УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (MASTER) РАД

Кандидат: **Урош Видојевић**
Број индекса: **E2101/2011**

Тема рада: **Једно решење подршке ДСМ-ЦЦ функционалности у оквиру ХббТВ стандарда**

Ментор рада: **Проф. др Никола Теслић**

Нови Сад, новембар, 2012.

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Редни број, **РБР**:	
Идентификациони број, **ИБР**:	
Тип документације, **ТД**:	Монографска документација
Тип записа, **ТЗ**:	Текстуални штампани материјал
Врста рада, **ВР**:	Завршни (Master) рад
Аутор, **АУ**:	Урош Видојевић
Ментор, **МН**:	Проф. др Никола Теслић
Наслов рада, **НР**:	Једно решење подршке ДСМ-ЦЦ функционалности у оквиру ХббТВ стандарда
Језик публикације, **ЈП**:	Српски / латиница
Језик извода, **ЈИ**:	Српски
Земља публиковања, **ЗП**:	Република Србија
Уже географско подручје, **УГП**:	Војводина
Година, **ГО**:	2012
Издавач, **ИЗ**:	Ауторски репринт
Место и адреса, **МА**:	Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)	7/40/0/6/10/0/0
Научна област, **НО**:	Електротехника и рачунарство
Научна дисциплина, **НД**:	Телевизијски системи
Предметна одредница/Кључне речи, **ПО**:	ДСМ-ЦЦ, ХббТВ, ВебКит
УДК	
Чува се, **ЧУ**:	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:	
Извод, **ИЗ**:	У овом раду је описана реализација интеграције програмског модула задуженог за приказ интернет садржаја и програмског модула за учитавање садржаја емитованог путем механизма цикличних објеката у дигиталној телевизији. Програмски модул задужен за приказ интернет садржаја који је коришћен у овом решењу је *WebKit*. Садржај емитован путем механизма цикличних објеката у овом случају јесу *Web* апликације које *WebKit* извршава.
Датум прихватања теме, **ДП**:	
Датум одбране, **ДО**:	
Чланови комисије, **КО**: Председник:	
Члан:	
Члан, ментор:	

Потпис ментора

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master Thesis	
Author, **AU**:		Uroš Vidojević	
Mentor, **MN**:		PhD Nikola Teslić	
Title, **TI**:		One solution of DSM-CC functionality support within HbbTV standard	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2012	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/40/0/6/10/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Television systems	
Subject/Key words, **S/KW**:		DSM-CC, HbbTV, WebKit	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This paper discribes implementation of integration of Web engine and software module for loading content which is transmitted through object carousel of television signal. Web engine which is used is WebKit. Content emitted by object carousel in this case are Web applications which are rendered by WebKit.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:		
	Member:		Menthor's sign
	Member, Mentor:		

Sadržaj

Spisak slika

Spisak tabela

Skraćenice

HbbTV - ***H**ybrid **B**roadcast **B**roadband **TV***, standard koji opisuje digitalnu hibridnu televiziju

MPEG-2 - ***M**oving **P**icture **E**xperts **G**roup **2***, standard za kodiranje pokretnih slika i pratećeg audia

DSM-CC - ***D**igital **S**torage **M**edia – **C**ommand and **C**ontrol*, deo MPEG-2 standarda koji standardizuje *„object carousel“*

IPTV - ***I**nternet **P**rotocol **T**elevision*, sistem putem koga se televizijski sadržaj šalje preko mreža koje su zasnovane na prosleđivanju paketa (kao što je Internet)

BBC - ***B**ritish **B**roadcasting **C**orporation*, Britanska nacionalna televizija

HDTV - ***H**igh **D**efinition **T**elevision*, televizija visoke rezolucije

DTV - ***D**igital **T**elevision*, digitalna televizija

DVB - ***D**igital **V**ideo **B**roadcasting*, skup standarda iz oblasti digitalne televizije

ETSI - ***E**uropean **T**elecommunication **S**tandard **I**nstitute*

MHP - ***M**ultimedia **H**ome **P**latform, Java* bazirane interaktivne aplikacije

CEA - ***C**onsumer **E**lectronics **A**ssociation*, organizacija za standarde i trgovinu iz oblasti potrošačke elektronike na tlu USA

W3C - ***W**orld **W**ide **W**eb **C**onsortium*, glavna svetska organizacija za standarde iz oblasiti World Wide Web-a

1. UVOD

U ovom radu prikazana je integracija dva programska modula: *WebKit-a i DSM-CC* modula. *WebKit* je *Web engine* - programski modul koji omogućava prikaz i izvršavanje *Web* baziranih aplikacija. *DSM-CC* modul je programski modul koji ima mogućnost da pročita podatke koji pristižu kroz tok cikličnih objekata televizijskog kanala. Mehanizam cikličnih objekata jeste algoritam koji se koristi za prenos celokupnog sadržaja bilo kakvih podataka. Čitav sadržaj se podeli na delove koji se zatim šalju počev od prvog do poslednjeg pa se ciklus slanja ponavlja. Ovaj algoritam se koristi prilikom slanja podataka koji su sastavni deo televizijskog kanala. Digitalna televizija omogućava da se pored audio i video signala šalju i druge komponente signala, tako da se mogu slati i podaci. Pritom signal kojim se prenosi televizijski kanal, a time i objekti koji se ciklično prenose, jeste *broadcast* signal. Preko toka cikličnih objekata mogu biti poslati bilo kakvi podaci, pa samim tim i *Web* aplikacije. *Web* aplikacija se izdeli na delove koji se zatim šalju preko toka cikličnih objekata. Tok cikličnih objekata se šalje zajedno sa drugim komponentama televizijskog kanala. Komponenta koja ima mogućnost da iz digitalnog televizijakog signala izdvoji tok cikličnih podataka, da od njegovih objekata ponovo sastavi aplikaciju koja je njime poslata i da aplikaciju stavi na raspolaganje korisniku se zove *DSM-CC* modul. Dakle, aplikacije koje *DSM-CC* modul učita *WebKit* treba da izvršava. Tačnije *DSM-CC* modul nije podržavao mogućnost da na zahev klijenta učita i smesti na lokalni sistem datoteka celu aplikaciju (emitovanu putem toka ciklicnih objekata). A nakon što je ova mogućnost dodata izbegli smo potrebu da uvodimo direktnu vezu između *WebKit-a* i programske podrske *TV* prijemnika. Prednosti ovakvog pristupa se ogledaju u tome što prilikom integracije *DSM-CC* modula i *WebKit-a* nema potrebe za izmenama *WebKit-a*, što je i bila tema ovog rada.

Rad se sastoji od sedam poglavlja.

U prvom poglavlju dat je kratak uvod u rad.

U drugom poglavlju dat je kratak istorijski pregled razvoja televizije koji je doveo do potrebe za ovim rešenjem. Takođe, date su teorijske osnove koje su neophodne za realizaciju i razumevanje ovog rešenja.

U trećem delu izložena je analiza problema koja je dovela do potrebe za ovim rešenjem.

U četvrtom delu je konceptuoalno opisan način na koji je problem rešen. Konceptualno su opisani novouvedeni programski podmoduli *DSMCC* modula kao i pojedinačne funkcije koje sačinjavaju novouvedene podmodule. Takođe, je opisan način na koji su integrisane nove i stare funkcionalnosti *DSMCC* modula u cilju rešenja problema koji je opisan u ovom radu.

U petom poglavlju je opisan način na koji je rešenje testirano i navedeni su rezultati testiranja.

U šestom poglavlju je kratak osvrt na rešenje i na mogućnosti rešenja.

U sedmom poglavlju je dat spisak literature koji je korišćen prilikom izrade ovog rešenja.

2. TEORIJSKE OSNOVE

U ovom poglavlju dat je najpre istorijski osvrt na razvoj televizije koji je doveo do potrebe da se implementira rešenje koje je opisano ovim radom. Potom je dat kratak pregled osnova prenosa televizijskog signala digitalnim putem , a nakon toga kratak pregled programskih modula koji su korišćeni u ovom radu.

2.1 Razvoj televizije i televizijskih standarda

Televizija je masovni medij. Informacije koje prenosi do velikog broja ljudi prenose se mehanizmom pokretnih slika. Kod ovog mehanizma se šalju slike sa određenom frekvencijom koja se naziva „*frame rate*“ i koji se razlikuje od zemlje do zemlje i od standarda do standarda, a kreće se u opsegu od 24*Hz* do 60*Hz*. Slike se šalju u tačno određenoj sekvenci gde se dve susedne slike razlikuju u položaju objekata na slici. Ove razlike u položaju objekata su tačno takve da kada se slike prezentuju posmatraču u sekvenci, posmatrač vidi kretanje objekata. Ovo je moguće, jer ljudsko oko nije u stanju da prepozna promene slika ako se one smenjuju sa frekvencijom koja je gore navedena ili većom, tako da se stvara „lažan“ utisak kretanja objekata. Ideja kod stvaranja televizije je bila da se sekvenca slika formirana na ovaj način šalje na daljinu putem telekomunikacionih veza, a zatim reprodukuje na uređajima nazvanim TV prijemnici. Prva slika je preneta na daljinu još krajem devetnaestog veka. Zatim je usledilo otkriće Nipkow-og diska koji je predstavljao prvi mehanički uređaj za skeniranje slika. Početkom dvadesetog veka je nastavljeno sa razvojem tehnologije neophodne za razvoj televizije. Dalji razvoj tehnologije omogućio je pojavu televizije krajem dvadesetih godina dvadesetog veka te je početkom ovog veka došlo do razvoja pojačavača amplitude na bazi vakumskih cevi. Ovaj pronalazak je omogućio da se prve slike prenesu žicom od skenera do prijemnika koji koristi ove pojačavače i da se na prijemniku reprodukuju. Prenošenje pokretnih slika još uvek nije bilo moguće, jer su selenijumske ćelije bile nedovoljno osetljive. Sredinom dvadesetih godina dvadesetog veka su prvi put prenesene pokretne slike na daljinu i stvorena je po prvi put iluzija „tečnih“ pokreta, jer se uspelo preneti 12,5 slika po sekundi, pošto je donja granica neophodna da stvori utisak „tečnih pokreta“ 12 slika po sekundi. U ovom periodu se tehnologija dalje razvijala, postizana je bolja rezolucija. Ovi napori su rezultovali time da je 1929. godine je prvi put emitovan *broadcast* televizijski signal u nekoliko država. *Broadcast* način emitovanja funkcioniše tako što emiter emituje signal u etar i pri tome ne emituje ka nekom specifičnom prijemniku već svako ko želi može da se priključi i

