Verifikacija sinkronizacije više uređaja u HbbTV okruženju

Ivešić, Zvonimir

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:200:225912

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni diplomski studij

VERIFIKACIJA SINKRONIZACIJE VIŠE UREĐAJA U HBBTV OKRUŽENJU

Diplomski rad

Zvonimir Ivešić

Osijek, 2016.

Sadržaj

1. UVOD 1
2. TEORIJSKE OSNOVE 2
   2.1. Televizija i internet 2
   2.2. Web tehnologije 2
      2.2.1. HTML i CSS 2
      2.2.2. JavaScript 3
      2.2.3. WebSocket 3
      2.2.4. Node.js 4
   2.3. HbbTV standard 4
      2.3.1. Arhitektura HbbTV sustava 5
      2.3.2. HbbTV aplikacije 6
      2.3.3. Dodatni uređaji 7
      2.3.4. Komunikacija i sinkronizacija dodatnih uređaja 8
   2.4. Sustav za testiranje HbbTV prijemnika 10
      2.4.1. Testni slučaj 12
      2.4.2. Testno okruženje 13
3. PROGRAMSKA SUČELJA POTREBNA ZA REALIZACIJU TESTOVA 15
   3.1. WebSocket programsko sučelje 15
   3.2. Pristupne točke terminala 17
   3.3. Sučelja za identifikaciju sadržaja i sinkronizaciju 18
   3.4. Emulirani objekti prijemnika 18
4. POSTUPAK IZRADE I PRIMJER TESTA 21
   4.1. Postupak izrade testa 21
   4.2. Primjer testa iz skupine APP2APP 22
   4.3. Primjer T2Unit testa 25
5. ZAKLJUČAK 26

LITERATURA 27

KRATICE 28

SAŽETAK 29

ABSTRACT 30

ŽIVOTOPIS 31

1. UVOD

U novije vrijeme se pojavio trend integracije televizijskog prijemnika s drugim uređajima u kući kao što su mobilni uređaji, tableti i drugi televizijski prijemnici. Općeprihvaćeni standard koji opisuje način na koji se uređaji trebaju povezivati nosi naziv HbbTV (engl. *Hybrid broadcast broadband Television*). Osim definiranja načina povezivanja uređaja, HbbTV nudi i druge mogućnosti kao što su HbbTV aplikacije kojima se obogaćuje iskustvo gledanja televizije.

Proizvođači uređaja prije plasiranja uređaja na tržište moraju provjeriti kompatibilnost uređaja s HbbTV standardom što se ispituje HbbTV testovima. Postoje različiti testovi koje propisuje HbbTV udruženje i koje uređaj mora proći kako bi dobio potvrdu da je sukladan s HbbTV standardom. U ovom radu fokus je na testovima kojima se provjerava uspješnost sinkronizacije više različitih uređaja u HbbTV okruženju. Budući da potrebna programska sučelja za izgradnju prethodno spomenutih testova nisu na raspolaganju, najprije je izrađena odgovarajuća programska podrška, a zatim su izrađeni testovi za verifikaciju sinkronizacije više uređaja u HbbTV okruženju.

Rad je strukturiran na sljedeći način. Na početku rada obrađene su teorijske osnove. Ukratko su opisani televizija i internet. Potom je predstavljen HbbTV, industrijski standard koji ima cilj objediniti televiziju i širokopojasnu dostavu sadržaja. Potom je opisan sustav za testiranje na kojem se izvršavaju testovi koji su krajnji cilj ovog rada. Navedene su i ukratko objašnjene *web* tehnologije korištene u ovom radu. HTML (engl. *HyperText Markup Language*) i *JavaScript* se koriste za izradu testnih slučajeva. *WebSocket* se u HbbTV standardu koristi za komunikaciju između aplikacija i druge stvari. Node.js je korišten za izradu *WebSocket* servera koji emulira rad HbbTV uređaja.

U trećem poglavlju su opisana programska sučelja i objekti koje je bilo potrebno izraditi kako bi se mogli provoditi HbbTV testovi. Programskim sučeljem za *WebSockete* su proširene mogućnosti ove tehnologije i omogućeno je nadziranje, testiranje i multipleksiranje velikog broja veza što je potrebno za ispunjenje zahtjeva HbbTV norme. Zatim su opisane pristupne točke i emulirani objekti HbbTV uređaja potrebni za pisanje i testiranje testova.

U posljednjem poglavlju je opisan postupak izrade jednog HbbTV testa. Taj postupak je prikazan i na jednom primjeru. Osim toga dan je i primjer T2Unit testa koji provjerava ispravan rad HbbTV testa.

2. TEORIJSKE OSNOVE

U ovom poglavlju ukratko su opisani osnovni pojmovi vezani za televiziju, internet i HbbTV standard. Upoznavanje s ovim pojmovima je potrebno za daljnje razumijevanje rada.

2.1. Televizija i internet

Televizija ili skraćeno TV je općeniti naziv za skup tehnologija koje omogućuju snimanje, emitiranje i prijem pokretnih slika, u počecima u crno-bijeloj tehnici, danas u boji. Emitirana slika je također popraćena zvukom. Riječ televizija se osim za navedeno kolokvijalno koristi i kao naziv za uređaj koji koristimo za prijem signala – televizijski prijemnik.

Temeljni princip rada televizije je pretvorba pokretne slike u električni signal pogodan za prijenos i obrnuti postupak pretvorbe na televizijskom prijemniku [1].

U današnje vrijeme većina država emitira televizijski signal digitalno. Prelaskom na digitalnu televiziju moguće je postići veću kvalitetu slike, u visokim rezolucijama, te prijenos više televizijskih kanala na istoj frekvenciji. Osim toga dostupne su i neke naprednije usluge kao što je elektronski programski vodič.

Internet je javno dostupna globalna paketna podatkovna mreža koja povezuje računala i računalne mreže korištenjem istoimenog protokola (engl. *Internet Protocol - IP*). To je mreža na koja se sastoji od milijuna kućnih, akademskih, poslovnih i vladinih mreža koje razmjenjuju informacije. Dostupne su i druge usluge kao što su elektronička pošta, prijenos datoteka, prikaz internetskih stranica i drugo [2].

2.2. Web tehnologije

U ovom poglavlju ukratko su opisani jezici *web* dizajna i druge *web* tehnologije koje se koriste za izradu ovog rada.

2.2.1. HTML i CSS

HTML je jezik za izradu internetskih stranica. Pomoću HTML-a se stvaraju hipertekst dokumenti te se u njima oblikuje sadržaj i stvaraju poveznice na druge HTML dokumente. Za prikaz hipertekst dokumenata se koristi neki od internet preglednika. HTML jezik omogućava internet pregledniku pravilan prikaz hipertekst dokumenata. Pri tome se nastoji da dokument (odnosno internetska stranica) izgleda jednako bez obzira na kojem se pregledniku, uređaju ili operacijskom sustavu dokument pregledava. HTML je prezentacijski jezik, služi za strukturirani

zapis nekakvih informacija te u njemu nije moguće izvršiti niti najjednostavnije matematičke operacije kao što su zbrajanje i oduzimanje [3].

HTML dokument se sastoji od HTML elemenata. Svaki se HTML element sastoji od para HTML oznaka (engl. *tags*). Dodatno, svaki element može imati atribute kojima se definiraju specifična svojstva tog elementa. Svaka oznaka počinje znakom manje (<) a završava znakom veće (>). Zatvarajuća oznaka se kreira na isti način s tim da se poslije znaka manje (<) dodaje i kosa crta (/).

CSS (engl. *Cascading Style Sheets*) je stilski jezik koji se koristi za opis prezentacije dokumenta napisanog pomoću HTML-a. Sam sadržaj internetske stranice se piše u HTML-u dok je oblikovanje internetske stranice zapisano u posebnoj CSS datoteci [4].

U ovom radu se HTML koristi za izradu HbbTV testova koji su zapravo HTML dokumenti.

2.2.2. JavaScript

JavaScript je skriptni programski jezik. Izvršava se u internetskom pregledniku na klijentskoj strani. Podržavaju ga svi relevantni internetski preglednici. Za razliku od primjerice programskih jezika C ili Java koji se najprije trebaju prevesti kako bi se mogli izvršavati, *JavaScript* se ne prevodi u izvršnu datoteku koja se pokreće već se kod izvršava slijedno naredbu po naredbu [5].

JavaScript jezik dodaje interaktivnost internet stranicama, odnosno statičnim HTML dokumentima te omogućava programiranje unutar HTML dokumenta. U *JavaScriptu* je moguće reagirati na neke događaje, npr. učitavanje stranice, korisnikov klik na neki gumb, na neki HTML element i slično. Moguće je pročitati ili izmijeniti sadržaj nekog HTML elementa. Također, moguće je odrediti koji preglednik korisnik upotrebljava i na osnovu toga prilagoditi stranicu korisnikovu pregledniku, odnosnu učitati onu koja najbolje odgovara istome. *JavaScriptom* je moguće pohraniti kolačiće (engl. *cookies*) na korisnikovo računalo te ih kasnije učitati [6].

JavaScript se u ovom radu koristi za izvršavanje sve logike koja je potrebna kako bi se izvršili testovi unutar testnog okruženja.

2.2.3. WebSocket

WebSocket protokol omogućava dvosmjernu komunikaciju između korisnika i udaljenog pružatelja usluge preko jedne TCP (engl. *Transmission Control Protocol*) veze. Uspostavljanje veze je kompatibilno s HTTP-om (engl. *HyperText Transfer Protocol*) kako bi poslužitelj na odgovarajućem portu mogao primati HTTP i *WebSocket* zahtjeve [7]. Protokol je dizajniran za

upotrebu u internetskim preglednicima i serverima, ali ga može koristiti bilo koja klijentska ili serverska aplikacija. Početno rukovanje se vrši putem HTTP zahtjeva, te se provodi „nadogradnja“ na TCP vezu. Na ovaj način se značajno smanjuje zaglavlje paketa i ostvaruje efikasnija komunikacija. Za razliku od dvostrane komunikacije HTTP-om gdje je klijent morao neprestano ispitivati server (engl. *long-polling*) ima li novih poruka koje mu treba poslati, kod *WebSocketa* server može sam slati poruke putem uspostavljene veze. Osim tekstualnih poruka, putem *WebSocketa* je moguće slati i binarne podatke. Također postoji i mehanizam za provjeru dostupnosti spojenih strana (klijenta i servera). Taj mehanizam se naziva *ping-pong*. Jedna strana šalje *ping* poruku određenog sadržaja na što druga strana mora u što kraćem vremenu odgovoriti *pong* porukom istog sadržaja [8].

