih slučajeva na DUT-u te s prikupljanjem rezultata i izvještaja testnih slučajeva. Test Harness:
    - Koristi informacije iz opisa HbbTV testnog slučaja i iz testnih materijala kako bi izvršio sve potrebne koraka prije izvršenja.
    - Započinje izvršenje testnog slučaja na DUT-u s time da pravi promjene u okruženju koje se temelje na vremenskim točkama koje su definirane u testnim materijalima i u odgovoru na API poziv iz testnog slučaja.
    - Sakuplja rezultate testnog slučaja koji se vrti na DUT-u

3. SPECIFIKACIJE I PROCES STVARANJA TESTNIH SLUČAJEVA

HbbTV testni slučajevi se temelje na proizvoljnim i obveznim zahtjevima koji su definirani u HbbTV tehničkoj specifikaciji. Testni slučajevi su vođeni i predloženi od strane HbbTV testirajuće grupe. Proces definiranja, implementacije i prihvaćanja HbbTV testnog slučaja je prikazan u koracima na slici 3.1.



Slika 3.1 Proces stvaranja testova

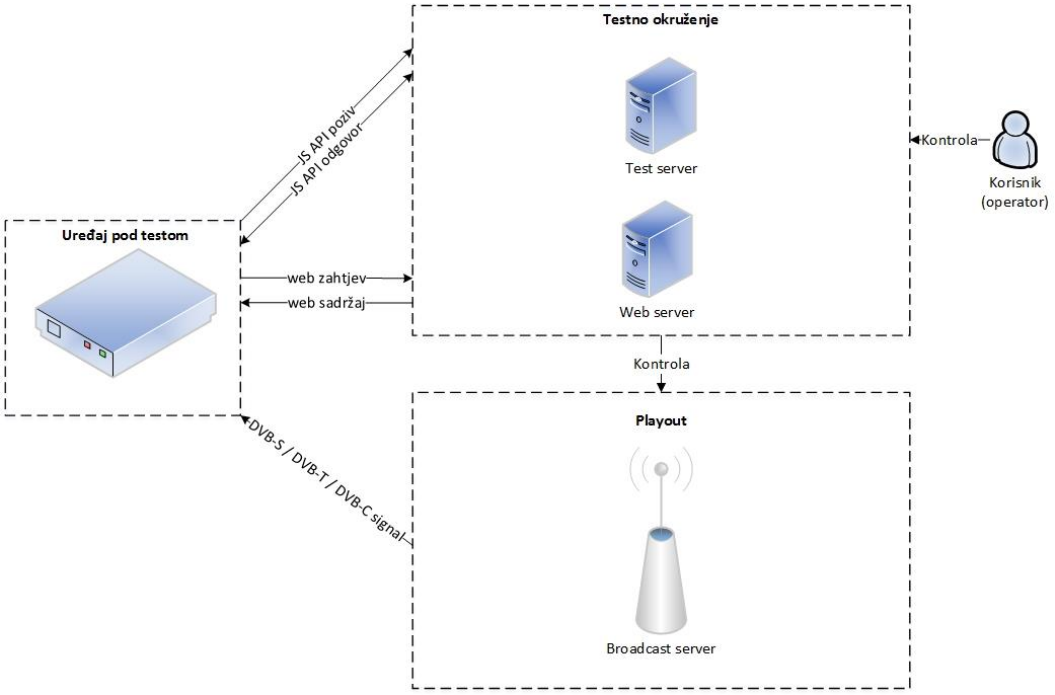
Kada je test odobren te izvršen od strane HbbTV testirajuće grupe, rezultat će mu imati vrijednost “PASS” ukoliko su zadovoljeni sljedeći kriteriji :

1. Svi testni preduvjeti su zadovoljeni
2. Svi rezultati pohranjeni od test Harness-a daju vrijednost “*true*”
3. Svi pozivi za analiziranje API metode su ocijenjeni i daju vrijednost “*true*”
4. Svi pozivi ka test API metodi koji imaju vezu i s testnim okruženjem moraju uspjeti
5. “*endTest*” API metoda je pozvana

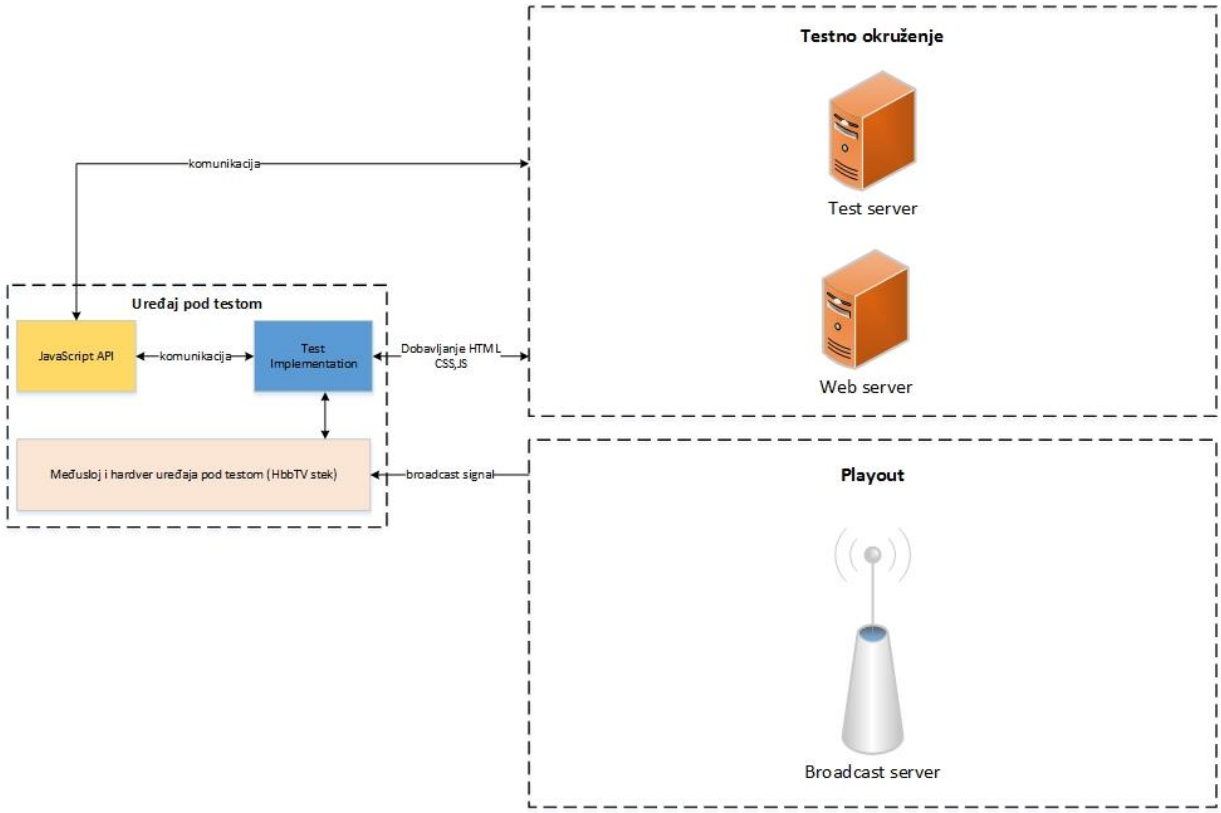
4. TEST API I DEFINICIJA PLAYOUT SET-A

Ovdje ćemo prikazati/opisati dva sučelja :

1. JavaScript API koji definira sučelje između Test Harness-a i DUT-a
2. Skup XML datoteka koje opisuju kako Test Harness interpretira testni slučaj. To nam dozvoljava definiranje i kontrolu DVB *playout*-a potrebnu za inicijaciju testa.



Slika 4.1 Pregled JavaScript API komunikacije između DUT-a i Test Harnessa



Slika 4.2 Detaljni prikaz JavaScript API apstrakcije Test Harness komunikacije

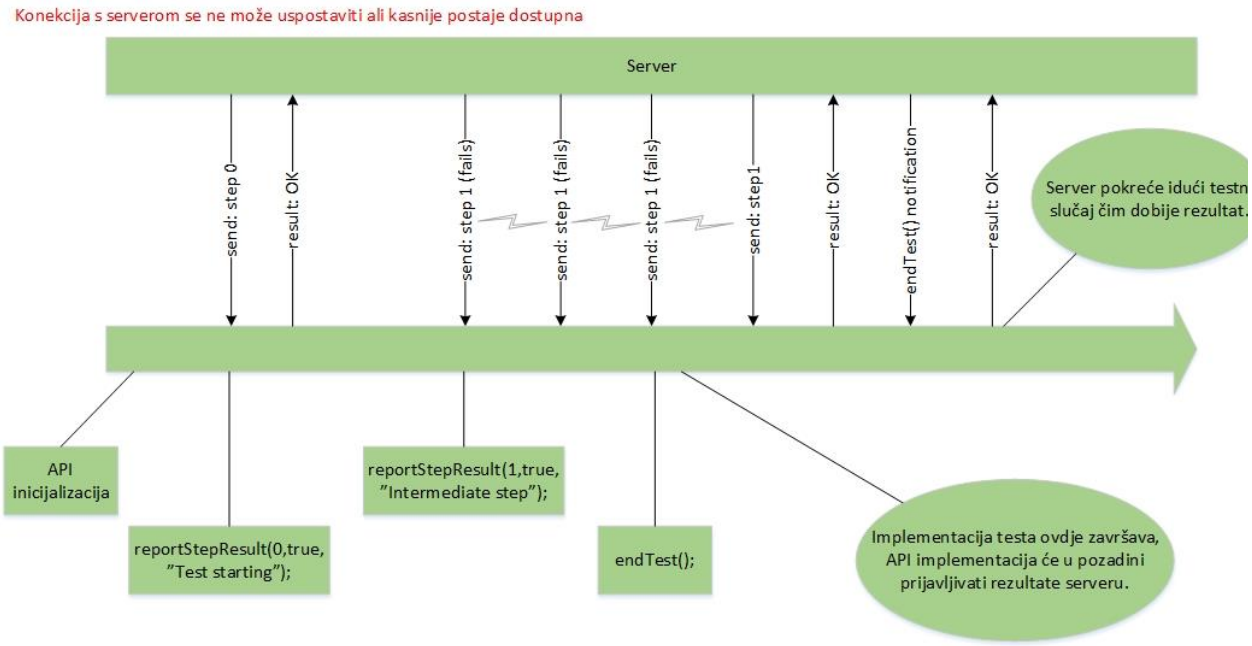
Raspored API-a koji je opisan, napravljen je tako da dozvoljava visok postotak automatizacije (nema potrebe za HbbTV Test okruženje da bude posve automatizirano). HbbTV JavaScript API-i su zbog toga dizajnirani kako bi mogli ručno upravljati testom. Postoji mogućnost višestruke implementacije pojedinog testa što dozvoljava kombinaciju testnih slučaja kako bi se napravilo kompletno HbbTV (automatizirano) testno okruženje. JavaScript API je podijeljen u 3 dijela :

1. API-i koji komuniciraju s testnim okruženjem. Oni obavještavaju testno okruženje o trenutnom statusu testa.
2. API-i koji komuniciraju s DUT-om. Ovaj dio može biti implementiran od strane DUT proizvođača ili od nekog drugog.
3. API-i koji komuniciraju s *playout* okruženjem. *Playout* okruženje je odgovorno za slanje ispravne AIT tablice DUT-u kako bi se test mogao pokrenuti na tom DUT-u.