reprodukuje poslati signal na svom prijemniku. Istovremeno je došlo do razvoja televizijskih sistema kod kojih su i predajnik i prijemnik zasnovani na isključivo elektronskim komponenama. *BBC* je 1936emitovao i prvi signal generisan pomoću isključivo elektronskog skenera. Televizijska tehnologija je nastavila značajno da se razvija narednih decenija, rezolucija se značajno povećala, došlo je do pojave televizije u boji početkom pedesetih godina da bi postala standard sredinom sedamdesetih. Tokom osamdesetih je nastavljeno sa razvojem televizije. Nastavljeno je sa poboljšanjem rezolucije te se pojavio pojam *HDTV (**H**igh **D**efinition **T**elevision)*, koji označava televiziju sa rezolucijom od 1000 linija i više. Kao podrška postizanju ovog kvaliteta slike razvijeni su novi analogni uredjaji.

Uvođenje *HDTV* je bio prelaz ka digitalnoj televiziji. Iako su u početku uređaji bili uglavnom analogni, standardi u kvalitetu uslovili su postepeni prelaz na digitalnu televiziju. Razvijeni su digitalni uređaji i standardi u kvalitetu su se povećali. Razvoj televizije u ovom pravcu doveo je konačno do pojave standarda iz oblasti digitalne televizije (*DTV – **D**igital **T**elevision*). Postoji više skupova standarda digitalne televizije koji se primenjuju na pojedinim kontinentima i državama. Skup standarda koji je primenjivan u ovom radu je *DVB (**D**igital **V**ideo **B**roadcasting)*[1]. Ovaj skup standarda se primenjuje na tlu Evrope i još u nekim državama drugih kontinenata. Standardi koji pripadaju ovom skupu su standardizovani od strane ***E**uropean **T**elecommunication **S**tandard **I**nstitute (ETSI)*. Kako je u ovom periodu *MPEG* razvojna grupa privodila kraju svoj rad na standardu za kodiranje audia i videa, *DVB* razvojna grupa je odlučila da usvoji *MPEG* standard [2] za kodiranje audia i videa kao svoj standard. Kao rezultat ovih napora pri kraju 1992. godine je objavljen prvi skup standarda iz oblasti digitalnog prenosa satelitskog signala *DVB-S,* a ubrzo potom i iz oblasti kablovskog *DVB-C* i zemljanog *DVB-T* prenosa digitalnog signala.

Putem *DVB* signala je moguće pored audia i videa prenositi i grafiku, slike i bilo kakvu drugu vrstu podataka. Ovo je moguće zbog digitalne prirode ovog signala. Tako da je moguće prenositi i aplikacije koje se mogu izvršavati na prijemnicima. Ovaj deo *DVB* projekta se naziva *MHP(**M**ultimedia **H**ome **P**latform)*, i predstavlja interaktivni deo digitalne televizije. *MHP* aplikacije su zapravo *Java* aplikacije. Kasnije su se pojavili i drugi standardi koji opisuju interaktivne aplikacije koje se šalju putem televizijskog signala. Ovi standardi su *MHEG*(***M**ultimedia and **H**ypermedia **E**xperts **G**roup*) koji koristi svoj objektni jezik kao i *HbbTV*(***H**ybrid **B**roadcast **B**roadband **TV***)[3] koji podrazumeva aplikacije bazirane na *Web* jezicima kao što su *JavaScript* i *HTML*. Na ovom mestu treba napomenuti da se *DSM-CC* modul čije su modifikacije bile glavna tema ovog rada može koristiti za dobavljanje bilo kog tipa navedenih aplikacija. Na taj način je televizija postala interaktivan servis koji moze da pruži nove udobnosti korisniku digitalne televizije. Neke od ovih interaktivnih aplikacija su: informacioni servisi, igre, interaktivno glasanje, e-mail, SMS, kupovina...

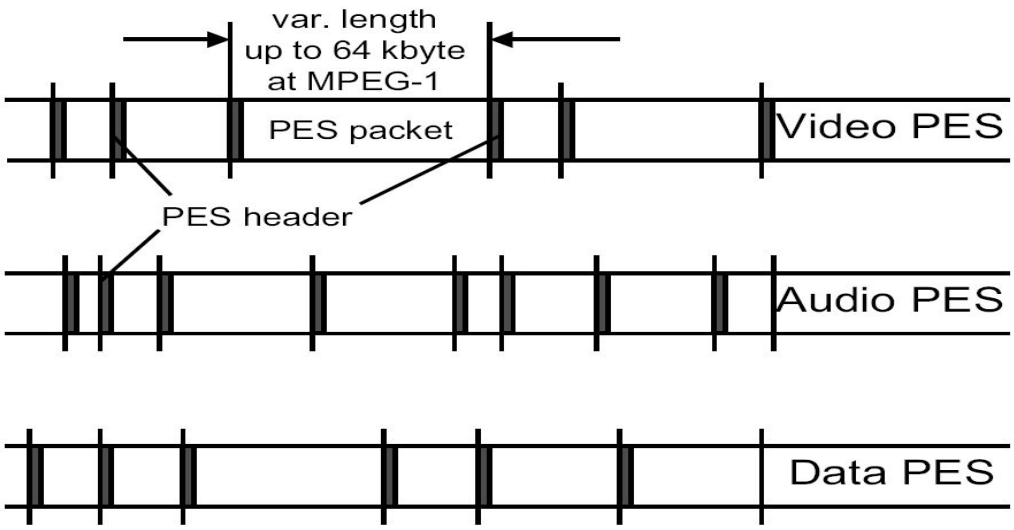
U drugoj polovini devedesetih i tokom dvehiljaditih sa razvojem interneta razvijali su se televizijski sistemi zasnovani na internet mreži. To su *broadband TV* i *IPTV*. Kao posledica ovakvog

razvoja televizije javila se ideja da se objavi standard koji bi objedinio *broadcast* i *broadband* način emitovanja interaktivnog televizijskog sadržaja tj. interaktivnih aplikacija i iskoristio sve prednosti *broacast-a i broadband-a*. Ova ideja je rezultovala 2009. godine u objavljivanju *HbbTV* specifikacije. Specifikacija je standardizovana 2010. godine od strane *ETSI*. Sam *HbbTV* standard je izgrađen na elementima već postojećih standarda i *Web* tehnologija kao što su *Open IPTV forum, CEA, DVB* i *W3C*.

2.2 Digitalni prenos televizijskog signala

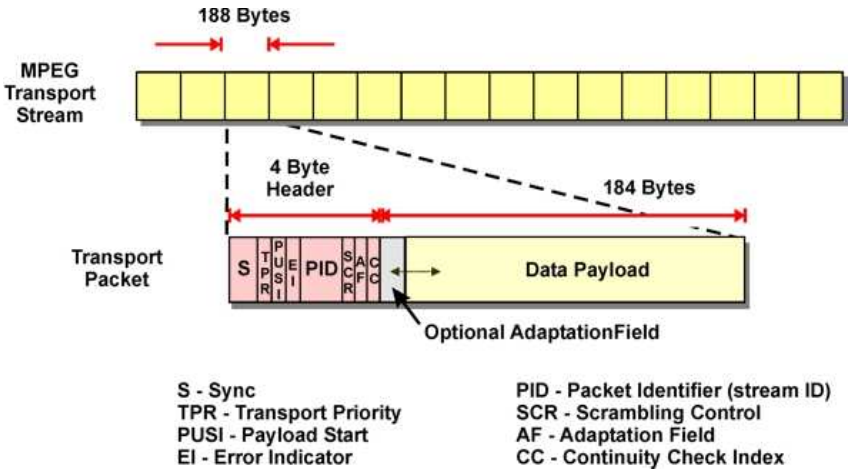
Digitalizacijom televizijskog signala je omogućeno da se signal kompresuje, te da se time značajno smanji protok podataka (eng. “bit rate”) koji je potreban za njegov prenos. Moguće je čuvanje *TV* podataka lokalno. Takođe je kvalitet višestruko bolji, jer nema interferencija sa drugim signalima. Moguće je prenositi sliku i zvuk visokog kvaliteta itd.