WebSocketi se koriste u HbbTV-u za komunikaciju između uređaja (dva terminala, terminala i dodatnog uređaja), za komunikaciju između aplikacija na uređajima i drugo.

2.2.4. Node.js

Node.js ili *Node* je platforma za izgradnju brzih i fleksibilnih serverskih aplikacija. Kod u *Nodeu* se piše u *JavaScriptu* i postoje mnogi moduli napisani u *JavaScriptu* koji olakšavaju rad. Postoje moduli za rad s datotečnim sustavom, za rad s mrežom, binarnim podacima, tokovima podataka i mnogi drugi. Također ugrađene su i biblioteke za HTTP i *socket* komunikaciju zbog čega je moguće pisanje internet servera u *Nodeu* bez dodatnih serverskih softvera [9].

Dio sustava za testiranje je napisan u *Nodeu*, kao i *WebSocket* serveri napravljeni u sklopu ovog rada.

2.3. HbbTV standard

HbbTV je industrijski standard i inicijativa kojoj je cilj uskladiti dostavu televizijskog sadržaja odašiljanjem i širokopojasnim putem. Na HbbTV standardu radi udruženje sastavljeno od pružatelja televizijskih usluga i proizvođača potrošačke elektronike.

Kod HbbTV-a je moguća reprodukcija sadržaja iz različitih izvora, uključujući emitirani prijenos, internet i povezane uređaje pa otuda i naziv hibridna televizija. Za gledanje ovakve, hibridne, televizije korisnicima je potreban hibridni TV ili STB (engl. S*et-Top Box*) uređaj (u daljnjem tekstu prijemnik) s raznim ulazima, uključujući širokopojasni priključak kao i prijemnik za prijem emitiranog TV signala (engl. *tuner*). Prijemnik može biti za digitalnu zemaljsku (engl.

Digital Video Broadcast - Terrestrial DVB-T, DVB-T2), digitalnu kabelsku (engl. *Cable*, DVB-C, DVB-C2) ili digitalnu satelitsku televiziju (engl. *Satellite* DVB-S, DVB-S2).

HbbTV standard se zasniva na postojećim standardima i *web* tehnologijama kao što je prikazano na slici 2.1.



Sl. 2.1. *Standardi na kojima se zasniva HbbTV standard [10].*

Dostupne usluge su poboljšani teletekst, video na zahtjev, sadržaj za podsjećanje na prošle emisije (engl. *catch-up services*), elektronski programski vodič, interaktivno oglašavanje, glasanje, igrice, povezanost s društvenim mrežama i druge multimedijske aplikacije.

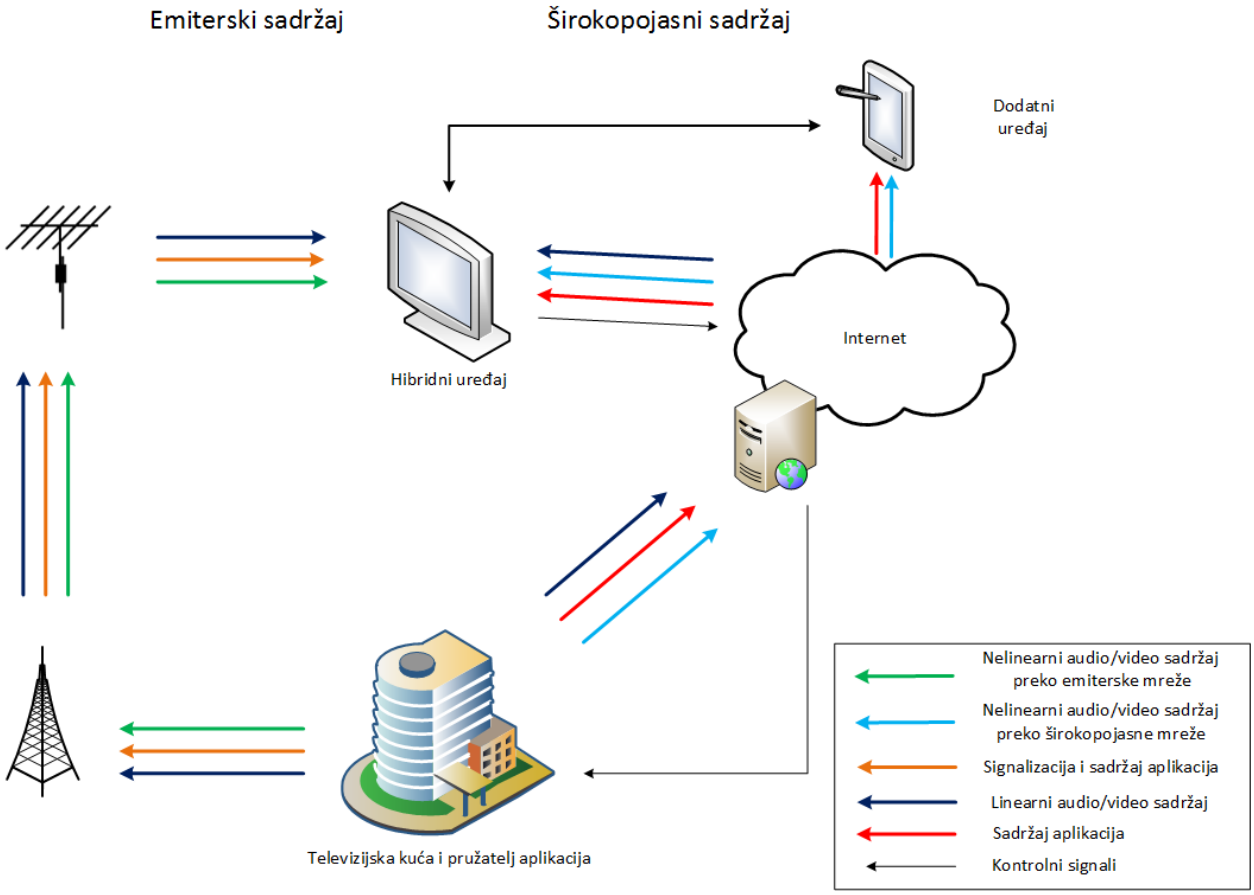
Hibridni uređaji koji podržavaju HbbTV omogućavaju korisnicima pristup svim ovim uslugama putem TV uređaja. Uz sve navedene usluge, korisnici mogu pristupati vlastitim sadržajima pohranjenim na vanjskom tvrdom disku ili u računalnom oblaku, i interaktivnim uslugama i internetskim aplikacijama [11].

2.3.1. Arhitektura HbbTV sustava

Hibridni uređaj se može spojiti na dvije mreže paralelno. Može biti spojen na DVB mrežu (npr. DVB-T, DVB-S ili DVB-C) za odašiljanje televizijskog signala. Putem te mreže hibridni uređaj može primati standardni linearni audio i video sadržaj, nelinearni audio i video sadržaj, signalizaciju i sadržaj aplikacija. Čak i ako uređaj nema širokopojasni pristup internetu, priključak na mrežu za emitiranje televizijskog signala omogućava primanje aplikacija vezanih uz emitirani sadržaj.

Dodatno, hibridni uređaj može biti spojen na internet širokopojasnom vezom. Ovime je omogućena dvosmjerna komunikacija s pružateljem usluga. Preko ovog sučelja, uređaj može primati sadržaj aplikacija i nelinearni audio i video sadržaj (kao što je video na zahtjev). Također putem ovog sučelja može biti podržano i preuzimanje audio i video sadržaja.

Širokopojasnim sučeljem se može povezati i na dodatne uređaje (eng. *Companion Screen, CS*), kao što su pametni telefoni ili tableti, ali i na druge HbbTV uređaje koji se nalaze na istoj lokalnoj mreži [12]. Primjer arhitekture prikazan je na slici 2.2.



Sl. 2.2. *Arhitektura HbbTV-a [12].*

2.3.2. HbbTV aplikacije

HbbTV aplikacije su bazirane na *web* aplikacijama i koriste postojeće *web* tehnologije: HTML, CSS i *JavaScript*. Ipak HbbTV aplikacije se razlikuju od uobičajenih *web* aplikacija. Kod razvoja moraju se uzeti u obzir različiti zahtjevi i ograničenja standarda. Za izradu se može koristiti *JavaScript* jezik kojim se proširuju mogućnosti aplikacija, što znači da je moguće pristupiti ugrađenim funkcionalnostima TV i STB uređaja. Moguće je pristupiti značajkama kao što su audio

i video reprodukcija. Također dostupne su i napredne opcije kao što je upravljanje sučeljem uređaja. Istovremeno je moguće prikazati samo jednu HbbTV aplikaciju na zaslonu uređaja. Pri izradi aplikacije je također potrebno voditi računa o tome da se ne zaključa pristup korisničkom sučelju ili ostatku sustava [12].