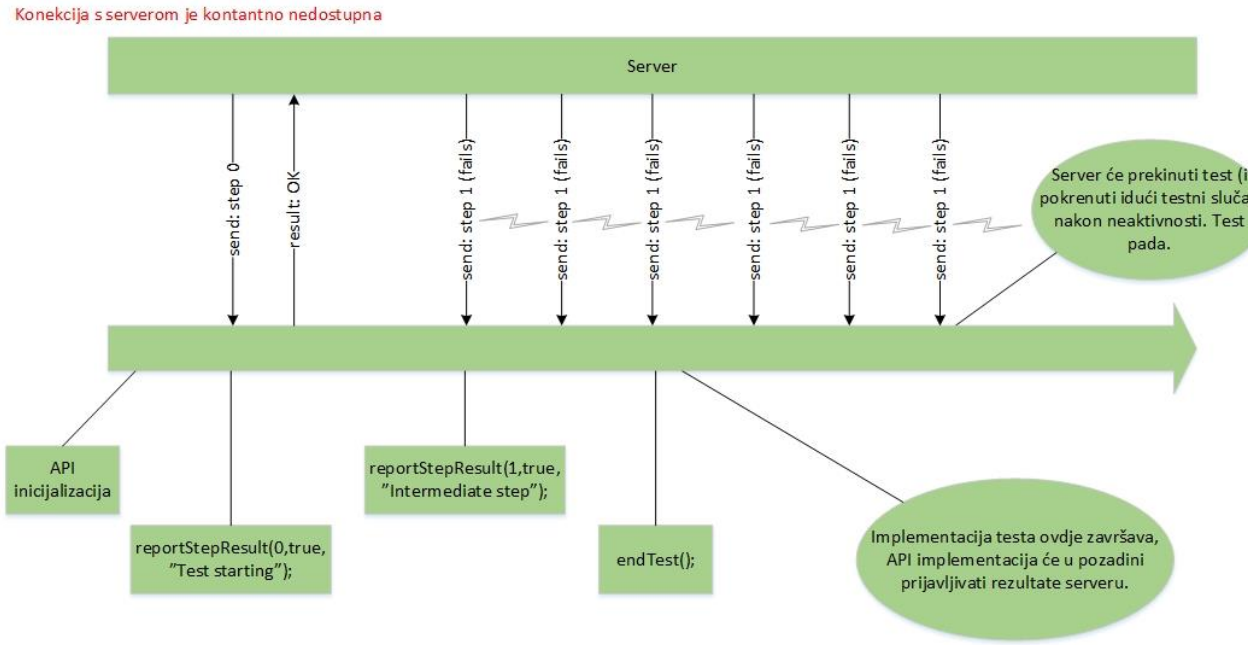
Dalje je dan prikaz nekih od JavaScript funkcija koje se koriste u pojedinom JavaScript API-u te je jedna od njih detaljnije pojašnjena (*reportStepResult*).

JavaScript funkcije API-a koji komuniciraju s testnim okruženjem	
Ime funkcije	Kratak opis funkcije
getPlayoutInformation()	funkcija vraća informacije o trenutnom *playout*-u
endTest()	funkcija ukazuje na to da je testni slučaj završen
reportMessage()	funkcija vraća poruke s informacijama ka Test Harness-u
waitForCommunicationCompleted()	funkcija provjerava da li su svi pozivi uspješno dostavljeni serveru
manualAction()	funkcija nalaže test operatoru da izvrši akciju

Također još jedna JavaScript funkcija API-a koja komunicira s testnim okruženjem je i *reportStepResult()*. Ova funkcija daje izvješće o rezultatima pojedinog koraka nazad k test Harness-u. Test Harness prijavljuje *stepId*, rezultat i komentare u testnom izvješću. Ona bi se trebala koristiti u svakom važnijem koraku unutar testa kako bi provjerili/prijavili njihove rezultate. Dan je primjer gdje imamo asinkronu komunikaciju sa serverom gdje su svi zahtjevi spremljeni u FIFO (*First In First Out*) red. Ukoliko komunikacija nije moguća (mreža nije dostupna) JavaScript API implementacija će stalno pokušavati da prijavi rezultate pojedinih koraka redu u pozadini. Zato dizajner testova mora biti siguran da će mrežna konekcija biti dostupna na kraju testa. Iduća dva dijagrama (slika 4.3 i 4.4) prikazuju komunikaciju s serverom u slučaju asinkrone komunikacije.



Slika 4.3 Komunikacija između DUT-a i Test Harness-a kada je mreža privremeno nedostupna



Slika 4.4 Komunikacija između DUT-a i Test Harness-a kada je mreža konstantno nedostupna

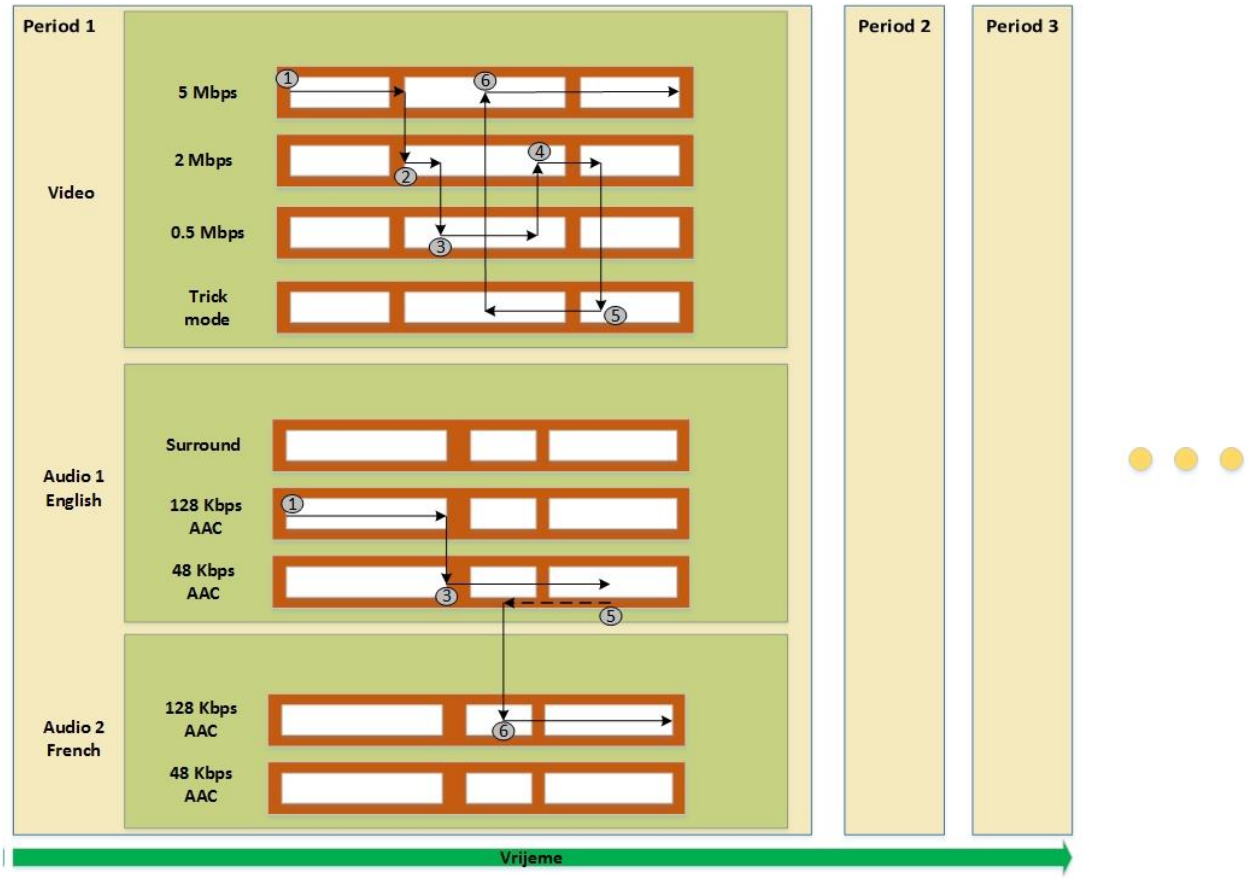
JavaScript funkcije API-a koji komuniciraju s DUT-om	
Ime funkcije	Kratak opis funkcije
initiatePowerCycle()	funkcija pokreće "*power cycle*" (paljenje gašenje u danim intervalima) DUT-a
sendKeyCode()	funkcija zahtjeva "*key code*" (ID tipke) koji ce biti poslan DUT-u
analyzeScreenPixel()	funkcija analizira trenutni zaslon i provjerava da li traženi element slike odgovara danoj referentnoj boji
analyzeScreenExtended()	funkcija analizira trenutni zaslon i obavlja detaljnu provjeru na trenutnom sadržaju zaslona
analyzeAudioFrequency()	funkcija analizira trenutni audio i obavlja frekvencijsku provjeru nad tim podacima
analyzeAudioExtended()	funkcija analizira trenutni audio i obavlja detaljnu provjeru nad tim podacima
analyzeVideoExtended()	funkcija analizira trenutni video i obavlja detaljnu provjeru nad tim podacima
analyzeManual()	funkcija naređuje test operatoru da uradi analizu opisanu u danim parametrima i snimi rezultate
selectServiceByRemoteControl()	funkcija zahtjeva odabir usluge u odnosu na dobivenoj sekvenci "*key code*-ova"

sendPointerCode()	funkcija zahtjeva od pokaznog uređaja da se pomjeri na određenu lokaciju zaslona DUT-a
moveWheel()	funkcija zahtjeva od uređaja s točkićem da se pomjeri relativno u odnosu na trenutni položaj

JavaScript funkcije API-a koji komuniciraju s *playout* okruženjem	
Ime funkcije	Kratak opis funkcije
changePlayoutSet()	funkcija mijenja trenutni *playout* (time može onemogućiti mrežnu konekciju)
setNetworkBandwidth()	funkcija ograničava maksimum podataka aplikacije dozvoljenih na mrežnom sučelju uređaja pod testom.