Kao što je već napomenuto grupa za razvoj digitalne televizije je usvojila *MPEG-2* standard za kompresiju digitalnog TV signala. Treba napomenuti da se koriste i neki drugi formati za svaku pojedinačnu komponentu (kao sto je *AC3* za audio signal), ali je *MPEG-2* dominantan. Današnji *TV* kanali mogu pored audia i videa da sadrže i razne interaktivne aplikacije. *MPEG-2* standard omogućava da se njime kompresuju bilo kakvi podaci kao što su audio, video i aplikacije, tako da se jedan kanal može sastojati od mnoštva komponenata kompresovanih u ovom formatu. Ovakve pojedine komponente se nazivaju elementarni tokovi podataka. Jedan kanal može sadržati npr. elementarni tok audia, videa, *HbbTV* aplikacije… Nakon što se pojedine komponente kompresuju one se odmah dele na pakete. Primer više elementarnih tokova podeljenih na pakete prikazan je na sledećoj slici:



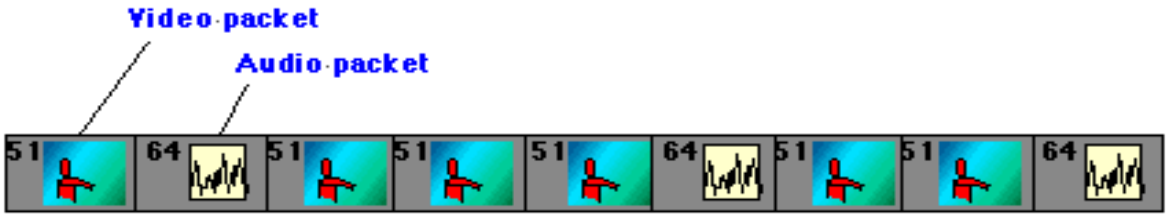
Slika 1: Primer više elementarnih tokova podeljenih na pakete

Ovakvi paketi se prosleđuju od predajnika do prijemnika putem paketski orjentisanih protokola. Paketi dobijeni deljenjem elementarnih tokova se multipleksiraju i šalju ka prijemnicima. Ovakav tok paketa se naziva *MPEG2* digitalni prenosni tok *(eng. transport stream)*. Na prijemnoj strani je ove pakete potrebno ponovo organizovati u logičke celine kakve su i poslate od strane predajnika te je stoga bilo potrebno uvesti i mehanizme kako za sinhronizaciju poslatih paketa tako i za sinhronizaciju audio i video signala. Ovi protokoli su standardizovani samim *MPEG-2* standardom, ali i posebnim televizijskim standardima (kao što je *DVB*). Na sledećoj slici je prikazan jedan *MPEG-2* digitalni prenosni tok gde je jedan paket izdvojen i prikazana njegova struktura.



Slika 2: Primer transport strima sa izdvojenim paketom

Paketi koji se šalju digitalnim prenosnim tokom nisu vremenski multipleksirani te stoga postoji oznaka uz svaki paket koja jednoznačno određuje kom elementarnom toku pripata određeni paket. Ta oznaka se naziva *Packet Identifier (PID).* Tako npr. u ovom slučaju video paketi imaju *PID* 51, a audio paketi 64. Digitalni prenosni tok koji sadrži samo audio i video pakete sa ovim *PID-ovima* je prikazan na sledećoj slici:



Slika 3: Primer transport strima

Kao sto je već rečeno elementarni tokovi mogu sadržati i podatke. Iz tog razloga uvedena su proširenja *MPEG-2* standarda koja su omogućila prenos: elektronskog programskog vodiča, uslovni

pristup sadržaju i razna druga proširenja. Mi ćemo se u nastavku rada koncentrisati na proširenje iz šestog dela *MPEG-2* standarda, a koje se odnosi na *DSM-CC*[4].

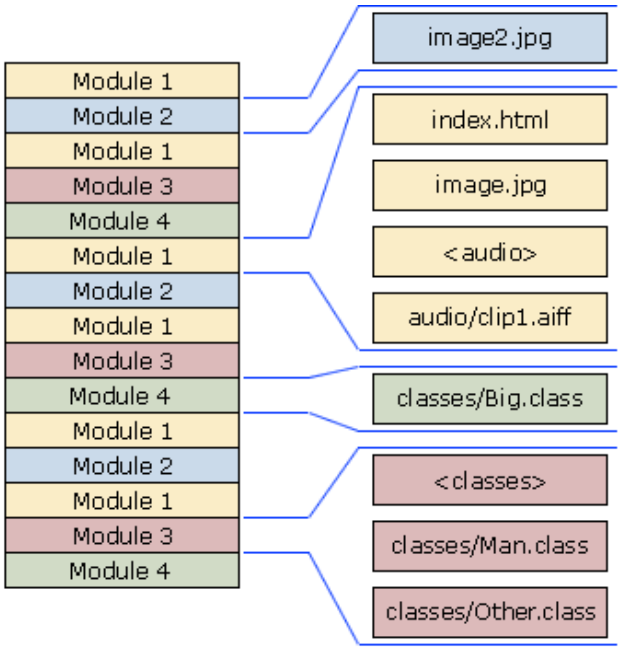
2.3 DSM-CC (**D**igital **S**torage **M**edia – **C**ommand and **C**ontrol) i WebKit

Kod *broadcast* emitovanja prijemnik nema mogućnost da uspostavi povratnu spregu sa predajnikom, odnosno nema mogućnost da zatraži sadržaj od predajnika. Prilikom ovakvog načina slanja sadržaja predajnik šalje sadržaj u *etar* i svako ko želi može se priključiti na komunikacijske veze i primati sadržaj, ali nema mogućnost da zatraži čitav sadržaj ili deo sadržaja od onoga ko šalje. Kao što je već rečeno *HbbTV* aplikacije su *Web* aplikacije. *Web* aplikacije se sastoje od mnoštva datoteka organizovanih u hijerarhiju u obliku sistema datoteka, a izvršava ih programski modul zadužen za prikazivanje *Web* sadržaja interpretirajući ove datoteke. Da bi mogao da izvrši kompletnu aplikaciju programski modul zadužen za prikazivanje *Web* sadržaja mora da ima na raspolaganju sve datoteke. Ukoliko je aplikacija smeštena na *Web* serveru programski modul zadužen za prikazivanje *Web* sadržaja može preko internet veza zatražiti bilo koju datoteku aplikacije, te u jednom trenutku može učitati samo one datoteke koje su mu potrebne. Kod *broadcast* načina emitovanja usvojeno je rešenje da predajnik šalje sve datoteke aplikacije u krug.

Kod *broadcast* načina emitovanja *DSM-CC* definiše mehanizam cikličkih objekata za prenos datoteka aplikacije. Algoritam koji se primenjuje je sledeći:

- Prenose se moduli najmanje veličine 64 KB.
- Nije dozvoljeno da se jedna datoteka šalje u delovima preko više modula.
- Ukoliko više datoteka ne prelazi po zbirnoj veličini 64 KB biće smeštene u modul veličine 64 KB.
- Ukoliko je jedna datoteka veća od 64 KB ona se šalje u okviru modula koji je veći od 64 KB.
- Nakon što se aplikacija podeli na module ovi moduli se šalju u krug. Zbog ovakvog načina slanja ovaj metod se zove – ciklični objekti (eng. *object carousel*). Pritom se neki moduli mogu slati češće od drugih. Ovim se omogućava da se neke datoteke kojima se češće pristupa šalju češće od drugih i time se smanji vreme potrebno za njihovo učitavanje, jer ne mora da se čeka pun krug da bi se ta datoteka mogla ponovo učitati. S druge strane se vreme učitavanja datoteka koje se ređe emituju povećava tako da treba pažljivo preraspodeliti datoteke po modulima i pažljivo odrediti učestanost slanja datoteka. Može se čak jedana ista datoteka emitovati više puta u različitim modulima.
- U modulima mehanizma cikličnih objekata se prenose i informacije o strukturi stabla direktorijuma aplikacije.

Na sledećoj slici je prikazan jedan tok ciklicnih objekata sa fajlovima različite veličine:



Slika 4: Primer object carousel-a

Zbir npr. veličina datoteka i ulaza direktorijuma index.html (1256B), image.jpg (4040B), <audio> i audio/clip1.aiff (26KB) ne prelazi 64 KB te se svi smeštaju u jedan modul. Pritom je <audio> ulaz audio direktorijuma. Dok datoteka image2.jpg (120KB) prelazi po veličini 64 KB pa je njen modul veći i iznosi 120KB.

Predajnik po navedenom algoritmu šalje sve datoteke aplikacije i informacije o strukturi sistema datoteka i na taj način ih stavlja na raspolaganje prijemnicima. Stoga je bilo neophodno razviti programski modul koji ima mogućnost da od modula poslatih mehanizmom cikličnih objekata sastavi aplikaciju i stavi je na raspolaganje drugim modulima koji imaju mogućnost da izvršavaju aplikaciju. Ovaj programski modul se zove *DSM-CC* modul. Ovaj modul "osluškuje" tok cikličnih objekata i kako moduli pristižu vrši njihovu analizu i formira sistem datoteka aplikacije. Informacije o hijerarhiji direktorijuma koristi da formira istu hijerarhiju na prijemnoj strani, a informacije o sadržaju datoteka koristi da formira same datoteke. Ovaj sistem datoteka smešta u operativnu memoriju. Pristup do ovog sistema datoteka je omogućen preko rukovaoca sistemom datoteka koji je deo samog *DSM-CC* modula.

S druge strane stoji programski modul zadužen za prikazivanje *Web* sadržaja. U ovom rešenju je korišćen *WebKit*, mada isto tako može biti korišćen bilo koji programski modul zadužen za prikazivanje *Web* sadržaja. *WebKit* ima mogućnost da reprodukuje *Web* aplikaciju ukoliko se njene datoteke nalaze na lokalnom sistemu datoteka. *DSM-CC* modul smešta aplikaciju u operativnu

memoriju tako da su datoteke u ovom rešenju kopirane iz operativne memorije na lokalni sistem datoteka i na taj način stavljene na raspolaganje *WebKit-u*.