Potrebno je ispuniti osnovne zahtjeve kako bi HbbTV sustav radio ispravno:

1. Omogućavanje aplikacije: Aplikacije dobivene iz prijenosnog toka se automatski pokreću. Korisnik odlučuje hoće li se one prikazati na zaslonu korisnikovog uređaja. Kada je aplikacija dostupna prikazuje se obavijest na ekranu. Za prikaz aplikacije potrebno je pritisnuti crveni gumb na daljinskom upravljaču (engl. *red button application*). Pritiskom na gumb aplikacija se prikazuje na ekranu preko video sadržaja. Ponovnim pritiskom na gumb aplikacija se prestaje prikazivati na zaslonu korisnikovog uređaja.
2. Upravljanje aplikacijom: Upravljanje je uglavnom ograničeno na daljinski upravljač. Preporuča se upotreba strelica, tipke OK i brojeva. Za određene tipke aplikacija pokreće odgovarajuće događaje.
3. Pisanje efikasnih aplikacija: TV i STB uređaji nemaju visoke performanse pa aplikacije trebaju zahtijevati malo resursa. Ovisno o uređajima veličina aplikacije bi trebala biti između 2 i 4 megabajta [12].

2.3.3. Dodatni uređaji

Dodatni uređaji su uređaji koje korisnik posjeduje, kao što su pametni telefoni ili tableti. Ovi uređaji su obično spojeni na kućnu mrežu, a time i na internet. Na jednom dodatnom uređaju se može nalaziti jedna ili više aplikacija za dodatne uređaje (engl. *Companion Screen Application, CSA*) [13].

Dodatnim uređajem korisnik može pretraživati dostupni sadržaj i odabrati što želi gledati. Moguće je i upravljanje izvođenjem sadržaja na prijemniku putem dodatnog uređaja. Povezanost s društvenim vezama je također moguća i vrlo lako ostvariva. Neke od aplikacija koje bi se mogle pokretati na dodatnim uređajima su kvizovi i glasanja, koji bi pratili sadržaj koji korisnik trenutno gleda. Sadržaj koji korisnik gleda se može proširiti dodatnim informacijama koje bi se prikazivale na korisnikovom dodatnom uređaju. Za vrijeme gledanja nogometne utakmice na dodatnom uređaju bi se mogle prikazivati statistike kao što su posjed lopte, dodijeljeni kartoni i pretrčani kilometri nogometaša. Na slici 2.3. je prikazana aplikacija za dodatne uređaje njemačke televizijske kuće ARD. Ova aplikacija korisniku pruža informacije o emisijama: opis emisija, pregledavanje najava o emisijama, dodavanje podsjetnika i drugo.

S jednom HbbTV aplikacijom može biti povezano više aplikacija na dodatnim uređajima. Ipak, ne smije se pretpostavljati i praviti HbbTV aplikacije s pretpostavkom da korisnik posjeduje neki dodatan uređaj [12].



Sl. 2.3. *Aplikacija za dodatne uređaje televizijske kuće ARD [14].*

2.3.4. Komunikacija i sinkronizacija dodatnih uređaja

HbbTV aplikacija i dodatni uređaj mogu međusobno komunicirati direktno preko lokalne mreže ili putem interneta (iako komunikacija putem interneta nije definirana HbbTV standardom). Za međusobnu komunikaciju HbbTV aplikacija prvo mora saznati postoji li dodatan uređaj spojen na lokalnu mrežu. U tu svrhu koristi funkcije upravitelja dodatnim uređajima (engl. *HbbTV Companion Screen Manager*, *HbbTVCSManager* detaljnije objašnjen u poglavlju 4.3.), programskog sučelja, odnosno objekta sadržanog u sklopu HbbTV prijemnika. Implementacija samih funkcija ovog objekta je prepuštena proizvođačima HbbTV prijemnika. Kada je dodatni uređaj pronađen, HbbTV aplikacija može zatražiti pokretanje ili instaliranje aplikacije na dodatnom uređaju. Ovaj zahtjev se obavlja preko upravitelja dodatnih uređaja. Upravitelj ovaj zahtjev šalje aplikaciji za pokretanje koja se nalazi na dodatnom uređaju koja odlučuje hoće li pokrenuti odnosno instalirati aplikaciju.

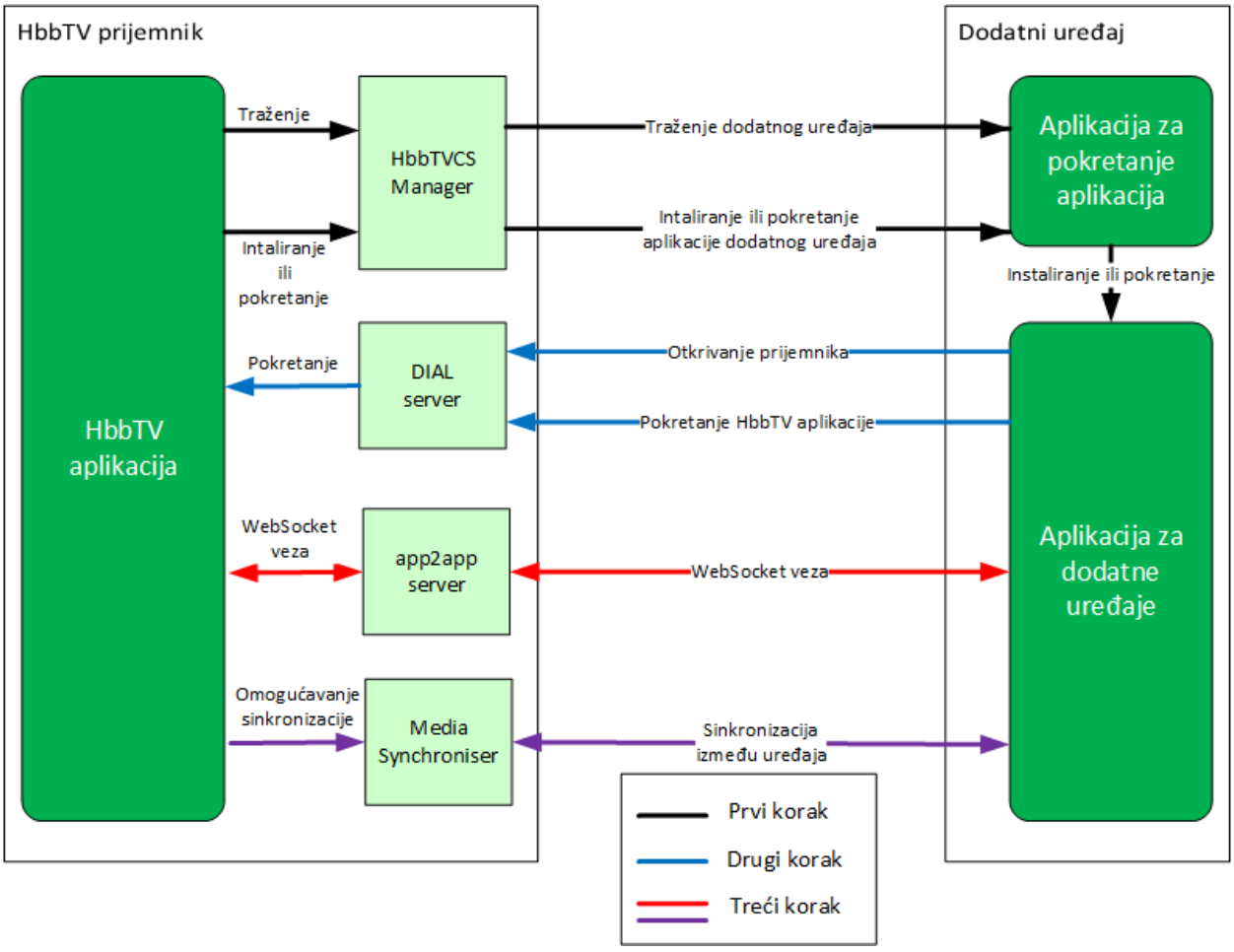
Nakon što je aplikacija na dodatnom uređaju pokrenuta, još uvijek se ne može spojiti na pristupne točke terminala jer joj nisu dostupne lokalne IP adrese terminala. Adrese pristupnih točki aplikacija može otkriti pomoću DIAL (engl. *DIscovery And Launch*) protokola. Osim dohvaćanja adresa pristupnih točki terminala, ovim protokolom je moguće iz aplikacije za dodatne uređaje poslati zahtjev za otvaranje HbbTV aplikacije na prijemniku. Nakon dohvaćanja adresa pristupnih točki, aplikacija na dodatnom uređaju se može spojiti na udaljenu pristupnu točku terminala. Ova pristupna točka je u sklopu servera za prosljeđivanje poruka (engl. *app2app server*) prijemnika koji preusmjerava poruke između aplikacije na dodatnom uređaju i HbbTV aplikacije u oba smjera. Kako bi poruke bile proslijeđene HbbTV aplikaciji, i ona mora biti spojena na server za prosljeđivanje poruka prijemnika. Na tom serveru se nalaze dva *WebSocket* servera odnosno pristupne točke, po jedna za HbbTV aplikaciju na samom uređaju i jedna za aplikaciju na dodatnom uređaju.

Osim servera za prosljeđivanje poruka, na prijemniku se nalaze i sučelja (*WebSocket* serveri) za sinkronizaciju sadržaja. Postoji više sučelja za sinkronizaciju sadržaja, ali dva su korištena u sklopu ovog rada. To su sučelje za identifikaciju sadržaja (engl. *Interface for Content Identification and other Information* – CSS-CII) i sučelje za sinkronizaciju (engl. *Interace for Timeline Synchronization* – CSS-TS). Adresa sučelja za identifikaciju sadržaja je također dohvaćena DIAL protokolom.