5. MPEG DASH

MPEG DASH je standard za isporuku videa, te način kako možemo dostaviti *live stream* putem Interneta. Cilj mu je najveća moguća kvaliteta sadržaja s što manje gubitaka i najmanjim mogućim među-spremnikom (*buffer*-om). MPEG DASH koristi sličan prilagodni datotečni sustav kao i ostali protokoli za strujanje medija. Prvo je sama video datoteka rastavljena na dijelove koji su duplicirani te kodirani različitim brzinama prijenosa. Drugo, manifest datoteka pomaže video uređaju da odredi gdje se nalaze pojedini djelići video datoteka (segmenti). Ti segmenti se dalje šalju preko mreže, gdje se u ovisnosti o korisniku (propusnosti mreže) odlučuje brzina prijenosa.

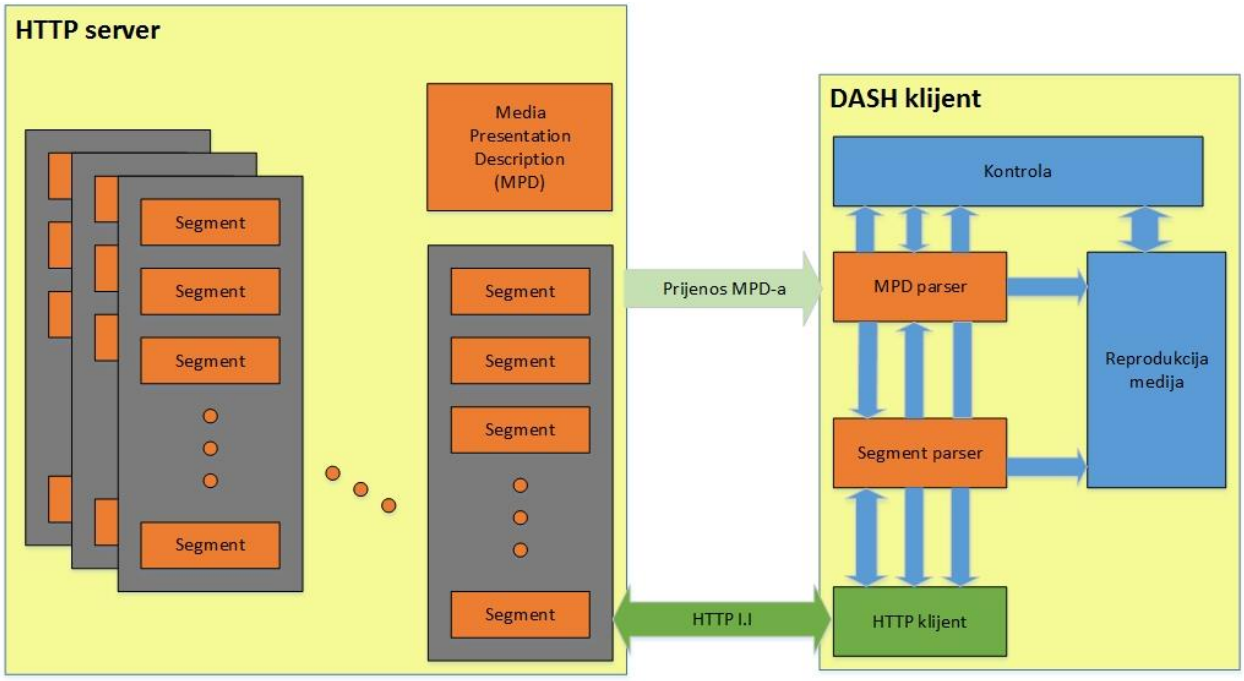


Slika 5.1 Primjer dinamičkog prilagodljivog strujanja[4]

Na slici 5.1 dan je jednostavan primjer *On Demand* dinamičkog prilagodljivog strujanja. Uočimo da se sadržaj sastoji od video i audio komponente. Video komponenta je kodirana različitim brzinama prijenosa: 5 *Mbps* (megabajta u sekundi), 2 *Mbps*, 0.5 *Mbps* te *trick* način koji predstavlja premotavanje sadržaja naprijed nazad. Pridružujuće audio komponente su dane u dva jezika, engleskom i francuskom te su kodirane s dvije brzine 128 *Kbps* AAC (*Advanced*

Audio Coding) te 48 *Kbps* AAC. Označeni krugovi predstavljaju akcije koje uređaj zahtjeva (mogućnost dobavljanja sadržaja na osnovu performansi mreže s klijent strane).[4]

MPEG DASH ima dvije glavne prednosti u odnosu na ostale protokole za strujanje medija. Prva je ta da je baziran na HTTP-u, web server tehnologiji koja je podloga cijelog web sadržaja, a druga da za MPEG DASH raste podrška, u nastojanju da se razvije jedan univerzalni standard za video strujanje.



Slika 5.2 Opseg MPEG DASH standarda[4]

Slika 5.2 prikazuje jednostavan scenarij strujanja između HTTP servera i DASH klijenta. Sadržaj je podijeljen u segmente, sačuvan na HTTP serveru te dostavljen putem HTTP-a. Da bi DASH klijent reproducirao sadržaj, dobavlja MPD te ga parsira (saznaje vremena segmenata, tip medija, rezoluciju, minimalni i maksimalni propusnost mreže ...). Koristeći dobivene informacije DASH klijent odabire odgovarajuću brzinu prijenosa i započinje strujanje dobavljanjem segmenata putem HTTP GET zahtjeva. U ovisnosti o propusnosti same mreže, klijent prilagođava kvalitetu dobavljenih segmenata (veća ili manja brzina prijenosa). MPEG DASH specifikacija opisuje samo MPD i formate segmenata. Dostavljanje MPD-a i segmenata, kao i ponašanje klijenta za dobavljanje, prilagođenje brzine prijenosa te reprodukcije sadržaja su izvan MPEG DASH opsega standarda.[4]

5.1 DASH manifest – MPD

MPEG DASH MPD je XML dokument koji sadrži informacije o medijskim segmentima, njihovim relacijama te podatcima, kako bi se omogućilo kretanje između njih, i ostale meta-podatke koji mogu biti potrebni klijentu. Opisani su najvažniji dijelovi (oznake ili *tag*-ovi) MPD-a, krećući od gornje razine (*Period*) prema dolje (*Segment*).

5.1.1 Period

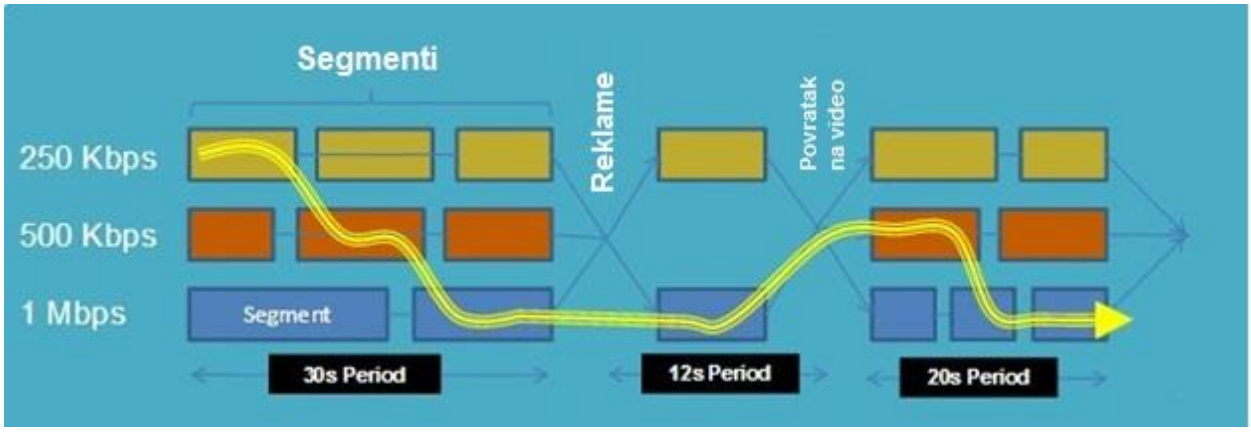
Period koji se nalaze na najvišoj razini MPD-a, opisuje dio sadržaja sa atributom početka te atributom trajanja. Postoji mogućnost korištenja višestrukih pojavljivanja Period *tag*-a npr. za neke scene, poglavlja, ili da se odvoje reklame od pravog sadržaja.[5]

5.1.2 AdaptationSet

*AdaptationSe*t-ovi sadrže medijski podatkovni tok ili set medijskih podatkovnih tokova. U najjednostavnijem slučaju Period može imati jedan *AdaptationSet* koji sadrži sav audio i video sadržaj, ali da bi se smanjila potrebna propusnost mreže, svaki tok može biti podijeljen u različite *AdaptationSet*-ove. Najčešći primjer je da imamo jedan video *AdaptationSet* i više audio *AdaptationSet*-ova (svaki za različiti jezik). *AdaptationSet*-ovi također mogu sadržavati prevode (*subtitle*) ili metapodatke.[5]

5.1.3 Representation

Representation-i omogućuju *AdaptationSet*-u da u sebi ima isti sadržaj ali različito kodiran. Najčešće imamo *Representation tag* s više različitih veličina zaslona i propusnosti mreže. Ovo omogućuje korisniku da zatraži najveću kvalitetu sadržaja koju može gledati bez ikakvog trzanja (učitavanja u među-spremnik/*buffering*) i bez ne iskorištenja propusnosti mreže na nepotrebnim elementima slike (720p TV ne treba 1080p sadržaj). *Representation* također može biti različito kodiran, kako bi omogućio podršku za klijente koji podržavaju neku drugu vrstu kodiranja. Mogućnost višestrukih *codec*-a može biti korisno i za uređaje koji se napajaju putem baterije, gdje uređaj može odabrati stariji *codec* jer je recimo potrošnja baterije manja.[5]



Slika 5.4 Prikaz procesa s klijentske strane[6]

Proces se sastoji od:

- Klijent skida/*download*-a MPD datoteku.
- Klijent *download*-a segment po segment na temelju *playout* procesa.
- Klijent odlučuje brzinu prijenosa

Faktori koji su bitni za odabir *Representation*-a su:

- Stanje među-spremnika
- Stanje mreže
- Kada korisnik odabire drugu rezoluciju npr. *full screen*
- Aktivnost uređaja i resursi