3. ANALIZA PROBLEMA

Kao što je već napomenuto svaki elementarni tok koji se prenosi u okviru digitalnog prenosnog toka jednoznačno je određen identifikatorom koji se naziva *PID*. Tok cikličkih objekata se prenosi kao elementarni tok. Moduli toka cikličnih objekata se prenose u okviru paketa elementarnog toka te ovi paketi imaju imaju svoj *PID.* Dakle, tok cikličnih objekata se šalje kao elementarni tok sa sopstvenim *PID-om*. Putem toka cikličnih objekata se mogu poslati bilo kakvi podaci, a u našem slučaju šalje se *HbbTV* aplikacija. *DSM-CC* modul parsira ovakav elementarni tok tako što iz paketa „izvuče“ module toka cikličnih objekata i zatim sastavlja datoteke aplikacije od njih. Uz datoteke se šalju i informacije o strukturi direktorijuma te se ove informacije koriste da bi se napravilo stablo direktorijuma i datoteke smestile u svoj direktorijum. Sistem datoteka je smeštem u operativnoj memoriji. Pristup do ovog sistema datoteka je omogućen kroz rukovalac sistemom datoteka koji je deo samog *DSM-CC* modula. Sistem datoteka obezbeđuje pristup do datoteka kroz svoj *API* gde se datoteci pristupa tako što se navede njena apsolutna putanja. Koreni direktorijum ima naziv DSM:// iza koga sledi ostatak putanje (npr. DSM://dir1/dir2/img.jpg).

Svaki servis u okviru digitalnog prenosnog toka je po *DVB* standardu jednoznačno određen *DVB URL-om*. Ovaj *URL* ima format dvb://<onID>.<tsID>.<sID>[.<ctag>], gde pojedino polje ima sledeće značenje:

- onID – jedinstveni identifikator mreže ili *broadcaster-a* koji je emitovao sadrzaj
- tsID – identifikator digitalnog prenosnog toka u okviru mreže
- sID – identifikator servisa u okviru digitalnog prenosnog toka
- ctag – ovo polje je opciono i identifikuje elementarne tokove u okviru servisa. Može identifikovati najmanje jedan, a najviše tri elementarna toka.

Treba napomenuti da ovaj format nije kompletan i da se mogu pojaviti još neka opciona polja, ali se na tome nećemo zadržavati, jer to nije bilo od interesa prilikom izrade ovog rešenja.

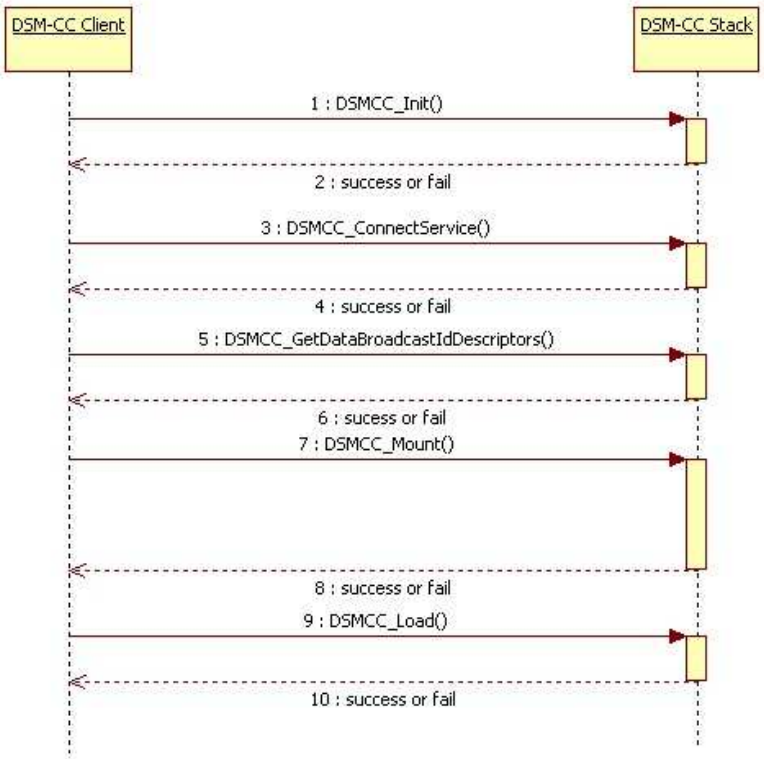
Korišćenjem *DSM-CC* sprege datotekama se pristupa na sledeći način:

- Najpre se mora inicijalizovati *DSM-CC* modul što se čini pozivom *DSMCC_Init()* funkcije.
- Nakon toga se vrši tzv. povezivanje na servis i time se izvrše potrebne pripreme da bi se omogućilo pribavljanje informacija o samom servisu (npr. informacija o postojećim HbbTV aplikacijama). Ova operacija se vrši pozivom funkcije *DSMCC_ConnectService().* Jedan od argumenata ove funkcije je *DVB URL* servisa.
- Potom se pozivom funkcije *DSMCC_GetDataBroadcastIdDescriptors()* dobiju informacije o *HbbTV* aplikacijama. Treba napomenuti da se pozivom ove funkcije dobijaju informacije o

svim servisima isporučenim putem toka cikličnih objekata, ali su nama od interesa jedino *HbbTV* aplikacije. Dostupne informacije uparuju *ctag* i *PID* elementarnog toka. Tako da se može učitati aplikacija identifikovana *DVB URL-om* (*ctag* je deo *URL-a*) čitanjem iz elementarnog toka identifikovanim odgovarajućim *PID-om*.

- Izvrši se smeštanje sistema datoteka aplikacije u operativnu memoriju. Pod ovime se podrazumeva proces formiranja sitema datoteka iz paketa digitalnog prenosnog toka o kome smo već govorili, i njegovog smeštanja u operativnu memoriju. Nakon sto se smeštanje završi datoteke se nalaze u operativnoj memoriji i može im se pristupiti. Funkcija kojom se vrši ovaj proces je *DSMCC_Mount().*
- Pošto je sistem datoteka aplikacije smešten u operativnu memoriju može se pristupiti datotekama aplikacije. Ovo se čini pozivom funkcije *DSMCC_Load().*

Dijagram toka poziva ovih funkcija je dat na sledećoj slici:



Slika 5: Dijagram toka poziva DSM-CC funkcija

Ovom sekvencom poziva koristeći *DSM-CC* modul možemo doći do svih datoteka koje se šalju tokom cikličnih objekata. Nakon poziva *DSMCC_Mount()* direktorijumi i datoteke se nalaze u operativnoj memoriji i mogu se učitati poizivom funkcije *DSMCC_Load().* Funkcija *DSMCC_Load()* kao jedan od argumenata prihvata punu putanju do datoteke koju učitava (npr.

DSM://dir1/dir2/img.jpg). Medjutim, struktura sistema datoteka se nije mogla dobaviti na osnovu predašnje sprege. S toga nije bila poznata hijerarhija direktorijuma kao ni datoteke koje se nalaze u njima, pa nisu bile poznate ni putanje do datoteka koje bi se učitale sa *DSMCC_Load()*. Informacije o sistemu datoteka postoje u samom *DSM-CC* modulu i one se inkrementalno nadograđuju kako direktorijumi i datoteke pristižu iz toka cikličnih objekata, ali one nisu bile dostupne korisniku *DSM-CC* modula.

U ovom rešenju je pronađen način da se ove informacije iskoriste i da se učitaju sve datoteke i direktorijumi aplikacije na osnovu ovih informacija. Način na koji su iskorišćene postojeće funkcionalnosti *DSM-CC* modula i dodate nove da bi se učitale sve datoteke *HbbTV* aplikacije na lokalni sitem datoteka i time stavljene na raspolaganje *WebKit-u* biće opisano u nastavku ovog rada.

4. KONCEPT REŠENJA

Osnovna ideja rešenja je bila da se dodaju nove i iskoriste već postojeće funkcionalnosti *DSM-CC* modula da bi se na lokalni sistem datoteka učitala aplikacija na osnovu *ctag* dela *DVB URL-a* (podsećanja radi *ctag* jednoznačno određuje elementarni tok). Osvrnimo se detaljnije na tok poziva funkcija iz sprege *DSM-CC* modula koju smo već naveli pa ćemo kroz taj pregled obrazložiti kako smo realizovali početnu ideju da se na osnovu *DVB URL-a* aplikacije ista i učita. Objasnićemo, takođe, i šta je bilo potrebno dodati u sam *DSM-CC* modul da bi se rešio zahtevani problem.

- *DSMCC_Init()* funkcija vrši inicijalizaciju samog *DSM-CC* modula, odnosno njegovih podmodula. Njenim pozivom se *DSM-CC* modul i startuje.
- *DSMCC_ConnectService()* funkcija kao jedan od argumenata prima *DVB URL* servisa čije tokove cikličnih objekata želimo da koristimo (treba napomenuti da uz jedan servis može ići više od jednog toka cikličnih objekata, koji nose različite podatke). Funkcija zatim prepoznaje elementarne tokove koji nose tokove cikličnih objekata servisa za koji je pozvana i beleži podatke o njima (npr. *ctag* i *PID* toka cikličnih objekata). Na taj način stavlja na raspolaganje ove informacije za kasnije korišćenje.

Predhodnim funkcijama se startovao *DSM-CC* modul i pripremljene informacije koje su potrebne korisniku. Osnovni zahtev koji je doveo do potrebe za ovim rešenjem, a to je da se učita aplikacija na osnovu *ctag-a* nije bio podržan spregom, pa se morala dodati nova funkcionalnost u samu spregu. Ovakva potreba je dovela i do realizacije novih podmodula u *DSM-CC* modul-u koji su nazvani *DSM-CC Loader i DSM-CC Glue*.