Aplikacija za dodatni uređaj se spaja na sučelje za identifikaciju sadržaja i dohvaća identifikator sadržaja (engl. *Content Identifier*) pomoću kojeg zna koji se vremenski osjetljiv sadržaj izvodi na prijemniku. Pomoću ovog sučelja aplikacija također dohvaća informacije o drugim sučeljima i njihove adrese. Nakon što dohvati identifikator sadržaja, aplikacija ga prosljeđuje sučelju za otkrivanje sadržaja (engl. *Interface for Material Resolution Service – CSS-MRS*). Sučelje za otkrivanje sadržaja vraća lokaciju sadržaja za dodatne uređaje koji odgovara vremenski osjetljivom sadržaju koji se izvodi na prijemniku. Sada aplikacija za koji sadržaj treba izvoditi na dodatnom uređaju, ali ga još mora uskladiti s sadržajem na prijemniku. Ovo se provodi putem sučelja za sinkronizaciju sadržaja. Aplikacija na dodatnom uređaju šalje sučelju za sinkronizaciju vremena kada može najranije i najkasnije izvoditi neki sadržaj. Prijemnik također sadržaj može izvoditi u određenom intervalu, odnosno postoje vremena kad najkasnije i najranije može izvoditi sadržaj. Sučelje uspoređuje vremena izvođenja prijemnika i dodatnog uređaja te izračunava vrijeme kada oba uređaja mogu izvoditi sadržaj. Ti se podaci tada šalju prijemniku i dodatnom uređaju. Dodatni uređaj ostaje spojen na sučelje za sinkronizaciju sadržaja te se sinkronizacija

obavlja u određenim vremenskim intervalima dokle god korisnik pregledava sadržaj i na dodatnom uređaju.

Međusobna komunikacija i sinkronizacija HbbTV prijemnika i dodatnog uređaja je ilustrirana na slici 2.4.



Sl. 2.4. *Uspostavljanje komunikacije između prijemnika i dodatnog uređaja.*

2.4. Sustav za testiranje HbbTV prijemnika

Svrha sustava za testiranje je omogućavanje testiranja uređaja, odnosno potencijalnog HbbTV prijemnika. Kako bi uređaj dobio HbbTV certifikat, mora proći sve skupove testova koje propisuje HbbTV udruženje.

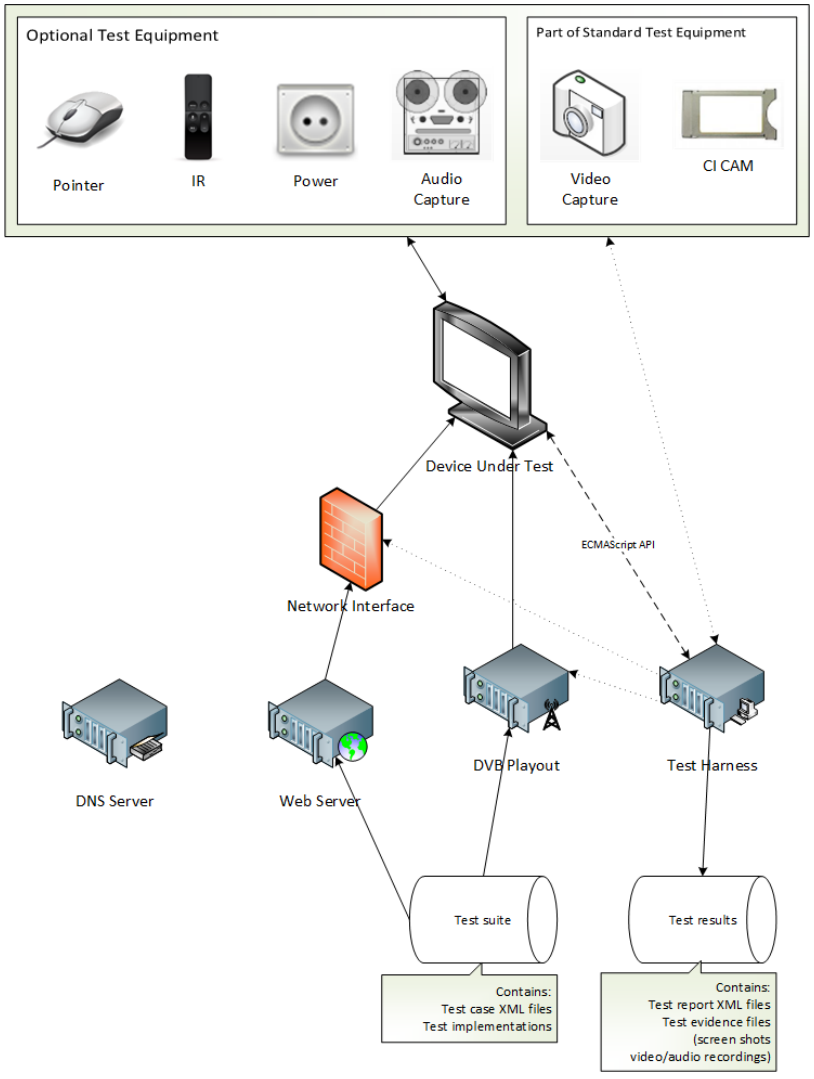
HbbTV sustav za testiranje se sastoji od:

- skupine prihvaćenih testnih slučajeva, koje pruža i održava HbbTV udruženje i

- testnog okruženja koje se koristi za provedbu testnih slučajeva i pohranu rezultata testiranja.

Samo testno okruženje čine (dano na slici 2.5.):

- uređaj koji se testira,
- sustav koji upravlja testnim slučajevima i rezultatima,
- standardna testna oprema obavezna za sva testna okruženja,
- dodatna testna oprema koja se može koristiti za pojednostavljenje ili automatizaciju nekih zadataka sustava za testiranje te
- infracrvena i internet veza između uređaja koji se testira i drugih stavki testnog okruženja.



Sl. 2.5. *Testno okruženje.*

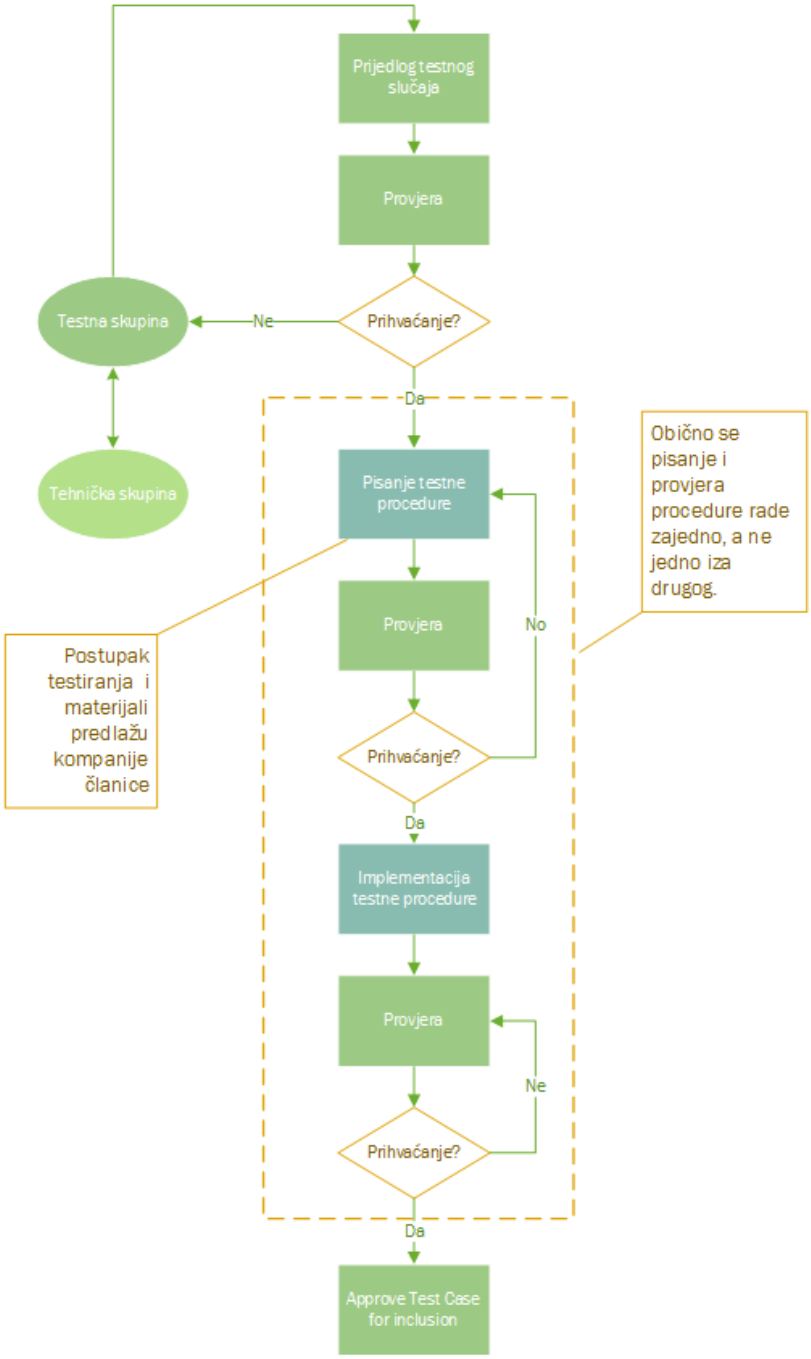
2.4.1. Testni slučaj

Skup HbbTV testova sastoji se od više testnih slučajeva. Svaki testni slučaj je jedinstveno određen i cilj mu je provjeriti određeni zahtjev HbbTV specifikacije. Svaki se test sastoji od više datoteka koje se nalaze u mapi koja nosi naziv samog testa:

- XML (engl. *eXtensible Markup Language*) datoteka testnog slučaja. Višenamjenski dokument u kojem se nalazi:
  - reference na specifikaciju i primjenjivost testa,
  - opis testa,
  - procedura izvršavanja i očekivani rezultati,
  - zapis o razvoju testa,
- datoteke s implementacijom testa (ako više testova koristi iste datoteke, one se mogu smjestiti u zajedničku mapu),
- konfiguracijske datoteke potrebne za izvršavanje testnih slučajeva na sustavu za testiranje,
- informacije o licenci i dostupnosti testova.

Testni slučajevi iz skupa testova mogu pripadati jednoj od dvije glavne grupe: prihvaćeni testni slučajevi ili dodatni testni slučajevi. Prihvaćeni testni slučajevi su oni koje je odobrilo HbbTV udruženje i koji su u skladu s HbbTV zahtjevima. Dodatni testni slučajevi koje nije odobrilo HbbTV udruženje su dostupni jer bi mogli biti od koristi programerima, korisnicima testova. Dodatni testni slučajevi mogu biti prihvaćeni ako ispunjavaju HbbTV zahtjeve.