5.1.4 MediaSegment

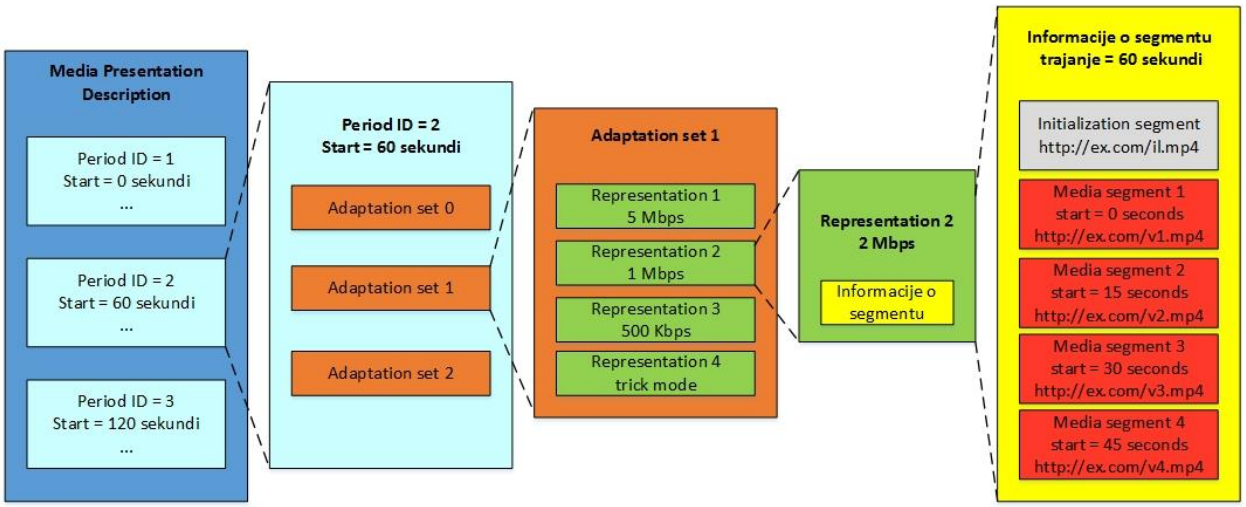
MediaSegment-i predstavljaju stvarne medijske datoteke koje DASH klijent pušta, puštajući ih jednu za drugom kao da se radi o jednoj datoteci (iako se stvari znaju zakomplicirati kada se rade prijelazi između reprezentacija). Stvarna lokacija tih datoteka može biti opisana putem *BaseURL* tag-a za jedno segmentnu reprezentaciju, liste segmenata *SegmentList* ili putem predloška *SegmentTemplate*. Informacije koje se odnose na sve segmente mogu se pronaći u *SegmentBase*. Početak i trajanje segmenata mogu biti opisani *SegmentTimeline*-om (važno za strujanje uživo kako bi klijent mogao brzo odrediti najnoviji segment). Segmenti također mogu biti u posebnim datotekama ili mogu biti zapisani u rasponima bajtova unutar jedne datoteke. Segmenti mogu imati malo trajanje (≈1-10 s) ili dugo (≈10s-2 sata).[5]

Trajanje segmenata	Prednosti	Nedostatci
Kratko trajanje(2-10s)	• Pogodno za Live profil • On Demand omogućen s Live profilom • Visoka granularnost prebacivanja na segmentnom nivou	• Velik broj datoteka • Velik broj URL-ova • Fiksna veličina zahtjeva • Granularnost prebacivanja na segmentnom nivou
Dugo trajanje(10s-2h)	• Mali broj datoteka • Mali broj URL-ova • Visoka granularnost prebacivanja • Fleksibilna veličina zahtjeva	• Potreba za indeksiranjem segmenata • Različito od Live profila

Slika 5.5 Prednosti i mane u zavisnosti od trajanja segmenata[6]

5.1.5 IndexSegment

IndexSegment-i dolaze u dva tipa: jedan *Representation IndexSegment* za čitav *Representation* ili jedan *IndexSegment* po svakom *MediaSegment*-u. *Representation IndexSegment* je uvijek zasebna datoteka, dok *IndexSegment* po svakom *MediaSegment*-u može biti raspon bajtova u istoj datoteci kao i *MediaSegment*. *IndexSegment*-i sadrže ISOBMFF (*ISO Base Media File Format*) „*sidx boxes*“ s informacijama o trajanju pojedinih medijskih segmenata (u bajtovima i vremenu), vrsti pristupnih točaka medijskom toku, ili pod-segmentne informacije u „*ssix boxes*“ (iste navedene informacije ali unutar segmenata). U slučaju *Representation IndexSegment*-a *ssix boxes* dolaze jedna za drugom ali im prethodi jedan *sidx*.[5]



Slika 5.6 Hijerarhijski model MPD-a[4]

Slika 5.6 prikazuje hijerarhijski podatkovni model MPD-a koji sadrži tri Period-a od kojih drugi period u sebi ima tri *AdaptationSet*-a, gdje *AdaptationSet1* sadrži četiri *Representation*-a, od kojih su tri kodirane različitom brzinom prijenosa dok je jedna „*trick mode*“ (premotavanje naprijed-nazad različitim brzinama). Konačno u *Representation2* nalaze se informacije o segmentima tj. početni/inicijalizacijski segment te četiri medijska segmenta.

Također je na slici 5.7 prikazan i stvarni zapis/struktura MPD-a, koji je dan u XML-u (standardni format), s svim dosad opisanim *tag*-ovima.

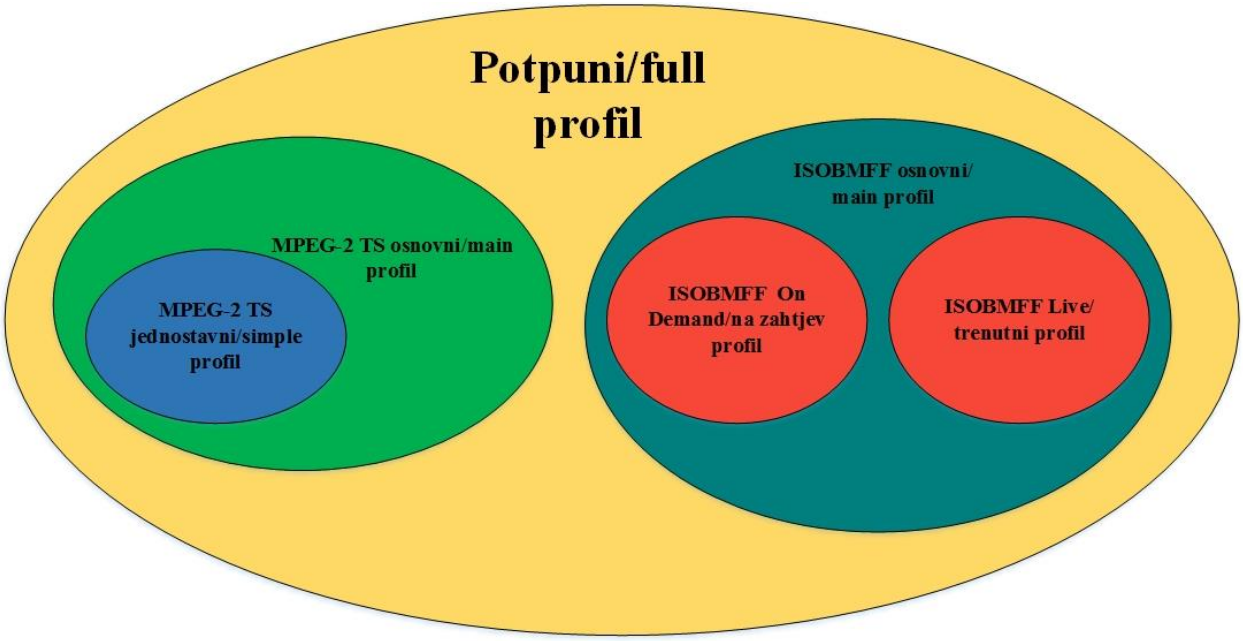
```xml
<?xml version="1.0"?>
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" profiles="urn:mpeg:dash:profile:full:2011" minBufferTime="PT1.5S">
    <!-- Ad -->
    <Period duration="PT30S">
        <BaseURL>ad/</BaseURL>
        <!-- Everything in one Adaptation Set -->
        <AdaptationSet mimeType="video/mp2t">
            <!-- 720p Representation at 3.2 Mbps -->
            <Representation id="720p" bandwidth="3200000" width="1280" height="720">
                <!-- Just use one segment, since the ad is only 30 seconds long -->
                <BaseURL>720p.ts</BaseURL>
                <SegmentBase>
                    <RepresentationIndex sourceURL="720p.sidx"/>
                </SegmentBase>
            </Representation>
            <!-- 1080p Representation at 6.8 Mbps -->
            <Representation id="1080p" bandwidth="6800000" width="1920" height="1080">
                <BaseURL>1080p.ts</BaseURL>
                <SegmentBase>
                    <RepresentationIndex sourceURL="1080p.sidx"/>
                </SegmentBase>
            </Representation>
        </AdaptationSet>
    </Period>
    <!-- Normal Content -->
    <Period duration="PT5M">
        <BaseURL>main/</BaseURL>
        <!-- Just the video -->
        <AdaptationSet mimeType="video/mp2t">
            <BaseURL>video/</BaseURL>
            <!-- 720p Representation at 3.2 Mbps -->
            <Representation id="720p" bandwidth="3200000" width="1280" height="720">
                <BaseURL>720p/</BaseURL>
                <!-- First, we'll just list all of the segments -->
                <!-- Timescale is "ticks per second", so each segment is 1 minute long -->
                <SegmentList timescale="90000" duration="5400000">
                    <RepresentationIndex sourceURL="representation-index.sidx"/>
                    <SegmentURL media="segment-1.ts"/>
                    <SegmentURL media="segment-2.ts"/>
                    <SegmentURL media="segment-3.ts"/>
                    <SegmentURL media="segment-4.ts"/>
                    <SegmentURL media="segment-5.ts"/>
                </SegmentList>
            </Representation>
        </AdaptationSet>
        <!-- Just the audio -->
        <AdaptationSet mimeType="audio/mp2t">
            <BaseURL>audio/</BaseURL>
            <!-- We're just going to offer one audio representation, since audio bandwidth isn't very
                 important. -->
            <Representation id="audio" bandwidth="128000">
                <SegmentTemplate media="segment-$Number$.ts" timescale="90000">
                    <RepresentationIndex sourceURL="representation-index.sidx"/>
                    <SegmentTimeline>
                        <S t="0" r="10" d="5400000"/>
                    </SegmentTimeline>
                </SegmentTemplate>
            </Representation>
        </AdaptationSet>
    </Period>
</MPD>
```

Slika 5.7 XML struktura MPD-a[5]

5.2 MPEG DASH profili

DASH profili predstavljaju set restrikcija na ponuđeni MPD i segmente, na njih se može gledati kao na dozvolu za DASH klijente da implementiraju samo svojstva koja su potrebna od

strane profila da bi mogli procesuirati MPD. Svi profili su definirani u ISO/IEC 23009 dokumentu te prikazani su i navedeni na slici 5.8.