DSMCC_GetDataBroadcastIdDescriptors() se poziva da bi se podaci koje je *DSMCC_ConnectService()* pripremila učitali u niz koji ova funkcija dobija kao argument. Dakle, ova funkcija popunjava elemente niza odgovarajućim vrednostima. Drugi argument koji prima *DSMCC_GetDataBroadcastIdDescriptors()* je tip toka cikličnih objekata za koji želimo da se podaci učitaju. Ukoliko želimo podatke o tokovima cikličnih objekata *HbbTV* aplikacija, ovaj argument treba da ima vrednost 0x123. Polja elemenata niza koji je popunjen pozivom *DSMCC_GetDataBroadcastIdDescriptors()* funkcije, a koja su od značaja za ovo rešenje prikazana su sledećom tabelom:

Tip polja	Naziv polja	Značenje polja
unsigned int	pid	*PID* na kome se šalje *object carousel*.
int	componentTag	Polje je koje ima vrednost pomenutog *ctaga-a*.
unsigned char*	cidDesc	Polje koje sadrži vrednost *carousel id-ja*. Ova vrednost se koristi prilikom smeštanja aplikacije.

Tabela 1: Polja niza koji sadrži informacije o elementarnim strimovima HbbTV aplikacija

Kako se funkcija *DSMCC_GetDataBroadcastIdDescriptors()* poziva posredno preko poziva *DSMCC Glue* funkcije na ovom mestu će biti objašnjeni razlozi zbog kojih je uveden novi podmodul *DSMCC Glue* u *DSMCC* modul.

Kao što se iz samog naziva podmodula može zaključiti (*Glue* ili sr. „lepak“), podmodul služi za povezivanje korisnika i samog *DSM-CC* modula, i to baš onako kako korisniku odgovara.

Razlozi za implementiranje novog podmodula *DSM-CC* modula koji je nazvan *DSMCC glue* su bili sledeći:

1. Kao što je već napomenuto već postojeća *DSM-CC* sprega nije imala funkcionalnosti koje su bile potrebne za implementaciju ovog rešenja (npr. nije imala funkciju za učitavanje čitave *HbbTV* aplikacije na lokalni sistem datoteka) te je napravljena sprega na višem nivou koja će podržavati nove funkcionalnosti.
2. Nova sprega će funkcionalnosti koje podržava ostvarivati tako što će koristiti funkcije već postojeće *DSM-CC* sprege, ali i funkcije novoimplementiranog podmodula koji se zove *DSM-CC loader* (ovaj podmodul je deo ovog rešenja i on će biti obrazložen kasnije u radu). Dakle, za ostvarivanje ciljeva ovog rešenja bilo je potrebno koristiti kombinaciju već postojećih i novih funkcionalnosti *DSM-CC* modula.
3. Neke od novih funkcionalnosti mogu trajati vremenski dugo pa je bilo potrebno implementirati algoritam za njihovo asinhrono pozivanje, da se pozivalac ne bi blokirao na pozivu iz *DSM-CC glue* podmodula koji traje dugo. (npr. učitavanje *HbbTV* aplikacije koja sadrži veliki broj fajlova može trajati 5 min.).
4. Ovako implementiran podmodul se može vrlo lako proširiti novim funkcionalnostima ukoliko to bude potrebno.

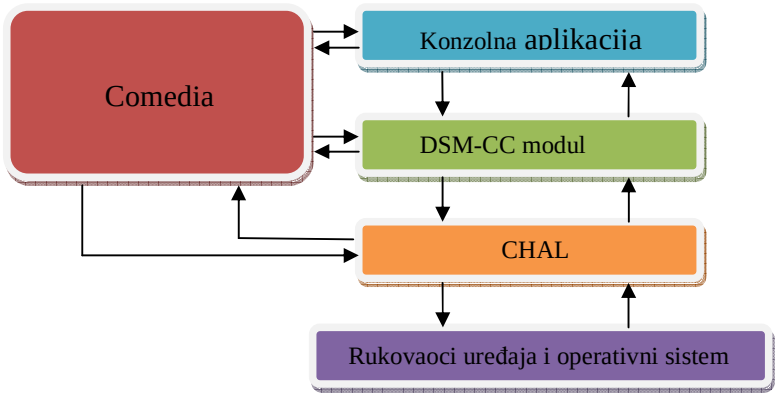
Jedna od funkcija *DSM-CC Glue* podmodula je *DSMCC_GLUE_PrintListOfHbbTVApplications()* koja koristeći *DSMCC_GetDataBroadcastIdDescriptors()* štampa listu *HbbTV* aplikacija koje su prisutne u trenutnom servisu. Odštampana lista aplikacija je prikazana na sledećoj slici:

```
Service contains 2 HbbTV applications, informations about them are given in the following list:
Application 0 :
        - pid = 0x87b = 2171
        - streamType = 0xb = 11
        - componentTag = 0x14 = 20
Application 1 :
        - pid = 0x87c = 2172
        - streamType = 0xb = 11
        - componentTag = 0x15 = 21

Do you want to load some of these applications? YES/NO...
yes
Please enter the PID of application you want to load in decimal format...
2171
```

Slika 6: Primer korišćenja DSM-CC Glue funkcionalnosti

Na ovom mestu treba napomenuti da je u ovom slučaju korisnik novih funkcionalnosti *DSM-CC* modula konzolna aplikacija. Ova aplikacija ima konzolnu spregu. U slučaju da je ova aplikacija aktivna ukucavanjem određenih komandi na konzoli mogu se pozvati funkcionalnosti iz *DSM-CC Glue* podmodula. Takodje, treba opisati strukturu čitavog sistema u koji je *DSM-CC* modul ukomponovan, jer *DSM-CC* modul kao komponenta ne može funkcionisati samostalno, već se oslanja i na druge programske module. Na slećoj slici je prikazana struktura programske podrške koja je korišćena u ovom rešenju:



Slika 7: Dijagram programskog sistema

Na slici su prikazani i neki programski moduli koji nisu do sada pominjani te ćemo stoga opisati šta pridstavljaju ti moduli, a nakon toga i na koji način komuniciraju svi moduli prikazani na slici.

Comedia je programska podrška TV prijemnika. To je centralni modul koji obavlja veliki broj funkcija. Funkcioniše kao mašina stanja koja održava stanje čitavog sistema. Startuje i inicijalizuje ostale module koji postoje u sistemu i reaguje na pobude poslate od drugih modula sistema. Ove pobude prebacuju *comedia-ju* u novo stanje što uglavnom rezultira pokretanjem novih akcija u sistemu. *Comedia* interaguje sa svim modulima u sistemu, a mi ćemo obrazložiti interakciju sa *DSM-*

CC modulom. *Comedia* poziva početne dve funkcije koje su potrebne da bi se *DSM-CC* modul startovao i povezao na servis, a to su već pomenute *DSMCC_Init()* i *DSMCC_ConnectService()*.

*CHAL(**C**omedia **H**ardware **A**bstraction **L**ayer)* je programski modul koji služi kao veza između drugih programskih modula sa rukovaocima uređajima i operativnim sistemom. Osnovna namena ovog modula se sastoji u tome da obezbedi nezavisnost drugih modula sistema od rukovaoca uređajima i operativnog sistema. Drugi moduli sistema pozivaju funkcije iz *CHAL-a*, a *CHAL* dalje poziva funkcije iz rukovaoca uređajima i operativnog sistema. Tako da u slučaju promene rukovaoca uređajima i operativnog sistema treba promeniti jedino *CHAL*.

Algoritam funkcionisanja sistema se sastoji u sledećem:

1. Konzolna aplikacija se prva pokreće i aktivira *Comedia-ju*. *Comedia* aktivira *DSM-CC* modul.
2. Konzolna aplikacija čeka da korisnik na konzoli ukuca neku od podržanih komandi. Postoji veliki broj komandi, ali nama su od značaja jedino skeniranje kanala i puštanje određenog kanala. Te treba zadati skeniranje kanala ukoliko nije već urađeno i potom pustiti željeni kanal.
3. Ove komande se dalje prosleđuju *Comedia-ji* koja poziva funkcije iz *CHAL-a*, koji dalje poziva funkcije rukovaoca uređajima i operativnog sistema. Poziva se veliki broj funkcija neophodnih za puštanje željenog kanala (između ostalih funkcije za puštanje audio i video signala). Funkcija koja je nama od značaja, a poziva se u ovom skupu, je već pomenuta DSMCC_ConnectService().
4. Nakon zadavanja ovih komandi tok kontrole je ponovo na konzolnoj aplikaciji koja čeka da joj se zadaju nove komande. U ovom trenutku korisnik može pozvati funkciju *DSMCC_GLUE_GetListOfHbbTVApplications()* koja popunjava već pomenuti niz sa informacijama o *HbbTV* aplikacijama. Ona to čini tako što poziva funkciju *DSMCC_GetDataBroadcastIdDescriptors()*. U ovom trenutku korisniku su na raspolaganju informacije o *HbbTV* aplikacijama. Pretražujući niz koji je popunila *DSMCC_GLUE_GetListOfHbbTVApplications()* funkcija korisnik može pronaći *PID-ove* koji odgovaraju *ctag-u* (podsećanja radi jedan *ctag* može identifikovati najviše tri elementarna toka, a svaki tok je jednoznačno određen *PID-om* te se stoga jednim *ctag-om* određuje najmanje jedan, a najviše tri *PID-a*).
5. Sledi učitavanje *HbbTV* aplikacija koje su identifikovane *ctag-om*. Za ovo učitavanje postoji odgovarajuća komanda u konzolnoj aplikaciji. Konzolna aplikacija pretražuje niz koji je dobijen u prethodnom koraku na vrednost željenog *ctag-a*. Ulazi ovoga niza sadrže uparene vrednosti *ctag-a* i *PID*, pa za svaki ulaz koji ima traženu vrednost *ctag-a* poziva funkciju iz *DSM-CC Glue-a DSMCC_GLUE_LoadApplicationByPID()* koja učitava željenu aplikaciju na lokalni sistem datoteka. Ova funkcija koristi funkcionalnosti novo implementiranog podmodula