Postupak izrade testnog slučaja je ilustriran na slici 2.6.



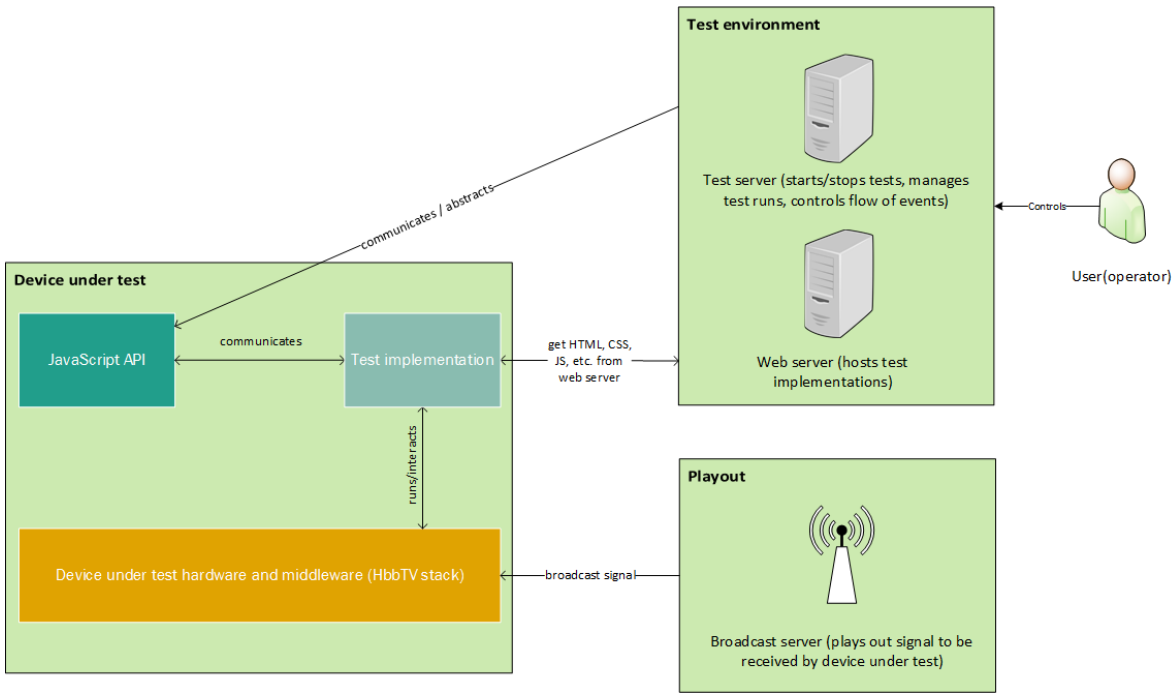
Sl. 2.6. *Postupak izrade testnog slučaja.*

2.4.2. Testno okruženje

Testno okruženje za provođenje skupine HbbTV testova na uređaju koji se testira se sastoji od dvije glavne komponente: standardne testne opreme i sustava za testiranje.

Standardna testna oprema odnosi se na skupinu uobičajenih alata koji su potrebni za pohranu, dostavu, generiranje i izvođenje testova na uređaju. Ovo uključuje *web* server i sustav za puštanje DVB signala.

Sustav za testiranje je sustav koji upravlja odabirom i izvršavanjem testova na testnom uređaju, kao i skupljanjem rezultate testova za izvještaj. Prikaz sustava za testiranje, uređaja koji se testira i međusobne komunikacije se nalazi na slici 2.7.



Sl. 2.7. *Testno okruženje i uređaj koji se testira*

Sustav za testiranje koristi informacije iz XML datoteke s opisom testa i iz drugih testnih materijala kako bi izvršio sve potrebne korake prije početka testa. Također pokreće izvršavanje testnog slučaja na uređaju koji se testira, pravi promjene u okruženju u određenim vremenskim trenucima, kako je opisano u testnim materijalima i s obzirom na pozive samo testnog slučaja. Sustav za testiranje skuplja rezultate testnog slučaja pokrenutog na uređaju koji se testira.

3. PROGRAMSKA SUČELJA POTREBNA ZA REALIZACIJU TESTOVA

U ovom poglavlju su opisana programska sučelja (engl. A*pplication Programming Interface, API*) i objekti izrađeni u sklopu ovog rada koji su potrebni za realizaciju HbbTV testova. Prvo je opisano programsko sučelje za *WebSockete* čija je najbitnija funkcionalnost multipleksiranje velikog broja zahtjeva i veza putem jedne *WebSocket* veze. Zatim su detaljnije objašnjene pristupne točke terminala za komunikaciju između HbbTV aplikacija i aplikacija na dodatnim uređajima. Osim pristupnih točki za komunikaciju između aplikacija opisane su i pristupne točke za identifikaciju i sinkronizaciju sadržaja. Na kraju poglavlja su opisani i emulirani objekti prijemnika.

WebSocket programsko sučelje je dio sustava za testiranje, a sve navedene pristupne točke i objekti bi se zapravo trebali nalaziti u sklopu HbbTV prijemnika. Budući da se testovi izrađuju na računalima, te se njihova ispravnost provjerava na računalima, funkcionalnosti HbbTV prijemnika (pristupne točke i objekti) su emulirane na računalu.

3.1. WebSocket programsko sučelje

Kako je već naznačeno u potpoglavlju 2.2.3., HbbTV koristi *WebSockete* za komunikaciju između aplikacija na terminalu i aplikacija na dodatnim uređajima, kao i za sinkronizaciju multimedijskog sadržaja na dodatnim uređajima. U sklopu ovog rada napravljeno je posebno *WebSocket* programsko sučelje zbog specifičnih zahtjeva HbbTV specifikacije. Također, postoje neki detalji koje testni slučajevi moraju moći nadzirati i testirati:

- Slanje *ping* okvira i primanje *pong* okvira
- Testiranje posebnih zaglavlja zahtjeva
- Upravljanje fragmentacijom prenesenih poruka
- Statusni HTTP kod kad je *WebSocket* veza odbijena

Osim navedenih detalja, potrebno je omogućiti testnim slučajevima otvaranje barem 400 *WebSocket* veza odjednom. U internetskom pregledniku nije moguće istovremeno otvoriti tako velik broj veza pa se to rješava u programskom sučelju multipleksiranjem.

WebSocket programsko sučelje je podijeljeno na dva dijela: klijentsku i serversku stranu. Serverska strana programskog sučelja je *NodeJS* aplikacija. U aplikaciji se nalazi *WebSocket* server na koji se spaja klijentska strana. Serverska strana također otvara nove *WebSocket* veze po

primitku zahtjeva za otvaranje veze. Klijentsku stranu predstavlja *JavaScript* biblioteka u kojoj se nalazi funkcija za otvaranje *WebSocket* veze. Pri uključivanju biblioteke u HTML datoteku testnog slučaja, otvara se jedna *WebSocket* veza prema serverskoj strani programskog sučelja. Putem te jedne otvorene veze serveru se šalju zahtjevi za otvaranje novih *WebSocket* veza. U tom zahtjevu uključena je adresa *WebSocket* servera na koji se potrebno spojiti. Serverska strana programskog sučelja se kao klijent spaja na *WebSocket* server koji se nalazi na adresi zaprimljenoj u zahtjevu.

Pri pozivu funkcije za otvaranje veze, stvara se objekt u kojem se pohranjuju svi parametri predani funkciji. Osim tih parametara objektu se dodaje i jedinstveni identifikator. Identifikator se šalje u zahtjevu za otvaranje veze i pri svakom slanju poruke prema serverskoj strani programskog sučelja. Kada primi zahtjev za otvaranje veze, server pohrani novootvorenu vezu zajedno s jedinstvenim identifikatorom i šalje ga pri svakom slanju poruke sa servera. Identifikator je ono po čemu se razlikuju sve otvorene veze. Kada se šalje poruka prema serverskoj strani, server pomoću identifikatora zna na koju otvorenu *WebSocket* vezu treba proslijediti poruku. Isto tako kada se prima poruka sa serverske strane, dolazi u paketu s identifikatorom. Izračun identifikatora je prikazan na slici 3.1.

```
var client_id = Date.now() + Math.random() + Math.random();

var obj = {
    "topic": "open-ws-connection",
    "payload": {
        "client_id": client_id,
        "url": url,
        "originHeader": originHeader,
        "websocketExtensionHeader": websocketExtensionHeader
    }
};
```

Sl. 3.1. *Računanje identifikatora i objekt koji se šalje u zahtjevu.*

Na slici 3.1. se također vidi objekt koji se šalje serverskoj strani u zahtjevu za otvaranje veze. Objekt sadrži temu (engl. *topic*) kako bi server znao što točno treba napraviti sa zahtjevom. Osim otvaranje veze, tema zahtjeva može biti poruka koju server treba proslijediti, *ping* zahtjev, zahtjev za zatvaranje veze ili zahtjev za zatvaranje TCP veze na kojoj je uspostavljena *WebSocket* veza. Kod zahtjeva za zatvaranje veze obavlja se zatvaranje veze definirano *WebSocket* specifikacijom. Šalje se zahtjev za zatvaranje veze, te se nakon toga čeka poziv povratne funkcije (engl. *callback*)

da je zatvaranje veze uspješno završeno. Kod zahtjeva za zatvaranje TCP veze samo se prekine TCP veza i poziv povratne funkcije se ne čeka niti je više moguć.

Nakon slanja zahtjeva za otvaranje veze i uspješno otvorene *WebSocket* veze, poziva se povratna funkcija. U toj povratnoj funkciji se vraća *WebSocket* objekt u kojem se nalaze metode za slanje poruke, slanje *ping* okvira, zatvaranje veze i zatvaranje TCP veze. Osim slanja tekstualne poruke, moguće je poslati i poruku s binarnim podacima.