Slika 5.8 MPEG DASH profili[6]

5.2.1 ISOBMFF On Demand profil

ISOBMFF *On Demand* (na zahtjev korisnika) profil je namijenjen da pruži osnovnu podršku *On Demand* sadržaju. Daje podršku za velike VoD biblioteke s minimalnom količinom upravljanja sadržaja. Dopušta skalabilno i učinkovito korištenje HTTP poslužitelja. Dan je unutar XML zapisa manifesta u MPD *tag*-u pod atributom „*profiles*“ gdje ima vrijednost „urn:mpeg:dash:profile:isoff-on-demand:2011“.Primarna ograničenja ovog profila su [6]:

- Uvjet da je svaki *Representation* dan kao jedan segment-
- Pod segmenti su usklađeni s *Representation-om* unutar *AdaptationSet-a*.
- Pod segmenti moraju započinjati s pristupnom točkom medijskog toka.

5.2.2 ISOBMFF Live profil

ISOBMFF *Live* profil je optimiziran za kodiranje uživo (*live*) i malo vrijeme čekanja na segmente, koji se sastoji od jednog *movie* fragmenta ISO formata s relativno malim trajanjem. Svaki *movie* fragment može biti zatražen kada je dostupan pomoću generiranog URL (*Uniform Resource Locator*) predloška, tako da nije potrebno zatražiti ažuriranje MPD-a prije svakog

zahtjeva za segmentom. Segmenti su ograničeni tako da mogu biti spojeni u nekim granicama, a dešifriraju se bez praznina između segmenata, tj. dolaze jedan za drugim bez čekanja, bez obzira na prilagodljivo prebacivanje između *Representation-a* unutar *AdaptationSet-a*. Iako je profil optimiziran za *live* usluge, ako je atribut u MPD *tag*-u (*Type*) postavljen na *static*, možemo prenositi i sadržaj koji nije *live*. Dan je unutar XML zapisa manifesta u MPD *tag*-u pod atributom *profiles* gdje ima vrijednost „*urn:mpeg:dash:profile:isoff-live:2011*“.[6]

5.2.3 MPEG-2 TS Main profil

Main profil nameće mala ograničenja na format medija segmenta za MPEG-2 sadržaj transportnog medijskog toka. Također ima mogućnost višestrukih *Representation-a* (nije potrebno naknadno spajanje), formati segmenata su rastavljeni tako da odgovaraju MPEG-2 granicama paketa. Preporučeno je da se obavi i indeksiranje i poravnavanje segmenata, te možemo integrirati HLS (*HTTP Live Streaming*) sadržaj s ovim profilom. Može se prepoznati unutar XML zapisa manifesta u MPD *tag*-u pod atributom *profiles* koji ima vrijednost „*urn:mpeg:dash:profile:mp2t-main:2011*“.[6]

5.2.4 MPEG-2 TS Simple profil

Simple profil je pod-set MPEG-2 *main* profila. Ima veća ograničenja na kodiranje i multipleksiranje sadržaja, kako bi omogućio jednostavnu implementaciju „*seamless switching*-a“. Profil se postiže tako što je zagarantirano da medija uređaj koji odgovara ISO/IEC 1318-1 može puštati bilo koji bit-*stream* generiran spajanjem uzastopnih segmenata iz bilo kojeg *Representation*-a unutar jednog *AdaptationSet*-a. Može se prepoznati unutar XML zapisa manifesta u MPD *tag*-u pod atributom *profiles* koji ima vrijednost „*urn:mpeg:dash:profile:mp2t-simple:2011*“.[6]

6. DASH KLIJENT

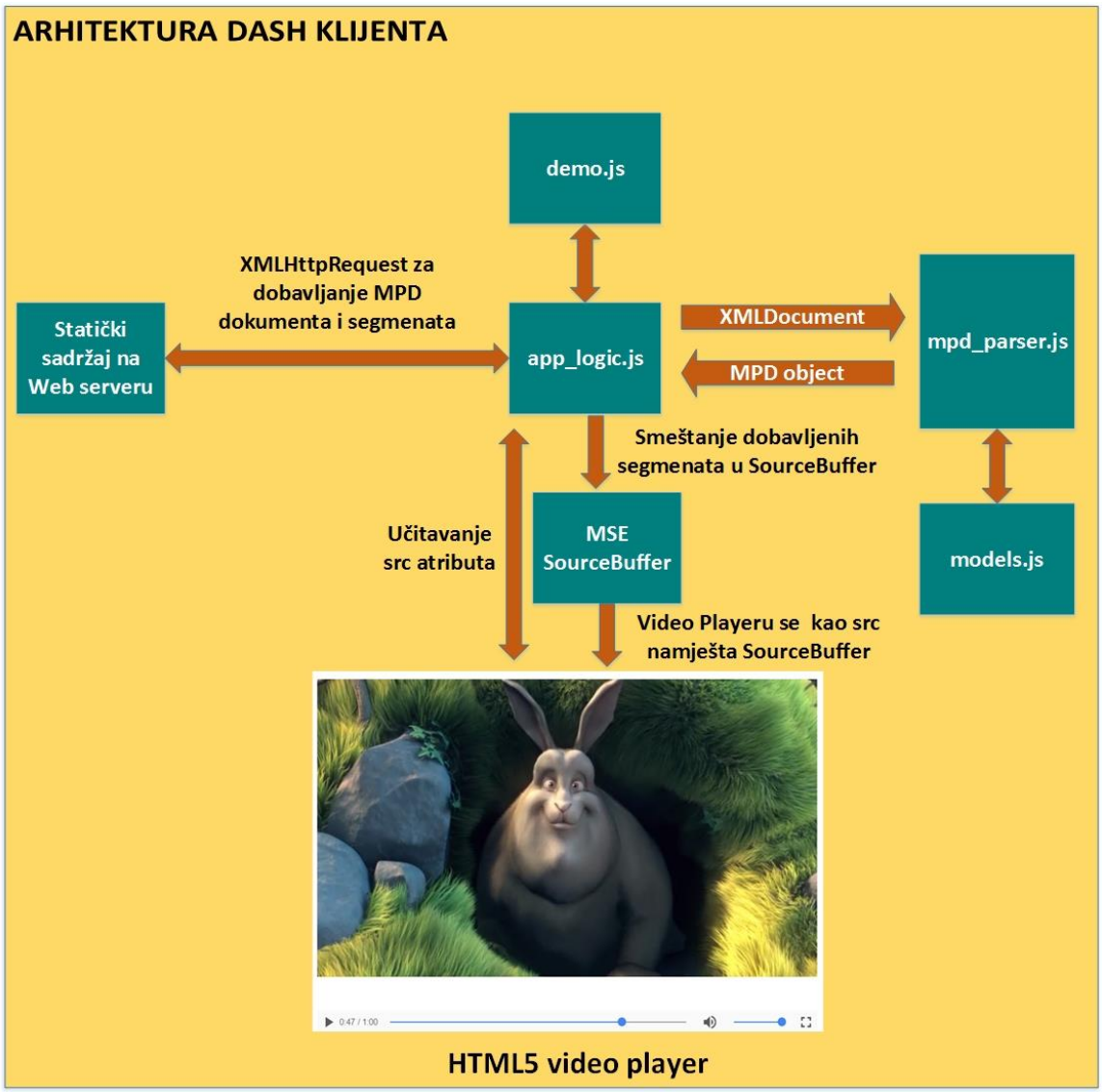
U sklopu rada obavljeno je prepravljane postojeće JavaScript biblioteke kako bi se napravio DASH klijent odnosno ugradile DASH funkcionalnosti u web preglednik. Kako bi klijent radio, funkcionalnosti su rastavljene u više modula. Moduli za implementaciju DASH klijenta su:

- *models.js* - datoteka u kojoj se nalazi model MPD-a, koji je prilagođen aplikacijskoj logici. Omogućava da se svaki element u XML strukturi MPD-a sprema („mapira“) na pojedinačni JavaScript objekt, omogućavajući aplikacijskoj logici jednostavno izvršavanje operacija nad pojedinačnim dijelovima MPD-a
- *mpd_parser.js* - modul za parsiranje MPD-a, oslanja se na aplikacijsku logiku, da bi MPD proslijedio kao *XMLDocument*, kao rezultat *XMLHttpRequest*-a. Prolaskom kroz čvorove *XMLDocument* stabla upotrebom XML DOM (*Document Object Model*) API-a. Za svaki tip čvora *XMLDocument* stabla, stvara se odgovarajući JavaScript objekt iz *models.js* modula. Ovim se kompletno „mapira“ *XMLDocument* stablo na model, koji je jednostavniji za korištenje u aplikacijskoj logici, pri tome nudeći razne opcije za proširenje kako skupa podataka tako i samih funkcionalnosti (koje bi se trebale implementirati u budućnosti).
- *app_logic.js* - datoteka u kojoj se nalazi aplikacijska logika klijenta, koja uključuje:
    - upotrebu XMLHttpRequest objekta radi dobavljanja sadržaja MPD-a.
    - dobavljanje segmenata medijskog toka te reprodukcija istih.
    - kontrolu ponašanja HTML5 video objekta.

Kako bi testirali DASH funkcionalnosti, izrađeni su i moduli koji olakšavaju praktičnu demonstraciju DASH klijenta (prikaz demo stranice) iako oni nisu dijelovi potrebni da bi se ispunila DASH funkcionalnost. Ti moduli su:

- *demo.js* - datoteka koja sadrži pomoćne funkcije za prikaz pojedinih informacija na zaslon i logika odabira MPD-a. Instancira DASH klijent iz *app_logic.js*, zahtjeva njegovu inicijalizaciju te joj prosljeđuje <video> *player* s kojim će raditi.
- *DASHP.html* - datoteka u kojoj se nalazi web stranica s DASH klijentom i gdje podešavamo video *tag* potreban za reprodukciju DASH sadržaja