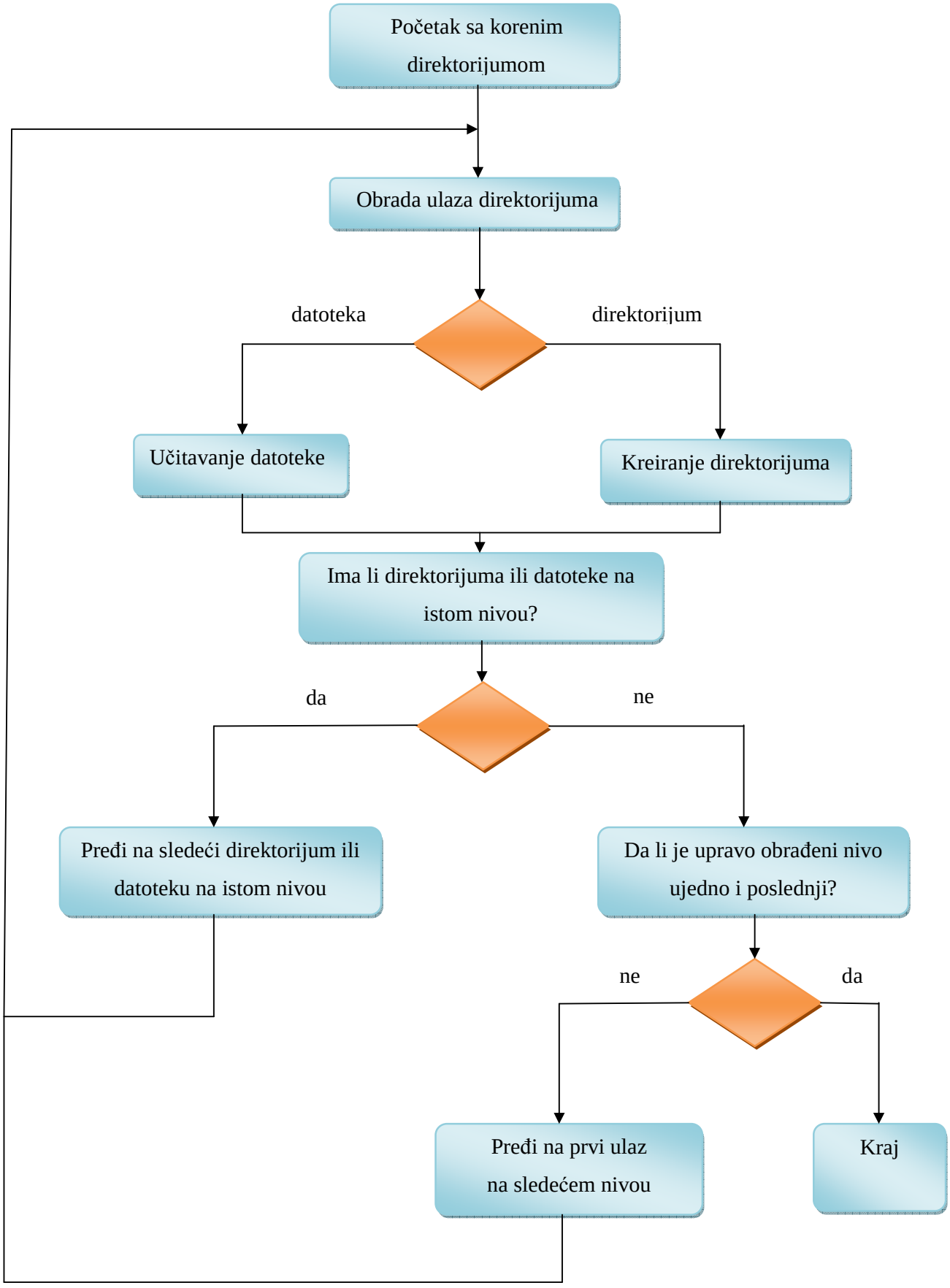
koji je nazvan *DSM-CC Loader*. Potreba koja je dovela do implementiranja ovog podmodula kao i način na koji je implementiran biće opisani u nastavku ovog rada.

4.1 DSM-CC Loader

Osnovna ideja kod učitavanja aplikacije je bila da se iskoristi funkcija *DSMCC_Load()*, tako što bi bila pozvana za svaku datoteku aplikacije. Ova funkcija kao jedan argument prima punu putanju do datoteke na sistemu datoteka organizovanom u operativnoj memoriji (npr. DSM://dir1/dir2/img.jpg) , a kao drugi argument funkciju koja će biti pozvana pošto se učitavanje datoteke završi. Medjutim, da bi smo na ovaj način dosli do datoteka moramo najpre aplikaciju smestiti u operativnu memoriju, a nakon toga i učitati njene datoteke. Smeštanje sistema datoteka aplikacije u operativnu memoriju vrši funkcija *DSMCC_Mount()*. Ova funkcija kao argument prima *PID* aplikacije tj. *PID* pod kojim se šalje tok cikličnih objekata. Od modula toka cikličnih objekata formira datoteke aplikacije. Informacije o strukturi direktorijuma koristi da formira stablo direktorijuma. Pošto ova funkcija uspešno završi smeštanje aplikacije u operativnu memoriju treba pozvati funkciju *DSMCC_Load()* i učitati njene datoteke. Medjutim, struktura sistema datoteka se nije mogla dobaviti na osnovu pređašnjih funkcionalnosti te se nije moglo doći ni do putanja do datoteka za koje bi učitavanje bilo pozvano. Informacije o sistemu datoteka postoje u samom DSM-CC modulu i zabeležene su u njegovim strukturama te je trebalo pronaći način da se ove informacije i iskoriste. Dakle, ideja se sastojala u tome da se napravi novi podmodul kome se može zadati da učita aplikaciju sa određenim *PID-om* na određenu putanju na lokalnom sistemu datoteka. Novi podmodul će to raditi koristeći već postojece funkcionalnosti DSM-CC modula i dodati nove funkcionalnosti da bi se zadatak ostvario. Novi podmodul je nazvan *DSMCC Loader*.

Centralna funkcija novouvedenog podmodula je *DSMCC_loadDSMCCFiles()*. Jedini argument ove funkcije je putanja na lokalnom sistemu datoteka na koju želimo da učitamo aplikaciju. Osnovna ideja kod implementiranja ove funkcije je sledeća:

Koreni direktorijum na lokalnom sistemu datoteka na koji će biti učitan koreni direktorijum aplikacije je direktorijum koji je prosleđen kao argument. Datoteke i direktorijumi koji se nalaze u nekom direktorijumu jesu ulazi direktorijuma. Datoteka koja se nalazi u korenom direktorijumu biće odmah i smeštena u njega. Za direktorijum koji se nalazi u korenom direktorijumu biće napravljen direktorijum sa istim imenom i u korenom direktorijumu na lokalnom sistemu datoteka. Ideja je da se svaki ulaz čim se na njega naiđe odmah i obradi, bilo da on predstavlja direktorijum bilo datoteku. Potom se prelazi na sledeći nivo stabla i direktorijumi i datoteke se obrađuju na isti način. Na taj način se vrši obrada n-arnog stabla direktorijuma po širini. Dijagram toka ovog algoritma je prikazan na sledećoj slici:



Slika 8: Algoritam obrade ulaza direktorijuma

4.1.1. Realizacija DSMCC_loadDSMCCFiles()

Funkcija *DSMCC_loadDSMCCFiles()* realizuje gore opisani algoritam. Informacije o stablu direktorijuma se nalaze zabeležene u strukturama koje se nalaze u samom modulu, i one se inkrementalno nadograđuju kako pristižu moduli iz toka cikličnih objekata. Informacije pristižu od korenog direktorijuma prema listovima tj. pristižu po širini, pa se predhodni algoritam prirodno uklopio u način formiranja stabla u operativnoj memoriji.

Stablo direktorijuma je u samom *DSM-CC* modulu zabeleženo u okviru liste. Lista sadrži čvorove stabla u redosledu obilaska stabla po širini. Pritom se lista ulančava počev od glave te je stoga koreni direktorijum na repu liste, potom slede svi njegovi sinovi u redosledu s leva na desno, zatim svi njihovi sinovi u istom redosledu itd. sve do listova. Pritom članovi ove liste ne sadrže informacije o direktorijumima već samo rukovaoce koji su zapravo celi neoznačeni brojevi na osnovu kojih se može doći do informacija o direktorijumima. Do informacija se dolazi pozivajući odgorarajuće operacije nad ovim rukovaocima. Funkcija vrši obradu stabla direktorijuma po sledećim koracima:

1. Vrši se pozicioniranje na kraj liste direktorijuma. Ovime smo se pozicionirali na element liste koji reprezentuje koreni direktorijum.
2. Inicijalizuje se semafor koji služi za sinhronizaciju operacija nad ulazima. U suprotnom bi se moglo dogoditi da se počne sa obradom ulaza direktorijuma pre nego što se završila obrada njegovog pretnodnika pa njegov prethodnih ne bi bio kompletno obrađen.
3. Ulazi se u petlju u kojoj se kreće kroz listu direktorijuma i vrši obrada njihovih ulaza. U ovoj petlji se vrše sledeći koraci:
    a) Na početku ove petlje se koristi pomenuti rukovalac da se dođe do strukture koja opisuje direktorijum. Ova struktura je tipa _tDIRi_Context i sadrži veliki broj polja, a ovde će biti navedena samo ona koja su bila od značaja pri izradi ovog rešenja.

Polja strukture _tDIRi_Context		
Tip polja	Naziv polja	Značenje polja
Nabrajanje _tDIRi_DirState	dirState	Označava u kom se stanju nalazi direktorijum u pogledu njegovih ulaza. (npr. da li su svi formirani itd.)
int	numEntry	Broj ulaza direktorijuma.
char*	mountPath	Putanja direktorijuma u sistemu datoteka u operativnoj memoriji.
_tDIRi_Entry*	ptrEntryList	Niz koji čiji elementi sadrže informacije o ulazima u direktorijumu.