3.2. Pristupne točke terminala

Terminal treba imati dvije pristupne točke za međusobnu komunikaciju aplikacija koje trebaju biti serverska strana *WebSocket* protokola:

- Lokalnu pristupnu točku na koju se povezuju HbbTV aplikacije koje se nalaze na samom terminalu
- Udaljenu pristupnu točku na koju se povezuju aplikacije na drugim uređajima u lokalnoj mreži, uključujući aplikacije na dodatnim uređajima koje korisnik posjeduje ili aplikacije koje se izvode na drugim HbbTV terminalima

HbbTV aplikacije se povezuju na lokalnu pristupnu točku terminala na kojem su pokrenute, ili na udaljenu pristupnu točku drugog terminala na istoj lokalnoj mreži. Aplikacije na dodatnim uređajima se povezuju na udaljene pristupne točke terminala.

Terminal treba podržavati minimalno deset istovremenih *WebSocket* veza lokalnoj pristupnoj točki i isto toliko na udaljenoj pristupnoj točki.

Lokalna i udaljena pristupna točka su realizirane u obliku zasebne *Node.js* aplikacije. Svaka točka posebno je realizirana kao *WebSocket* server. Kada se HbbTV aplikacija ili aplikacija na dodatnom uređaju spoji na jednu od pristupnih točki, npr. HbbTV aplikacija lokalnu pristupnu točku, stavlja se na stog. Tada aplikacija za pristupne točke čeka da se druga aplikacija spoji na drugu pristupnu točku (u ovom slučaju aplikacija na dodatnom uređaju na udaljenu pristupnu točku). Provodi se postupak provjere postoji li aplikacija spojena na lokalnu pristupnu točku. Ako postoji, skida se sa stoga, i uparuje se s aplikacijom spojenom na lokalnu pristupnu točku. Nakon spajanja, šalje se poruka da je uparivanje uspješno završeno (engl. *pairingcompleted*) prema obje spojene aplikacije. Kada je uparivanje uspješno završeno, poruke primljene na jednoj pristupnoj točci se preusmjeravaju na drugu odgovarajuću uparenu pristupnu točku.

Unutar aplikacije za pristupne točke se također nalazi jednostavan REST (engl. *Representational State Transfer*) server koji služi za konfiguriranje aplikacije. Slanjem objekta s odgovarajućim parametrima na server moguće je ugasiti ili ponovno pokrenuti aplikaciju za pristupne točke. Također je moguće i konfigurirati aplikaciju za pristupne točke da ne provjerava zaglavlje zahtjeva kod spajanja HbbTV aplikacija ili aplikacija na dodatnim uređajima na pristupne točke. Konfiguriranje je napravljeno zbog specifičnih zahtjeva pojedinih testnih slučajeva i testiranja pravilnog rada testnih slučajeva.

3.3. Sučelja za identifikaciju sadržaja i sinkronizaciju

Sučelja korištena u sklopu ovog rada su sučelje za identifikaciju sadržaja i sučelje za sinkronizaciju. Osim ovih sučelja, postoje i druga sučelja sa sličnom svrhom koja moraju biti implementirana u sklopu HbbTV prijemnika.

Sučelja za identifikaciju sadržaja i za sinkronizaciju su napravljena kao *WebSocket* serveri. Nisu u posebnoj aplikaciji, nego u sklopu glavne aplikacije sustava za testiranje. Nakon što se klijent spoji na sučelje za identifikaciju sadržaja, sučelje odmah klijentu pošalje JSON (engl. *JavaScript Object Notation*) objekt s podacima. Taj objekt sadržava: identifikator sadržaja, status izvođenja sadržaja, adresa sučelja sa sinkronizaciju, adresa sučelja za otkrivanje sadržaja i drugo. U implementaciji za potrebe rada se ne šalju svi podaci.

Sučelje za sinkronizaciju je također *WebSocket* server napravljen u istoj datoteci. Ovo sučelje treba vraćati različite podatke za različite testove. Kako bi se to postiglo omogućena je konfiguracija sučelja. Konfiguracija se vrši spajanjem na sučelje i slanjem objekta s odgovarajućim sadržajem, „naslovom“ objekta. Ako primljeni objekt ima odgovarajući sadržaj, pročitaju se njegovi podaci i postave za slanje klijentima koji se spoje. Ako stigne objekt čiji sadržaj nije konfiguracija, znači da se radi o klijentu kojemu se šalju postavljeni podaci.

3.4. Emulirani objekti prijemnika

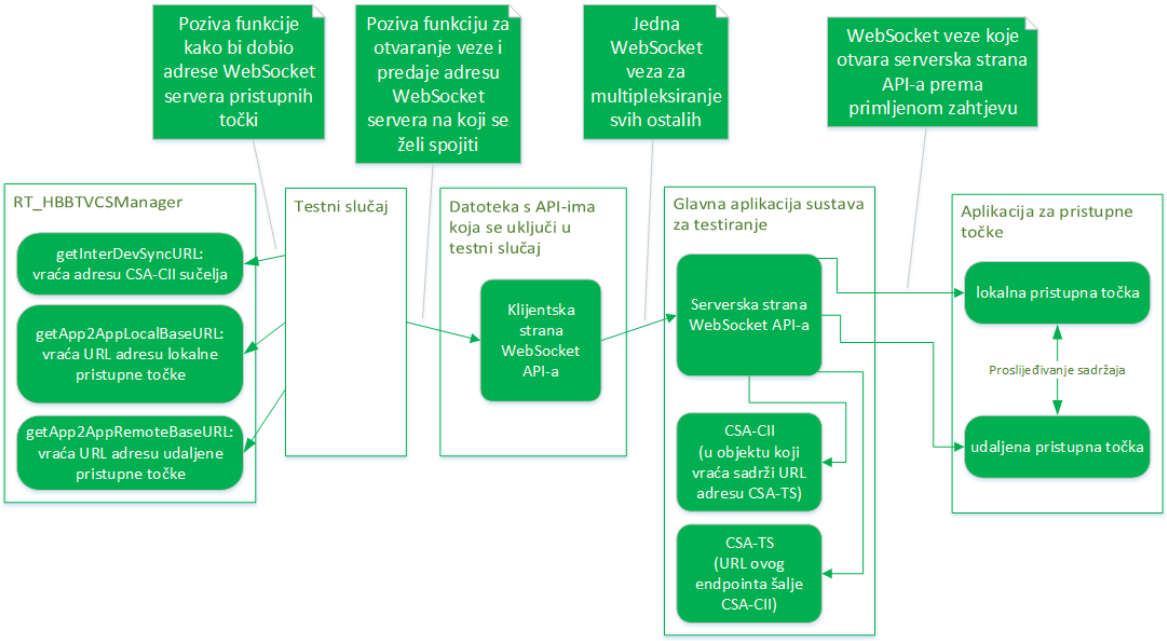
Ovi objekti emuliraju rad HbbTV prijemnika na način da vraćaju vrijednosti koje bi vratio sam prijemnik. Ovi objekti se inače nalaze u sklopu prijemnika, ali budući da se testovi izrađuju na računalu, bilo ih je potrebno emulirati u obliku *JavaScript* objekata. Objektom za sinkronizaciju medijskog sadržaja (*RT_MediaSynchroniser*) se upravlja sadržajem predviđenom za sinkronizaciju. Definirane su metode za inicijalizaciju samog objekta, dodavanje i uklanjanje sadržaja u odnosu na koji se vrši sinkronizacija, omogućavanje i onemogućavanje sinkronizacije.

RT_HbbTVCSManager objekt služi za upravljanje dodatnim uređajima. Sve njegove funkcionalnosti, odnosno metode su dane u tablici 3.1.

Tab. 3.1. *Sve funkcije RT_HbbTVCSManager objekta.*

Funkcija	Opis funkcije
discoverCSLaunchers	Vraća sve pronađene aplikacije za pokretanje aplikacija dodatnih uređaja na lokalnoj mreži
discoverTerminals	Vraća sve druge HbbTV prijemnike pronađene na lokalnoj mreži.
launchCSApp	Pokreće ili instalira aplikaciju na dodatnom uređaju
getInterDevSyncURL	Vraća adresu sučelja za identifikaciju sadržaja
getAppLaunchURL	Vraća adresu pristupne točke za pokretanje aplikacija prijemnika na kojem se izvršava HbbTV aplikacija koja je uputila poziv funkcije
getApp2AppLocalBaseURL	Vraća adresu lokalne pristupne točke
getApp2AppRemoteBaseURL	Vraća adresu udaljene pristupne točke

RT_HbbTVCSManager, programsko sučelje, pristupne točke i druga sučelja su prikazana na slici 3.2.



Sl. 3.2. *Programska sučelja, pristupne točke i RT_HbbTVCSManager.*

4. POSTUPAK IZRADE I PRIMJER TESTA

HbbTV testni slučajevi odnosno testovi su podijeljeni u skupine. Za potrebe verifikacije sinkronizacije više uređaja u HbbTV okruženju izrađeni su testovi iz dvije skupine testova: *MDEVSYNC* i *APP2APP*. *MDEVSYNC* testovima se provjerava uspješna sinkronizacija multimedijskog sadržaja na HbbTV prijemniku i ostalim korisnikovim uređajima. *APP2APP* skupina sadrži testove kojima se provjerava je li komunikacija između HbbTV aplikacije i aplikacije na dodatnom uređaju uspješna i odvija li se po zadanim pravilima HbbTV standarda. Skupine testova mogu biti podijeljene u podskupine. U sklopu ovog rada napravljeni su *APP2APP* testovi iz podskupine *APP2APP_02*. Nakon izrade HbbTV testova napravljeni su i T2Unit testovi koji provjeravaju ispravan rad HbbTV testova.

4.1. Postupak izrade testa

Izrada testa započinje proučavanjem XML datoteke s opisom testa (engl. *assertion*). U toj datoteci se nalazi naslov testa, opisano je što se točno i na koji način testira. Dane su i reference na dio HbbTV specifikacije gdje je opisan ispravan rad funkcionalnosti koja se testira. U datoteci je sadržan i kratak opis testa (engl. *assertion text*). Nakon izrade testa, u ovoj je datoteci potrebno opisati na koji je način realiziran test odnosno opisati pojedinačno svaki korak testa.