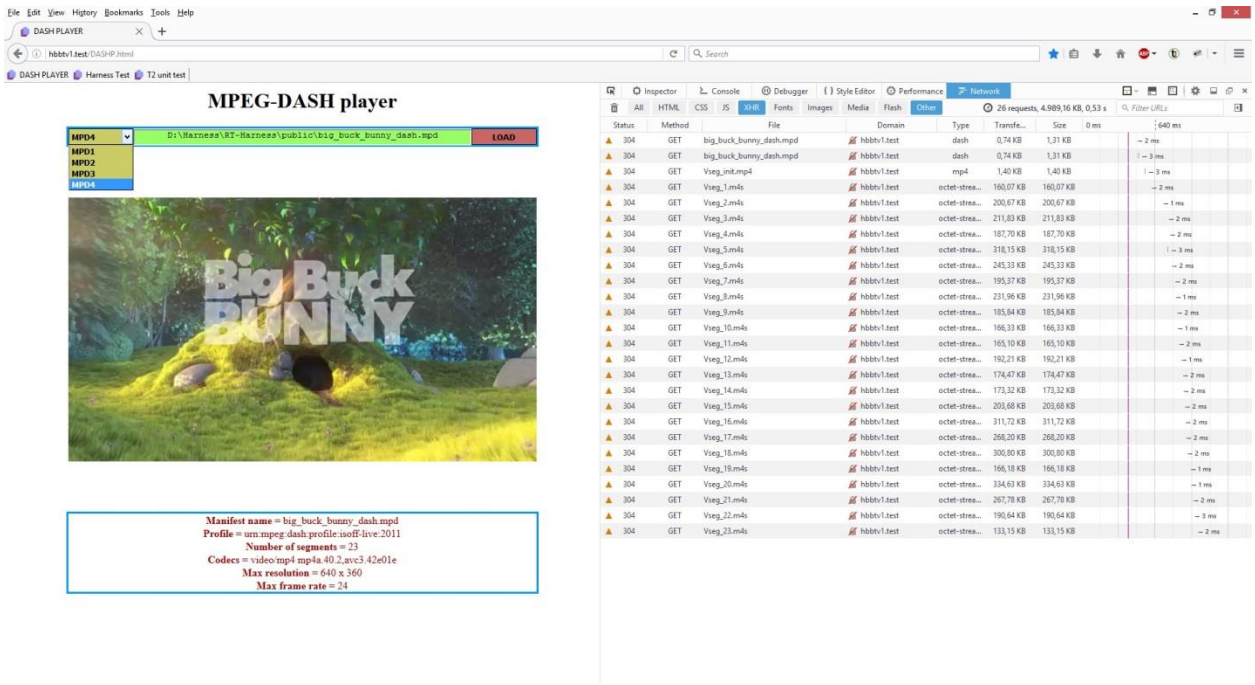
Na slici 6.1 je prikazana arhitektura izrađenog DASH klijenta.



Slika 6.1 Arhitektura DASH klijenta

Na slici 6.2 je prikazan DASH klijent/*player* koji je izrađen koristeći MSE (*Media Source Extensions*). Korisnik u prikazanom klijentu radi iduće korake kako bi dobio traženi sadržaj:

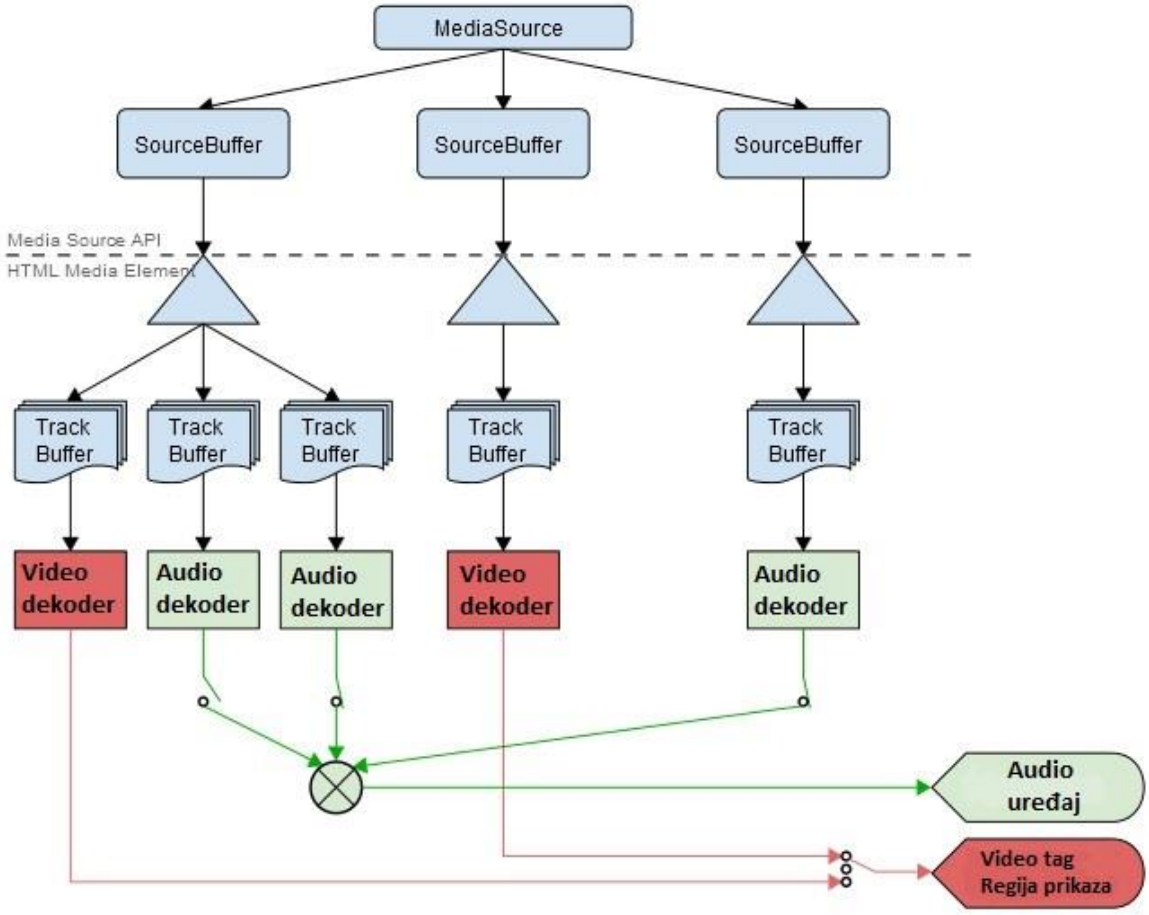
1. Pokreće Harness sučelje potrebno za testiranje klijenta.
2. Pokreće web stranicu DASHP.html.
3. Odabire sadržaj koji želi pogledati iz liste MPD-ova te pritiska tipku *LOAD*
4. Započinje se učitavanje te parsiranje odabranog MPD-a.
5. Započinje se dohvaćanje segmenata a samim time i reprodukcija sadržaja.



Slika 6.2 Primjer dobavljanja segmenata

6.1 MSE

API bez kojeg bi implementacija DASH funkcionalnosti u web preglednik bila nemoguća je MSE. Proširenjem mogućnosti HTML medija elementa, dozvoljava JavaScript-u da generira medijski tok potreban za reprodukciju sadržaja, gdje se time „otvaraju vrata“ za različite vrste upotrebe kao što je prilagodljivo strujanje. Iz modela MSE koji je dan na slici 6.3 vidimo da *sourcebuffer* može u sebi sadržavati i audio i video komponente ili komponente neovisne jedne od drugoj.



Slika 6.3 MSE model[7]

Ciljevi pri razvoju MSE su bili da:

- Dozvole JavaScript-u da stvara medijske tokove (video i audio) bez obzira na koji način se segmenti dobavljaju.
- Definiraju model spajanja i *buffering*-a segmenata za adaptivano strujanje, umetanje reklama, uređivanje videa ...
- Smanje potrebu za parsiranjem medija u JavaScriptu.
- Da iskoriste pred-memoriju (*cache*) web preglednika koliko god je moguće.
- Ne zahtijeva podršku za bilo koji medijski format ili *codec*.

Na slici 6.4 prikazan je jednostavan primjer korištenja MSE za dobavljanje segmenata i reprodukcije istih.

```
var ms = new MediaSource(); //inicijalizacija MSE API-ja odnosno definiranje MediaSource objekta

var video = document.querySelector('video');    //video element web stranice
video.src = window.URL.createObjectURL(ms);     //pridruživanje MediaSource objekta klasičnom HTML video elementu
ms.addEventListener('sourceopen', function(e) { //listener sourceopen događaja koji prati trenutno stanje MediaSource objekta
  ...
  var sourceBuffer = ms.addSourceBuffer('video/webm; codecs="vorbis,vp8"');     //definiranje novog sourceBuffer-a
  sourceBuffer.appendBuffer(oneVideoWebMChunk);     //dodavanje segmenata(oneVideoWebMChunk) sourceBuffer-u
  ....
}, false);
```

Slika 6.4 Primjer korištenja MSE

7. OPIS I PRIMJER TESTA

Kao dio praktičnog rada, dobivena su tri paketa testova DASH_04, DASH_07 i DASH_08. U svakom se nalazi više testova koji su različiti, sve ukupno je odrađeno 66 testova, od kojih je odabran jedan te prikazan čitav proces (od početnog zadatka do izrade testa, te samog testiranja ispravnosti). To uključuje:

1. Dobivanje XML dokumenta od klijenta u kojoj su zapisane informacije o testu (naziv, što i kako treba raditi, za koji standard ...).
2. Proučavanje načina na koji bi se to moglo riješiti (čitanje MPEG DASH specifikacije).
3. Izrada samog testa koji se otvara u Web pregledniku (html5 format).
4. Izrada T2Unit testova kojim se testira i provjerava ispravnost napisane html5 datoteke.
5. Izrada JSON datoteke u kojoj se nalazi plan izvođenja T2Unit testova
6. Testiranje

Kako bi vidjeli na primjeru sve ove korake, u narednim poglavljima prikazat ćemo jedan test iz DASH_08 paketa.

7.1 XML struktura DASH testa

Za primjer uzimamo test pod nazivom DASH-ERRORHANDLE0030. Od klijenta dobivamo XML dokument koji je prikazan u prilogu 1. Iz tog dokumenta iščitamo naziv testa, tko je klijent, za koju verziju HBBTV-a se radi test, te najvažnije od svega, što test treba da radi što je zapisano u *assertionText tag*-u. Nakon što su odrađeni svi koraci, taj dokument s našim dodanim testnim koracima se mora ažurirati, s vremenom kad je isporuka danog testa, te imenom kompanije koja radi isporuku. Ažuriran XML dokument je dan u prilogu 2, gdje se vide dva dodana *contributor tag*-a, opisani testni koraci (*testStep*) te njihovo očekivano ponašanje (expectedBehaviour), kriterij koji govori kad je test prošao (*passCriteria*), naziv MPD-a koji je korišten (*mediaFile*), te dva dodana *historyItem tag*-a gdje se nalaze datumi isporuke napravljenog paketa testova.