Tabela 2: Polja strukture _tDIRi_Context

Unutrašnji član ove strukture je struktura _tDIRi_Entry koja opisuje ulaze direktorijuma. Polja ove strukture koja su bila od značaja prilikom izrade ovog rada su prikazana sledećom tabelom.

Polja strukture _tDIRi_Entry		
Tip polja	Naziv polja	Značenje polja
char*	ptrName	Predstavlja ime ulaza tj. datoteke ili direktorijuma
unsigned int	uKind	Vrsta ulaza, da li je ulaz predstavlja datoteku ili direktorijum
_tDIRi_EntryState	entryState	U kom stanju se nalazi ulaz. (npr. ulaz je učitan)

Tabela 3: Polja strukture _tDIRi_Entry

b) Iz putanje direktorijuma na sistemu datoteka se ekstrahuje relativna putanja koja se nalazi iza DSM:/ (npr. relativna putanja od DSM://dir1/dir2 je /dir1/dir2). Ovako ekstrahovana putanja se smešta u posebnu promenljivu i koristi se da bi se napravila putanja za smeštanje direktorijuma na lokalnom sistemu datoteka. Ova putanja se formira tako što se ekstrahovana relativna putanja nadoveže na putanju koju je korisnik zadao kao argument funkcije. (npr. ako korisnik želi da koreni direktorijum aplikacije bude smešten na putanji /home/korisnik/Desktop/HbbTVAplikacija, direktorijum koji se trenutno obrađuje biće smešten na putanji: /home/korisnik/Desktop/HbbTVAplikacija/dir1/dir2). Ova putanja će biti korišćena kasnije prilikom obrade ulaza.

c) Sačeka se da ulazi tekućeg direktorijuma budu formirani. Ovo se čini tako što se ispita polje *dirState* primerka strukture *_tDIRi_DirState* koje ukazuje u kom se stanju nalaze ulazi direktorijuma. Ukoliko ima vrednost *eDIRi_DIR_STATE_TREE_COMPLETE* znači da su ulazi direktorijuma formirani i može se pristupiti njihovoj obradi, a ukoliko ne, čeka se 0.1 sekunda i provera se vrši ponovo. Ovde treba napomenuti da funkcija *DSMCC_Mount()* ne smešta celokupan sistem datoteka u operativnu memoriju već samo otpočinje proces smeštanja dok sam proces traje i nakon završetka ove funkcije i to tako što se stablo inkrementalno nadograđuje kako pristižu objekti iz toka cikličnih objekata.

d) Ulazi se u petlju u kojoj se obrađuju ulazi direktorijuma. Za broj izvršavanja petlje koristi se polje *numEntry* primerka strukture *_tDIRi_Context* koje sadrži informaciju o broju ulaza u direktorijumu koji se trenutno obrađuje. Ulazi se nalaze u nizu koji je član strukture *_tDIRi_Context* i predstavljen je poljem *ptrEntryList* ove strukture.

Elementi ovog niza su tipa *_tDIRi_Entry* i predstavljaju ulaze direktorijuma. Za svaki od ovih ulaza primenjuju se sledeći koraci:

i. Najpre se u posebnim promenljivama sačuvaju putanja do direktorijuma u sistemu datoteka koji se nalazi u operativnoj memoriji u kome nalazi tekući ulaz kao i ime tekućeg ulaza. (npr. putanja do direktorijuma može biti DSM://dir1/dir2 dok ime ulaza može biti img.jpg). Putanja do direktorijuma se uzima iz polja *mauntPath* primerka strukture *_tDIRi_Context* koji predstavlja trenutni direktorijum, dok se ime ulaza uzima iz polja *ptrName* primerka strukture *_tDIRi_Entry* koji predstavlja trenutni ulaz.

ii. Zatim se pristupa daljoj obradi ulaza u zavisnosti šta ulaz predstavlja. Informacija o tome šta ulaz predstavlja nalazi se u polju *uKind* primerka strukture *_tDIRi_Entry* koji predstavlja trenuni ulaz. Ukoliko polje ima vrednost *kTABLE_BIOP_FIL* ulaz predstavlja datoteku a ukoliko ima vrednost *kTABLE_BIOP_DIR* ulaz predstavlja direktorijum.

U slučaju datoteke prvo se proverava da li je datoteka učitana. Ovo se čini proveravanjem polja *entryState* primerka strukture *_tDIRi_Entry* koji predstavlja trenutni ulaz.Ukoliko polje ima vrednost *eDIRi_ENT_STATE_LOADED* datoteka je učitana u sistemu datoteka organizovanom u operativnoj memoriji, a ukoliko nema ovu vrednost sačeka se 0.1 sekunda i provera se vrši ponovo. U trenutku kad polje dobije vrednost *eDIRi_ENT_STATE_LOADED* pristupa se njegovom učitavanju na lokalni sistem datoteka. Učitavanje se vrši pozivanjem funkcije *DSMCC_loadFile()* koja je sastavni deo podmodula *DSMCC Loader*. Kao argumente ova funkcija prihvata putanju do direktorijuma na lokalnom sistemu datoteka na koju datoteka treba da bude učitana, putanju do direktorijuma na sistemu datoteka u operativnoj memoriji u kome se datoteka nalazi kao i ime same datoteke (npr. u direktorijum /home/korisnik/Desktop/HbbTVAplikacija/Dir1/Dir2 treba učitati datoteku iz direktorijuma DSM://Dir1/Dir2 sa imenom img.jpg). Ove putanje su formirane u prethodnom koraku. Funkcija *DSMCC_loadFile()* će biti objašnjena kasnije.

U slučaju direktorijuma poziva se funkcija *DSMCC_createDir()* koja kreira direktorijum na lokalnom sistemu datoteka. Kao argumente ova funkcija prihvata putanju do direktorijuma na lokalnom sistem datoteka u kome se direktorijum kreira kao i ime ulaza koje je u ovom slučaju ime direktorijuma. (npr. u direktorijumu /home/korisnik/Desktop/HbbTVAplikacija/Dir1/Dir2 se

kreira direktorijum Dir3. Ovi argumenti su formirani u pretnodnom koraku. Funkcija *DSMCC_createDir()* će biti objašnjena u nastavku rada.

4. Prelazi se na sledeći direktorijum u listi direktorijuma. Kreće se u listi od repa ka glavi.
5. Pošto su obrađeni svi direktorijumi u listi izlazi se iz spoljašnje petlje. Nakon toga uradi se operacija *wait* na semaforu koji služi za sinhronizaciju obrade ulaza da bi se sačekalo da i poslednji ulaz bude obrađen, a zatim se funkcija završava. Vraća vrednost 1 ili 0 u zavisnosti od toga da li se dogodila neka greška prilikom učitavanja ili ne.

Nakon što funkcija uspešno završi svoj rad sistem datoteka koji se nalazi u operativnoj memori i koji je pod kontrolom *DSMCC* modula je kopiran u direktorijum koji je zadat kao argument ove funkcije te ga *WebKit* može koristiti. Funkcija *DSMCC_loadDSMCCFiles()* kao što je već napomenuto poziva funkcije *DSMCC_loadFile()* i *DSMCC_createDir()* koje će biti objašnjene u nastavku rada.

4.1.2. Realizacija DSMCC_loadFile()

Osnovna ideja kod realizacije ove funkcije je bila da iskoristi već postojeće funkcionalnosti *DSMCC* modula da bi datoteku koja se nalazi u sistemu datoteka u operativnoj memoriji kopirala u lokalni sistem datoteka. Pritom funkcija kao argumente prima putanju do direktorijuma na lokalnom sistemu datoteka u koji se datoteka želi kopirati, putanju na sistemu datoteka u operativnoj memoriji gde se datoteka nalazi i ime same datoteke. (npr. u direktorijum /home/korisnik/Desktop/HbbTVAplikacija/Dir1/Dir2 kopiraj datoteku img.jpg iz direktorijuma DSM://Dir1/Dir2).

Funkcija koristi funkciju *DSMCC_Load()* iz *DSMCC* modula. Ova funkcija je asinhrona. Ona se pozove da učita određenu datoteku kaja je zadata putanjom u sistemu datoteka u operativnoj memoriji. Ona datoteku kopira iz sistema datoteka u operativnoj memoriji u određenu strukturu u operativnoj memoriji. Nakon što završi ovo kopiranje ona poziva povratnu funkciju koja joj je prosleđena kao argument. U ovoj povratnoj funkciji se koristi memorijska struktura u koju je *DSMCC_Load()* funkcija kopirala datoteku da bi se ista konačno kopirala na lokalni sistem datoteka.

Funkcija je realizovana po sledećem algoritmu:

1. Najpre se pozove operacija *wait* na semaforu koji se koristi za sinhronizaciju operacija nad ulazima direktorijuma, jer se može desiti da se prethodno zadata operacija nad ulazom direktorijuma nije završila pa bi novozadata prepisala nove podatke preko već postojećih u već pomenutoj memorijskoj strukturi u koju *DSMCC_Load()* kopira datoteku.

2. Nakon toga se koriste argumenti funkcije da bi se formirale putanje izvorišta i odredišta kopiranja datoteke. (npr. izvorište može biti DSM://Dir1/Dir2/img.jpg dok odredište može biti /home/korisnik/Desktop/HbbTVAplikacija/Dir1/Dir2/img.jpg)
3. Ime odredišta se sačuva u globalnoj promenljivoj da bi ga kashije povratna funkcija koristila, dok se ime izvorišta iskoristi da se sa njime pozove *DSMCC_Load()* funkcija. Ovoj funkciji se prosleđuje povratna funkcija koju treba da pozove kad završi učitavanje.
4. Ukoliko je poziv funkcije *DSMCC_Load()* završen uspešno vraća se status o uspehu u suprotnom vraća se status o grešci. Ovi statusi pripadaju nabrajanju koje se nalazi u okviru samog *DSMCC* modula i koji nose informaciju o statusu završene operacije.