Sam test se piše u HTML-u u jednoj HTML datoteci. Nacrt testa je prikazan na slici 4.1.

```
<html>
<head>
    <script Biblioteke potrebne za test> </script>
    <title>Naslov testa</title>
    <script type="text/javascript">
    // Logika testa u JavaScriptu
        window.onload = function() {
            asyncWaterfall( //Funkcija koja slijedno poziva korake testova
                [step1, step2,step3,...,stepN]
            );
        };
        function step1() {
            // Radi nešto
            if(sth){ //Provjera
                // Prvi korak uspješan
            }
            else {
                // Greška u prvom koraku
            }
        }
        function step2() {
            ...
        }
        ...
    </script>
</head>
<body>
    <object type="application/hbbtvCSManager" id="cs-man"></object>
</body>
</html>
```

Sl. 4.1. *Nacrt jednog HbbTV testa u HTML-u.*

Na početku testa u zaglavlje se uključuju *JavaScript* biblioteke potrebne za uspješno izvršavanje testa. U oznaci za naslov je zapisan naslov testa. Svaki korak testa je napravljen kao zasebna *JavaScript* funkcija. Koraci, odnosno funkcije se slijedno pozivaju i izvršavaju. Ako se dogodi pogreška u nekom koraku, izvođenje testa se prekida i u sustavu za testiranje se ispisuje poruka s opisom pogreške i brojem koraka u kojem se dogodila pogreška. Ako se svi koraci uspješno izvedu, bez pogreške, test je uspješno prošao.

Nakon izrade testa, potrebno je provjeriti ispravan rad istoga. U tu svrhu se pišu posebni testovi koji će provjeriti rad HbbTV testa. Ti testovi se nazivaju T2Unit testovi. Cilj T2Unit testova je izazivati pogreške i provjeravati hoće li HbbTV test prekinuti izvođenje testa na mjestu pogreške s odgovarajućom porukom za izazvanu pogrešku. Ako HbbTV test prekine izvršavanje na odgovarajućem mjestu s odgovarajućom porukom, T2Unit test je prošao. Ako izvršavanje testa ne bude prekinuto te se test nastavi ili se prekine na nekom drugom mjestu s nekom drugom porukom koja nije očekivana, T2Unit test nije prošao. Jednim T2Unit testom se provjerava jedna grana HbbTV testa, odnosno izaziva se jedna pogreška u testu. To znači da je za jedan HbbTV test potrebno više T2Unit testova, odnosno onoliko koliko ima mogućih pogrešaka u HbbTV testu i još jedan dodatni za uspješno izvršen test bez pogreške. Svi T2Unit testovi za jedan HbbTV test se grupiraju u *TEST_PLAN*. To je JSON datoteka u kojoj je zapisan naziv plana, popis T2Unit testova, naziv HbbTV testa na kojeg se odnose navedeni T2Unit testovi i mapa u kojoj se isti nalazi.

4.2. Primjer testa iz skupine APP2APP

Testovi iz skupine APP2APP služe za provjeru komunikacije između HbbTV aplikacije na prijemniku i aplikacije na dodatnom uređaju. Naziv jednog HbbTV testa je sastavljen od imena skupine testova iz koje je test i četiri znamenke. Test koji će ovdje biti objašnjen je APP2APP0378.

```
<?xml version="1.0" encoding="UTF-8"?>
<testCase xmlns="http://www.hbbtv.org/2012/B/testCase" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
id="org.hbbtv_APP2APP0378" version="1">
    <originInformation>···
    </originInformation>
    <title>App2App - 10 pairings - 25 small messages per pairing in 10 sec to remote end-point</title>
    <appliesTo><spec name="HBBTV" version="1.3.1"/></appliesTo>
    <specReference name="HBBTV" version="1.3.1">···
    </specReference>
    <assertionText><![CDATA[ When a HbbTV application has 10 paired connections  with 10 companion screen
    applications, and each companion screen application sends a binary message with a size of 512 bytes every
    400 ms over a period of 60 seconds the terminal immediately relays the binary message to the HbbTV
    application via the corresponding connection.]]></assertionText>
```

Sl. 4.2. *Isječak iz XML datoteke s opisom testa.*

Naslov i opis testa su vidljivi na slici 4.2. Prijevod opisa testa glasi: „Kada HbbTV aplikacija ima 10 uparenih veza s 10 aplikacija na dodatnim uređajima i svaka aplikacija na dodatnim uređajima šalje binarne poruke veličine 512 bajta svakih 400 milisekundi tijekom 60 sekundi, terminal te poruke preusmjerava HbbTV aplikaciji preko odgovarajuće veze.“

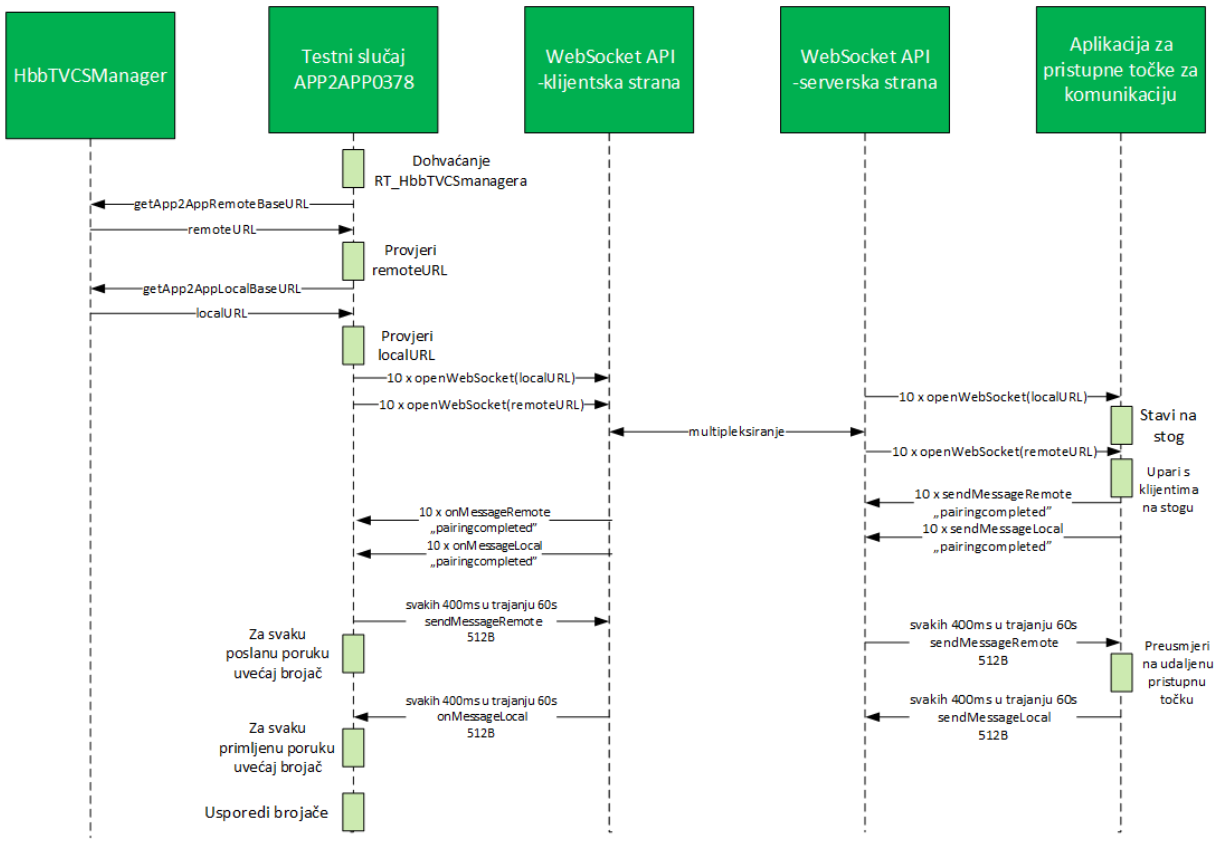
Kako bi se HbbTV aplikacije i aplikacije na dodatnim uređajima mogle spojiti na lokalnu i udaljenu pristupnu točku, potrebno je znati adrese pristupnih točki. Prvi korak testa je dohvaćanje adrese udaljene pristupne točke. Adrese se mogu dobiti pozivanjem metoda iz *RT_HbbTVCSManager* objekta. Ovaj objekt se uključi u test unutar objektne HTML oznake. Kako bi se mogao koristiti u *JavaScript* kodu, potrebno je dohvatiti referencu na taj objekt i pohraniti. Kada je referenca na objekt dohvaćena, moguće je pozvati metodu iz objekta koja vraća adresu udaljene pristupne točke i pohraniti je u varijablu. Poziv metode se u kodu nalazi unutar *try-catch* bloka, kako bi se uhvatile moguće pogreške i zaustavilo izvođenje testa. Zatim se provjerava vrijednost adrese. Ako je vrijednost adrese *null* ili *undefined,* test se prekida. Adresa udaljene pristupne točke je zapravo adresa *WebSocket* servera što znači da prvih nekoliko znakova adrese mora biti „ws://“ pa se i to provjerava. Ako je i ova zadnja provjera uspješna, prvi korak testa je uspješno završen.

Drugi korak test je dohvaćanje adrese lokalne pristupne točke. Ovaj korak je analogan prvom koraku pa neće biti detaljno opisan. Veća razlika je u tome što je referenca na *RT_HbbTVCSManager* već pohranjena u globalnoj varijabli pa ju nije potrebno dohvaćati.