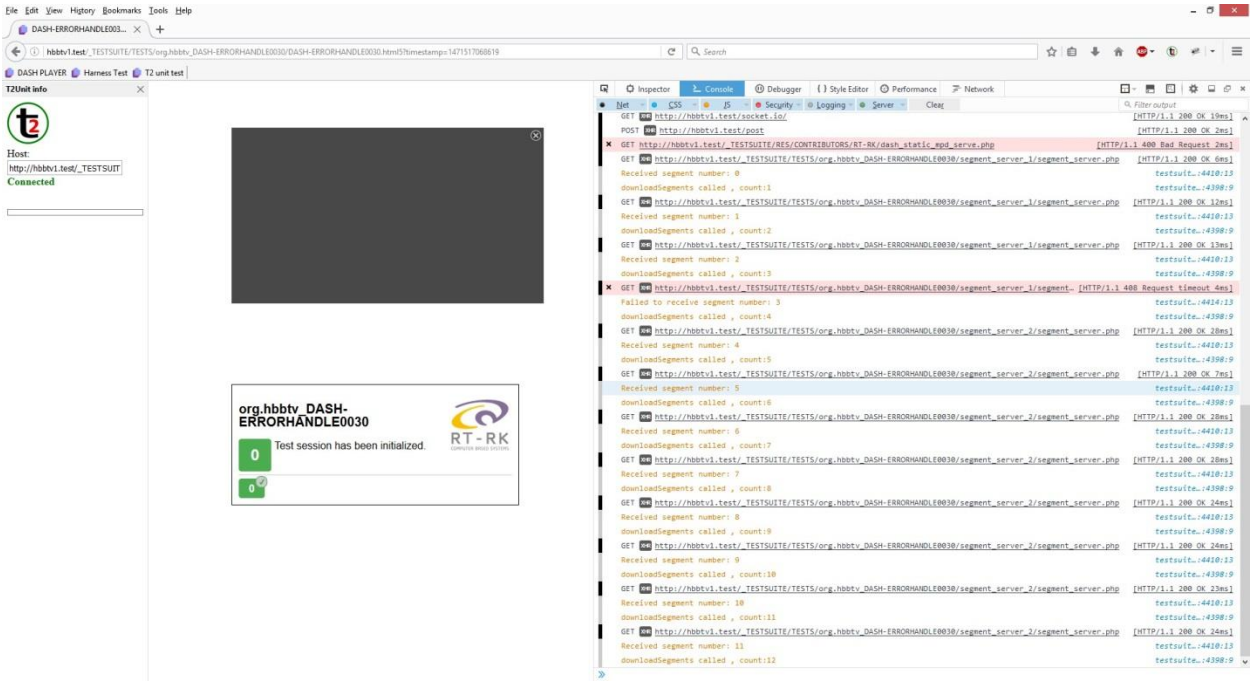
7.2 HTML5 DASH test

Html5 dokument predstavlja mjesto gdje se pišu funkcije testa koji se mora napraviti na osnovu *assertionText*-a iz XML dokumenta. U njemu se koristi HTML, JavaScript te PhP (*Hypertext Preprocessor*) skripte. U prilogu 3 se može vidjeti cjelokupni izrađen HTML5 dokument za dani primjer. Također, za potrebe testa su napisane dvije PhP skripte u kojima su

postavljeni parametre za samo testiranje testa (koji segment će davati grešku, na kojem serveru se nalaze svi segmenti ...). PhP skripte se mogu pogledati u prilogu 4 i prilogu 5.

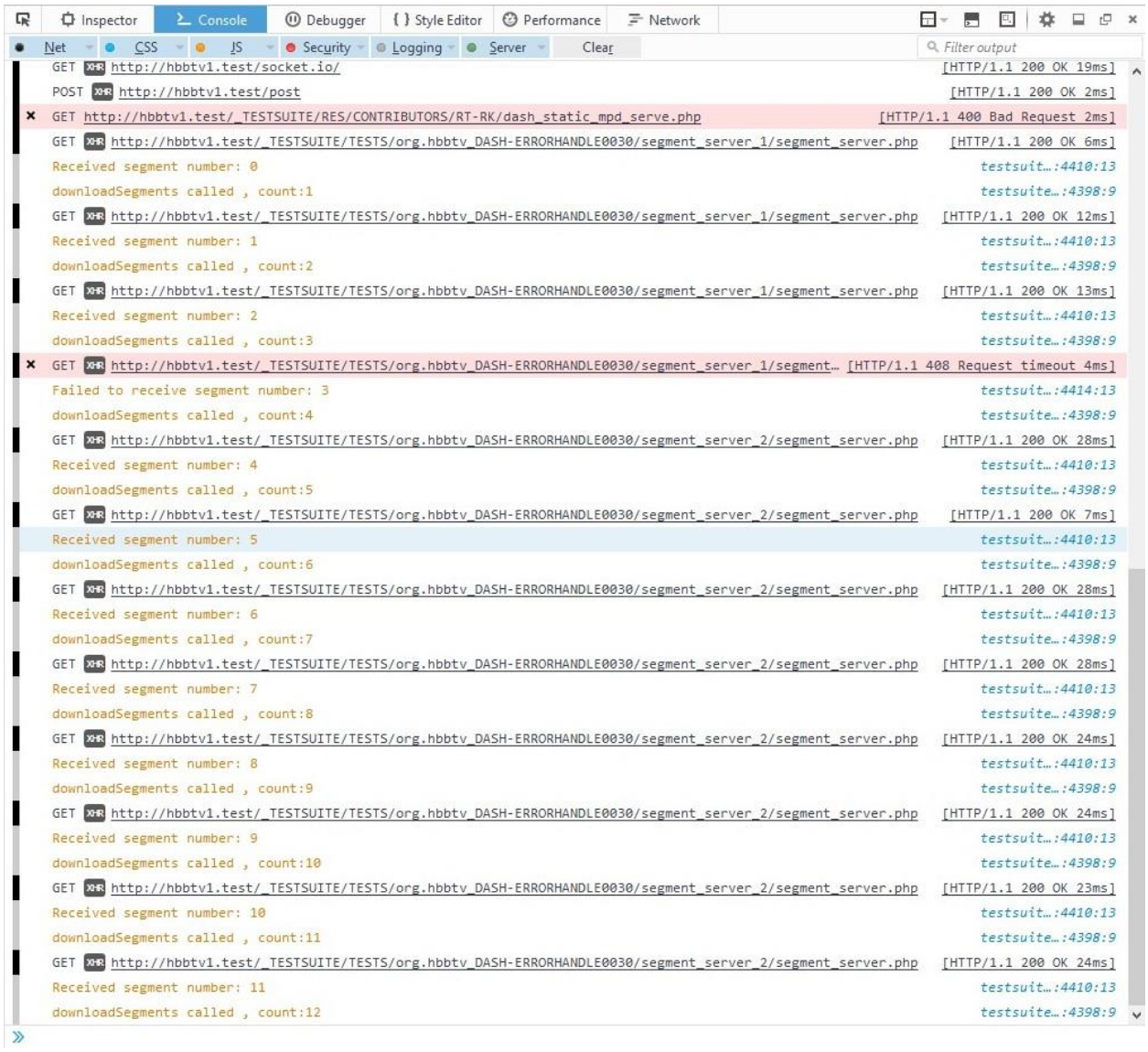
7.3 T2Unit testovi i testni plan

T2Unit testovi predstavljaju testove koji kroz nekoliko koraka prvjeravaju napisani test. Cjelokupan T2Unit test za ovaj primjer sastoji se od deset JavaScript datoteka. Kako postoji 5 koraka/provjera za ovaj test, svaka provjera ima svoje dvije datoteke. U prilogu 6 dan je prikaz jednog od koraka provjere, dok se ostali mogu pogledati u dokumentima na CD-u. Vidi se da *t2_seg_sequence_without_retry* dokument poziva *seg_sequence_without_retry* koji dobavlja inicijalni segment te ostale medijske segmente sa servera. Kako je u prilogu 4 (segment_server_1) deklarirano da segment broj 3 baca grešku, od jednog servera se traže segmenti, i kada naiđemo na treći segment (odnosno segment koji ima grešku), dobavljanje segmenata se prebacuje na drugi server, bez ponovnog traženja segmenata s prvog servera. Na slici 6.1 se vidi obavljeno testiranje unutar samog preglednika.



Slika 7.1 Cjelokupan izgled preglednika za vrijeme testiranja

Dalje na slici 6.2 vidimo detaljne informacije u konzoli o dobavljanju segmenata. Kao što je rečeno, segmenti se dobavljaju s prvog servera, te kada se naiđe na segment s greškom u ovom slučaju treći segment, zahtjevi za segmentima šalju se na drugi server.



Slika 7.2 Dobavljanje segmenata (ispis u konzoli)

Također je korištena *proprietery* ekstenzija za web preglednik imena „*T2Unit info*“ koja prikazuje koliko T2Unit testova postoji, koliko ih je ispravno (*pass*) odnosno pogrešno (*fail*) te njihove informacije. Ekstenzija preglednika u radu prikazana je na slici 6.3.

Slika 7.3 *T2Unit info* ekstenzija za web preglednik

Testni plan za ovaj te ostale testove je JSON datoteka u kojoj se spisak svih T2Unit testova koji se izvršavaju i popis putanja gdje se ti testovi nalaze. Primjer testnog plana za ovaj test je dan u prilogu 8. Nakon završetka testiranja s *T2Unit info*, stvara se još jedan JSON (*JavaScript Object Notation*) dokument koji također daje rezultate testova, ali s detaljnijim opisom.

8. ZAKLJUČAK

Dizajniran s svrhom neovisnosti o *codec*-u, fleksibilnosti, mogućnosti korištenja postojeće tehnologije, adaptivnosti, te pružanja korisniku najbolje moguće kvalitete sadržaja, četiri godine nakon što je MPEG DASH standard objavljen od strane ISO organizacije, na odličnom je putu da postane glavni standard za strujanje, odnosno da pruži univerzalnu platformu za isporuku multimedije. Korištenjem MSE proširena je postojeća JavaScript biblioteke kako bi podržavala DASH funkcionalnosti, te je realiziran DASH klijent, u kojem za razliku od postojećih rješenja korisnik ima potpunu kontrolu njegova ponašanja. Za potrebe testiranja korištenjem razvijenog klijenta postoji mogućnost stvaranja proizvoljnih test scenarija (npr. dobavljanje samo određenih segmenata, dobavljanje više segmenata nego što je potrebno ...). Također, treba spomenuti kako ostala rješenja nemaju mogućnost izvještavanja o greškama (DVB *reporting*-a) dok će ovdje ta mogućnost biti implementirana. Još jedna prednost ove implementacije je ta da jer se radi o JavaScript biblioteci, može se jednostavno integrirati gdje god to korisnik poželi i bit će ostvarene DASH funkcionalnosti, odnosno adaptivno strujanje.

LITERATURA

[1] https://www.w3.org/Protocols/rfc2616/rfc2616.html [pristup ostvaren 15.6.2016]

[2] http://searchnetworking.techtarget.com/answer/What-is-the-difference-between-OSI-model-and-TCP-IP-other-than-the-number-of-layers [pristup ostvaren 15.6.2016]

[3] ETSI TS 102 796 V1.3.1 - HbbTV

[4] Antony Vetro - https://www.computer.org/cms/Computer.org/ComputingNow/homepage/2011/1211/T_MM1_TheMPEGDASHStandard.pdf [pristup ostvaren 15.8.2016]

[5] https://www.brendanlong.com/the-structure-of-an-mpeg-dash-mpd.html [pristup ostvaren 15.8.2016]

[6] Thomass Stockhammer, MPEG DASH - https://www.w3.org/2011/09/webtv/slides/W3C-Workshop.pdf [pristup ostvaren 15.8.2016]

[7] https://www.w3.org/TR/media-source/ [pristup ostvaren 15.8.2016]

[8] http://dashif.org/ [pristup ostvaren 15.8.2016]

[9] ETSI TS 103 285 V1.1.1 - MPEG-DASH Profile for Transport of ISO BMFF

[10] http://www.hbbtv-developer.com/site/blog/?p=879 [pristup ostvaren 16.8.2016]

[11] https://www.wowza.com/forums/content.php?508-How-to-do-MPEG-DASH-streaming [pristup ostvaren 16.8.2016]

[12] MPEG DASH streaming reference and resources - https://msdn.microsoft.com/en-us/library/dn551368(v=vs.85).aspx [pristup ostvaren 18.8.2016]

[13] Christian Timmerer, Dynamic Adaptive Streaming over HTTP - http://www.slideshare.net/christian.timmerer/dynamic-adaptive-streaming-over-http-from-content-creation-to-consumption [pristup ostvaren 18.8.2016]

[14] http://www.encoding.com/mpeg-dash/ [pristup ostvaren 18.8.2016]

POPIS KRATICA

HbbTV	Hybrid Broadcast Broadband TV
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
HTTP	Hypertext Transfer Protocol
XML	eXtended Markup Language
PHP	Hypertext Preprocessor
MPEG	Moving Picture Experts Group
3GPP	3rd Generation Partnership Project
DASH	Dynamic Adaptive Streaming over HTTP
TCP/IP	Transmission Control Protocol/Internet Protocol
QoS	Quality of Service
OSI	Open Systems Interconnection
MHEG	Multimedia and Hypermedia Experts Group
DVB	Digital Video Broadcasting
DTV	Digital television
VOD	Video on Demand
EPG	Electronic Program Guide
DSM-CC	Digital Storage Media Command and Control
AIT	Application Information Table
DUT	Device Under Test
FIFO	First In First Out
API	Application Programming Interface
DOM	Document Object Model

MPD	Media Presentation Description
ISO	International Organization for Standardization
ISOBMFF	ISO Base Media File Format
URL	Uniform Resource Locator
MPEG TS	MPEG transport stream
JSON	JavaScript Object Notation
MSE	Media Source Extensions
AAC	Advanced Audio Coding

SAŽETAK

U diplomskom radu „*Verifikacija DASH podrške u HbbTV okruženju"* dan je pregled HbbTV industrijskog standarda s posebnim osvrtom na MPEG DASH funkcionalnost, standarda koji definira adaptivne tehnike slanja multimedijskih sadržaja preko HTTP-a. Detaljno je opisan sam DASH standard, njegova MPD struktura, uloga, profili te sama implementacija. Opisan je podskup testnih slučaja kojima se provjerava pravilna implementacija podrške za DASH sadržaje korištenjem HbbTV test API-a u skladu s pravilima testiranja koja definira HbbTV udruženje. Proširena je postojeća JavaScript biblioteku s funkcionalnostima kako bi se izradio DASH klijent potreban za provjeru testnih slučaja u okruženju web pretraživača.

Ključne riječi: HbbTV, RT-Harness, MPEG DASH, HTTP, MPD, segment, klijent, testni slučaj, provjera, video strujanje, dinamičko, adaptivno, MSE

ABSTRACT

In the theses „*Verification of DASH support in HbbTV environment*", a review of HbbTV industrial standard is given, with a special reference to MPEG DASH functionality, a standard which defines adaptive tehnique of sending multimedia content over HTTP. DASH standard, his MPD structure, role, profiles and implementation is described. A step by step example is given of test cases in which, proper implementation of support for DASH content using HbbTV test API in accordance with the rules of testing given by HbbTV Association is verified. Existing JavaScript library was updated, to produce DASH client for verification of test cases in web browser.

Keywords: HbbTV, RT-Harness, MPEG DASH, HTTP, MPD, segment, client, test case, verification, video streaming, dynamic, adaptive, MSE

ŽIVOTOPIS

Nemanja Nikić rođen je 16. ožujka 1992. godine u Vukovaru. 1998. kreće u Osnovnu školu Negoslavci u Negoslavcima. Nakon završetka osnovne škole, upisuje Tehničku školu Nikole Tesle u Vukovaru smjer tehničar za računalstvo. 2010. godine upisuje stručni studij smjer informatika na Elektrotehničkom fakultetu u Osijeku. Po završetku stručnog studija 2013. godine upisuje razlikovnu godinu, smjer komunikacije i informatika, te nakon njenog završetka 2014. godine upisuje diplomski studij smjer komunikacije i informatika. U rujnu 2015. godine postaje stipendista u Institutu RT-RK Osijek.

PRILOZI

Prilog 1 - dobiveni XML dokument

Prilog 1 smo zbog same veličine dokumenta možete pogledati na CD-u. Pronaći ćete ga pod imenom org.hbbtv_DASH-ERRORHANDLE0030_original.xml

Prilog 2 - ispunjeni XML dokument

Prilog 2 smo zbog same veličine dokumenta možete pogledati na CD-u. Pronaći ćete ga pod imenom org.hbbtv_DASH-ERRORHANDLE0030.xml

Prilog 3 - html5 test dokument

Prilog 3 smo zbog same veličine dokumenta možete pogledati na CD-u. Pronaći ćete ga pod imenom DASH-ERRORHANDLE0030.html5

Prilog 4 - *segment_server_1* php skripta

```
<?php

        /*****************************************************************************
*****************************************************

         *      DASH Error Handling - miscellaneous request errors [static MPD; HTTP 408]

         * @assertionText -       A live profile MPD with @type == static has a single Adaptation Set
           and contains multiple absolute BaseURLs within its MPD element, as follows:

         *

         * &lt;BaseURL0&gt; with @priority == 1, @serviceLocation == "A"

         * &lt;BaseURL1&gt; with @priority == 2, @serviceLocation == "B"

         *

         * When a terminal, which has been playing the DASH stream described by this MPD with no
errors since the session began, encounters an HTTP 408 (Request timeout) error code, it switches from
making segment requests using &lt;BaseURL0&gt; to making segment requests            using
&lt;BaseURL1&gt; after 0 or 1 failed retry attempts.


*****************************************************************************************
*************************************************/

        /**

        * @desc This php file accepts requests for mpeg dash segments and return their content; Also
        returns HTTP 408 for specified segment

        *

        */

        $ERROR = TRUE; // segment server should return error on segment request

        $ERROR_CODE = 'HTTP/1.1 408 Request timeout';

        $ERROR_SEGMENT = 3; // request for specified segment will return error

        $REQ_SEGMENT_COUNT = TRUE; //Use segment counter from request log, not from
        segment URL



        $SEGMENT_LOAD_REPORT = TRUE;

        $SEGMENT_LOAD_REPORT_FILE = 'load_report.json';



        include dirname(__FILE__) . '/../../../RES/CONTRIBUTORS/RT-
        RK/dash_segment_server.php';
```

Prilog 5 - *segment_server_2* php skripta

```
<?php

        /*****************************************************************************
*****************************************************

         *        DASH Error Handling - miscellaneous request errors [static MPD; HTTP 408]

         * @assertionText -       A live profile MPD with @type == static has a single Adaptation Set
        and contains multiple absolute BaseURLs within its MPD element, as follows:

         *

         * &lt;BaseURL0&gt; with @priority == 1, @serviceLocation == "A"

         * &lt;BaseURL1&gt; with @priority == 2, @serviceLocation == "B"

         *

         * When a terminal, which has been playing the DASH stream described by this MPD with no
errors since the session began, encounters an HTTP 408 (Request timeout) error code, it switches from
making segment requests using &lt;BaseURL0&gt; to making segment requests using
&lt;BaseURL1&gt; after 0 or 1 failed retry attempts.


*****************************************************************************************
*************************************************/

        /**

        * @desc This php file accepts requests for segments and return their content, also makes file
           which indicates that this URL was used.

        *

        */

        $ERROR = FALSE;


        $SEGMENT_LOAD_REPORT = TRUE;

        $SEGMENT_LOAD_REPORT_FILE = 'load_report.json';


        include dirname(__FILE__) . '/../../../RES/CONTRIBUTORS/RT-
        RK/dash_segment_server.php';
```

Prilog 6 - *t2_seg_sequence_without_retry*

```
T2UNIT.loadFixtures("seg_sequence_without_retry.js");
T2UNIT.on("reportStepResult", function(step, result, comment) {
    console.log('\n\nSTEP: \n\n'  + step);
    if (!result) {
        T2UNIT.finished(false,"Test case failed in step "+step+", comment:"+comment);
    }
});
```

Prilog 7 - *seg_sequence_without_retry*

```
window.addEventListener("load", function(event) {

        URL1 = "hbbtv1.test/_TESTSUITE/TESTS/org.hbbtv_DASH-
ERRORHANDLE0260/segment_server_2/segment_server.php";

});

// request log without retry attempt


/**

 * Fired after the initTestSession

 */

function t2Fixture() {

   var segmentList = [];

   // request log without retry attempt

   segmentList.push("segment_server_1/segment_server.php?sid="+ SESSION_ID +
        "&url=Video_1min_3secseg_LIVE/media_video.fmp4/video/Vseg_init.mp4");

   for (var i = 1; i < 4; i++) {

      segmentList.push("segment_server_1/segment_server.php?sid="+ SESSION_ID +
        "&url=Video_1min_3secseg_LIVE/media_video.fmp4/video/Vseg_" + i + ".m4s");

   }

   for (var i = 3; i < 22; i++) {

      segmentList.push("segment_server_2/segment_server.php?sid="+ SESSION_ID +
        "&url=Video_1min_3secseg_LIVE/media_video.fmp4/video/Vseg_" + i + ".m4s");

   }

   overridenObjects.AVControl.setSegmentList(segmentList);

}

// the report can only be fired after the SESSION_ID has been acquired.

// the reportConfig has to be set after the initTestSession  step

applyFixtureBeforeStep('step1_startStreamPlayback', t2Fixture);
```

Prilog 8 - test plan JSON dokument

Prilog 8 smo zbog same veličine dokumenta možete pogledati na CD-u. Pronaći ćete ga pod imenom ERRORHANDLE0030.json

Prilog 9 - test plan result JSON dokument

Prilog 9 smo zbog same veličine dokumenta možete pogledati na CD-u. Pronaći ćete ga pod imenom ERRORHANDLE0030.json.result.json