U trećem koraku se uspostavlja veza iz HbbTV aplikacija prema lokalnoj pristupnoj točki. Kako uspostavljanje veze prema pristupnoj točki nije ništa drugo nego otvaranje klijentske veze prema *WebSocket* serveru, te se veze otvaraju iz samog testa koji u ovom slučaju predstavlja HbbTV aplikaciju na terminalu. Za otvaranje veze koristi se funkcija iz *WebSocket* programskog sučelja. Poziv ove funkcije se nalazi u petlji u kojoj se funkcija poziva deset puta. Osim adrese *WebSocket* servera, odnosno adrese lokalne pristupne točke, kao parametri funkcije navode se i funkcije koje će se pozvati: nakon uspješnog spajanja, po primitku poruke i ako dođe do greške prilikom otvaranje veze. Ako se dogodi pogreška prilikom otvaranja veze, pozvat će se predana funkcija u kojoj će se izvođene testa prekinuti s odgovarajućom porukom. Funkcija koja se poziva po primitku poruke provjerava ispravnost primljene poruke i ako je veličina poruke jednaka 512 bajta, brojač primljenih poruka se uvećava za jedan. U funkciji koja se poziva nakon uspješnog otvaranja

veze se nalazi brojač koji se uvećava pri svakom pozivu te funkcije. Nakon uvećavanja, vrijednost brojača se provjerava i ako je jednaka 10, treći korak je uspješno završen.

U četvrtom koraku se uspostavlja veza iz aplikacija na dodatnim uređajima prema lokalnoj pristupnoj točki. Analogno trećem koraku gdje se u testu otvaraju *WebSocket* veze umjesto u HbbTV aplikacijama i u četvrtom koraku se otvaraju u testu umjesto iz aplikacija na dodatnim uređajima. Ako se dogodi pogreška pri otvaranju veze, također će se prekinuti izvođenje testa. U funkciji koja se poziva nakon uspješnog otvaranje veze, također se broji koliko je veza otvoreno. Kada se otvori svih 10 veza, postavi se brojač vremena koji će prijaviti grešku unutar deset sekundi. Ovaj brojač se briše u funkciji koja se poziva pri primitku poruke. Briše se kada je primljeno deset poruka sadržaja „*pairingcompleted*“ što znači da su sve veze uspješno uparene. Osim toga postavlja se novi brojač vremena od 60 sekundi. Tijekom tog vremena šalju se binarne poruke veličine 512 bajta, svakih 400 milisekundi putem svake uparene veze. Po isteku 60 sekundi, provjerava se jeli broj poslanih poruka jednak broju ispravnih primljenih poruka. Ako je jednak, test je uspješno prošao, u suprotnom test je pao. Dijagram koji prikazuje sve korake testa se nalazi na slici 4.3.



Sl. 4.3. *Dijagram APP2APP0378 testa.*

4.3. Primjer T2Unit testa

Primjer će biti T2Unit test za jednu granu opisanog testa APP2APP0378. Izgled T2Unit testa je prikazan na slici 4.4.

```
var EXPECTED_COMMENT = "Message sent and message received counts are not equal";

T2UNIT.loadFixtures("step4_capabilities_messagecount.js");

T2UNIT.on("reportStepResult", function(step, result, comment) {
    console.log('\n\nSTEP: \n\n'  + step);

    if (!result && step === 4 && comment === EXPECTED_COMMENT) {
        T2UNIT.finished(true,"Test case failed in step "+step+" as expected. Comment: " + comment);
    }
    if (result && step >= 4) {
        T2UNIT.finished(false,"Unit test fails because Test case didn't failed as expected in step 4. ");
    }
});
```

step4_capabilities_messagecount.js ●

```
1   function breakfunction() {
2         messageSentCount++;
3   }
4
5   applyFixtureAfterStep(3, breakfunction);
```

Sl. 4.4. *T2Unit test za APP2APP0378.*

Cilj ovog T2Unit testa je zapravo izazvati pogrešku kod provjere jeli broj poslanih poruka jednak broju primljenih poruka. Kod koji izaziva pogrešku se nalazi u posebnoj datoteci. Taj kod je umetnut u test koji se izvodi. Pogreška je izazvana tako da je globalna varijabla u koju je pohranjen broj poslanih poruka povećana za jedan što će dovesti do pada testa prilikom provjere broj poslanih i primljenih poruka.

Za prolaz ovog T2Unit testa, HbbTV APP2APP0378 test mora pasti prilikom provjere broja poslanih poruka i primljenih s očekivanim komentarom. Ako se APP2APP0378 test završi uspješno ili neuspješno s nekom drugom pogreškom osim očekivane, T2Unit test je pao.

5. ZAKLJUČAK

U ovom radu su obrađene bitne teorijske osnove HbbTV-a. Opisana je ukratko arhitektura HbbTV-a i dodatnih uređaja, te što su HbbTV aplikacije i koje su njihove karakteristike. Objašnjeno je što su dodatni uređaji, njihova uloga u HbbTV sustavu i koje mogućnosti pružaju. Također je opisan način na koji dodatni uređaji komuniciraju s HbbTV prijemnikom na lokalnoj mreži i kako se provodi sinkronizacija sadržaja na dodatnim uređajima sa sadržajem na HbbTV prijemniku.

Kako bi se mogli napraviti testni slučajevi, napisano je programsko sučelje za *WebSockete* kao i objekti i ostale potrebne funkcije za izradu testova. Sve navedeno je sada dio sustava za testiranje i može se koristiti za izradu sličnih HbbTV testova i provođenje testiranja uređaja.

Napravljen je dio testova iz MDEVSYNC paketa kojima se provjerava sinkronizacija medijskog sadržaja na više uređaja i svi testovi iz APP2APP paketa kojima se provjerava komunikacija između aplikacija na različitim uređajima, kao i T2Unit testovi za sve navedene testove. Pokretanjem testova i T2Unit testova je provjeren njihov ispravan rad te su uočene pogreške ispravljene.

U objektima *RT_MediaSynchroniser* i *RT_HbbTVCSManager* su moguće izmjene i dopune. Te eventualne dopune zajedno s programskim sučeljem za *WebSockete* su jedan od preduvjeta za izradu preostalih testova iz *MDEVSYNC* paketa.

LITERATURA

[1] https://hr.wikipedia.org/wiki/Televizija [19.8.2016.]

[2] https://hr.wikipedia.org/wiki/Internet [19.8.2016.]

[3] https://www.w3.org/TR/REC-html40-971218/intro/intro.html#h-2.2 [22.8.2016.]

[4] https://hr.wikipedia.org/wiki/CSS [22.8.2016.]

[5] http://www.znanje.org/knjige/computer/JavaScript/2010/index.htm [23.8.2016.]

[6] http://www.mathos.unios.hr/wp/wp2009-10/P8_Java.pdf [23.8.2016.]

[7] https://tools.ietf.org/html/rfc6455#page-4 [22.8.2016]

[8] https://www.fer.unizg.hr/_download/repository/11._predavanje%5B7%5D.pdf [24.8.2016]

[9] http://blog.king-ict.hr/hello-nodejs [24.8.2016]

[10] https://www.hbbtv.org/overview/ [18.8.2016]

[11] https://en.wikipedia.org/wiki/Hybrid_Broadcast_Broadband_TV [21.8.2016.]

[12] Hybrid Broadcast Broadband TV, ETSI TS 102 796 V1.3.1.

[13] Digital Video Broadcasting (DVB);Companion Screens and Streams;Part 1: Concepts, roles and overall architecture, ETSI TS 103 286-1 V1.1.1

[14] http://programm.ard.de/TV/ARD-EPG---HbbTV/ARD-EPG [26.8.2016.]

KRATICE

HbbTV	Hybrid broadcast broadband Television
HTML	HyperTekst Markup Language
TV	Televizija
IP	Internet Protocol
CSS	Cascading Style Sheets
TCP	Transmission Control Protocol
HTTP	HyperText Transfer Protocol
STB	Set-Top Box
DVB-T,C,S	Digital Video Broadcasting – Terrestrial, Cable, Satellite
CS	Companion Screen
CSA	Companion Screen Application
DIAL	DIscovery and Launch
CSS-CII	Interface for Content Identification and other Information
CSS-TS	Interace for Timeline Synchronization
CSS-MRS	Interface for Material Resolution Service
XML	eXtensible Markup Language
API	Application Programming Interface
REST	Representational State Transfer
JSON	JavaScript Object Notation

SAŽETAK

U ovom radu su dane osnovne informacije o HbbTV standardu s naglaskom na dodatne uređaje koje standard podržava. Spomenute su i korištene *web* tehnologije: HTML, *JavaScript*, *WebSocketi* i *NodeJS*.

Cilj rada je bio izraditi programska sučelje, objekte i funkcije potrebne za realizaciju HbbTV testova za verifikaciju sinkronizacije više uređaja u HbbTV okruženju. Po izradi navedenog napravljeni su i testni slučajevi iz MDEVSYNC i APP2APP paketa kao i T2Unit testovi koji provjeravaju njihov ispravan rad.

Ključne riječi: HbbTV, sustav za testiranje, testni slučajevi, programsko sučelje

ABSTRACT

Device synchronization verification for HbbTV systems

This paper contains basic information about HbbTV standard and especially about Companion screen devices which standard defines. Used web technologies were also mentioned: HTML, JavaScript, WebSockets and NodeJS.

The aim of this thesis was to make API-s, objects and functions needed to make HbbTV test cases that verify multiple devices synchronization in HbbTV environment. After that, test cases from MDEVSYNC and APP2APP packages and T2Unit tests for verifying MDEVSYNC and APP2APP tests were made.

Key words: HbbTV, test Harness, test cases, application programming interface

ŽIVOTOPIS

Zvonimir Ivešić rođen je 14. rujna 1992. godine u Đakovu. Osnovnu školu pohađao je u Velikoj Kopanici. Po završetku osnovne škole, upisuje Gimnaziju Matije Mesića u Slavonskom Brodu, Prirodoslovno-matematički smjer. Nakon završetka gimnazije, 2011. godine upisuje preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku koji završava 2014. godine. Iste godine upisuje diplomski studij Procesno računarstvo na istom fakultetu. Trenutno je student 5. godine. Posjeduje vozačku dozvolu B kategorije i služi se engleskim jezikom u govoru i pismu.

Zvonimir Ivešić