4.1.3. Realizacija DSMCC_createDir()

Ova funkcija ima zadatak da izvrši obradu ulaza direktorijuma ukoliko je ulaz direktorijum (u ovom slučaju se direktorijum nalazi u okviru direktorijuma). Obrada se vrši tako što se kreira direktorijum na lokalnom sistemu datoteka koji ima isto ime kao ulaz koji se obrađuje.

Kao argumente funkcija prima putanju do direktorijuma na lokalnom sistemu datoteka u koji se želi kreirati direktorijum, dok je drugi argument ime direktorijuma koji se kreira.

Funkcija je realizovana po sledećem algoritmu:

1. Najpre se uradi operacija *wait* na semaforu koji služi za sinhronizaciju operacija nad ulazima direktorijuma. Ova operacija će blokirati sve druge operacije nad ulazima direktorijuma dok se kreiranje ne završi. Na taj način se onemogućuje da se pozove operacija učitavanja datoteke ili kreiranja drugog direktorijuma u direktorijumu koji se upravo kreira.
2. Nakon toga se poziva operacija kreiranja direktorijuma na lokalnom sistem datoteka.
3. Poziva se operacija *signal* na semaforu koji služi za sinhronizaciju operacija nad ulazima direktorijuma. Ova operacija omogućava nastavak obrade ulaza, a moguća je, jer se obrada trenutnog ulaza (u ovom slucaju direktorijuma) u potpunosti završila.
4. Funkcija se završava i vraća status izvršene operacije.

4.1.4. Realizacija _DSMCC_NfyLoadCB()

Ova funkcija se prosleđuje funkciji *DSMCC_Load()* kao argument i biće pozvana kad se završi smeštanje datoteke iz sistema datoteka koji se nalazi u operativnoj memoriji u strukturu koja se nalazi u operativnoj memoriji, a koje je inicirano od strane *DSMCC_Load()* funkcije. *DSMCC_Load()* funkcija ne vrši smeštanje datoteke već samo zada tu operaciju drugim delovima *DSMCC* modula i odmah se završi. Kada se smeštanje završi poziva se *_DSMCC_NfyLoadCB().*

Funkcija je realizovana po sledećem algoritmu:

1. Izvrši se kopiranje sadržaja datoteke iz strukture koja se nalazi u operativnoj memoriji u strukturu koja je lokalna i nalazi se u funkciji. Ovo kopiranje se vrši funkcijom *DSMCC_GetData().*
2. Putanja do datoteke koja se kreira je formirana u okviru *DSMCC_loadFile()* funkcije i zabeležena u globalnoj promenljivoj i sada koristi da se otvori datoteka na lokalnom sistemu datoteka. (npr. /home/korisnik/Desktop/HbbTVAplikacija/Dir1/Dir2/img.jpg).
3. Nakon što je datoteka otvorena vrši se kopiranje datotete, bajt po bajt, iz strukture u upravo otvorenu datoteku na lokalnom sistemu datoteka.
4. Poziva se funkcija *DSMCC_Unload()* koja vrši oslobađanje memorijskih struktura za ponovno učitavanje datoteke od strane *DSMCC_Load()* funkcije.
5. Poziva se operacija *signal* na semaforu koji služi za sinhronizaciju operacija za obradu ulaza direktorijuma. Ova operacija omogućuje da se nastavi obrada ulaza, jer je prethodnim kreiranjem datoteke operacija sa datotekom završena i može se preći na obradu drugog ulaza.

Pošto se ova funkcija završi dateka je kreirana na lokalnom sistemu datoteka.

4.2. DSMCC Glue

U uvodnom delu ovog poglavlja obrazloženi su razlozi za uvođenje ovog podmodula i navedeni primeri korišćenja nekih od njegovih funkcija u slučaju da je korisnik *DSMCC* modula konzolna aplikacija. U ovom poglavlju će biti dat detaljan pregled realizacije funkcija ovog modula.

Ovaj podmodul se sastoji iz dva dela. U prvom delu su realizovane funkcije koje obezbeđuju novu spregu ka *DSMCC* modulu. Drugi deo služi da bi se omogućilo asinhrono pozivanje funkcija iz nove sprege, jer izvršavanje nekih funkcija može trajati vremenski dugo (npr. 5 min.). Na taj način se izbegava da korisnik ostane vremenski dugo blokiran na nekoj od funkcija iz sprege.

Najpre će biti opisan deo koji služi za asinhrono pozivanje funkcija nove sprege, a potom i funkcije nove sprege. Ovaj redosled izlaganja je odabran, jer funkcije nove sprege koriste funkcionalnosti dela za asinhrono pozivanje.

4.2.1. DSMCC Glue deo za asinhrono pozivanje funkcija

Osnovna zamisao kod realizacije ovog dela *DSMCC Glue-a* je bila da postoji jedan *FIFO(**F**irst **I**n **F**irst **O**ut)* red u koji bi korisnik stavljao poruke, dok bi ovaj red „osluškivala“ jedna programska nit koja bi se neprestano izvršavala. Ove poruke bi nosile informaciju o tome koja operacija treba da se izvrši. Programska nit bi po stavljanju poruke u red istu i uzela. Poruka bi se onda protumačila i u zavisnosti od njenog tipa pozvala odgovarajuća operacija. Korisnik bi argumente neophodne za

pozivanje operacije takođe smestio u poruku. Dakle, korisnik bi napravio poruku sa operacijom koju želi da pozove i smestio je u red. Operacija bi kasnije bila pozvana od strane programske niti.

Svi članovi ovog dela *DSMCC Glue-a* imaju u svom nazivu reč *task* (sr. „zatadak“) što je imalo za cilj da asocira na funkciju ovog dela.

4.2.1.1. Format poruke

Poruka koja se stavlja u pomenuti red se sastoji iz dva dela. Prvi deo je tip poruke koji služi da bi se odredilo koja operacija treba da se pozove. Drugi deo služi da bi se preneli argumenti operacije. Sama poruka je struktura koja se zove *tDSMCC_GLUE_Task_Message*. Struktura je predstavljena sledećom tabelom:

Članovi strukture *tDSMCC_GLUE_Task_Message*		
Tip člana	Ime člana	Značenje člana
Nabrajanje *tDSMCC_GLUE_Task_MessageType*	*msgType*	Tip poruke
Unija	*msgData*	Argumenti operacije

Tabela 4: Članovi strukture *tDSMCC_GLUE_Task_Message*

Tip poruke je predstavljen nabrajanjem *tDSMCC_GLUE_Task_MessageType*. Ovo nabrajanje ima dva člana koji predstavljaju dve operacije koje se mogu pozvati. Prva operacija je učitavanje čitave aplikacije na osnovu *PID-a*. Ova operacija može trajati vremenski dugo (npr. 5 min.) te je nju bilo neophodno pozivati asinhrono. Druga operacija je prekidanje izvršavanja programske niti koja izvršava operacije. Ovo nabrajanje je predstavljeno sledećom tabelom:

Članovi nabrajanja *tDSMCC_GLUE_Task_MessageType*	
Ime člana	Značenje člana
eDSMCC_GLUE_TYPE_LOAD_APPLICATION_BY_PID	Učitavanje čitave HbbTV aplikacije na osnovu *PID-a*
eDSMCC_GLUE_TYPE_TERMINATE	Zaustavljanje programske niti koja izvršava operacije

Tabela 5: Članovi nabrajanja *tDSMCC_GLUE_Task_MessageType*

Argumenti operacije su predstavljeni unijom *msgData* čiji svaki član nosi argumente za pojedinu operaciju. Tako npr. za operaciju *eDSMCC_GLUE_TYPE_LOAD_APPLICATION_BY_PID* argumente nosi član *applicationRequest*. Tako da ukoliko korisnik zeli da pozove operaciju koja je predstavljena tipom poruke *eDSMCC_GLUE_TYPE_LOAD_APPLICATION_BY_PID* formiraće

poruku ovog tipa i popuniti član *applicationRequest* unije *msgData.* Ostali članovi unije u slučaju operacije predstavljene sa *eDSMCC_GLUE_TYPE_LOAD_APPLICATION_BY_PID* nisu bitni i ne moraju biti popunjeni, jer neće biti ni korišćeni. Treba napomenuti da je *applicationRequest* i jedini član unije *msgData,* jer je učitavanje *HbbTV* aplikacije na osnovu *PID-a* jedina operacija koja se poziva asinhrono, a da ima argumente. Naravno ukoliko se bude ukazala potreba za dodavanjem novih asinhronih operacija unija *msgData* se lako može proširiti novim članovima, isto kao što se nabrajanje *tDSMCC_GLUE_Task_MessageType* može proširiti novim tipovima operacija. Na sledećoj slici su prikazana polja člana *applicationRequest* unije *msgData*.

Polja *applicationRequest-a,* člana unije *msgData*		
Tip polja	Ime polja	Značenje polja
int	*pid*	*PID* aplikacije koju želimo da učitamo
int	*cid*	*CID* aplikacije koju želimo da učitamo
*char**	*loadPath*	Putanja na lokalnom sistemu datoteka na koju želimo da učitamo aplikaciju
tDSMCC_GLUE_NfyApplicationLoadCB	*cbNfyCB*	Povratna funkcija koja se poziva pošto se učitavanje završi

Tabela 6: Polja *applicationRequest-a,* člana unije *msgData*

4.2.1.2. Programska nit koja izvršava operacije

Programska nit koja izvršava operacije jeste telo funkcije *DSMCC_GLUE_Task_Main.* Ova funkcija je realizovana po sledećem algoritmu:

1. Uzme se poruka iz *FIFO* reda u koji korisnici stavljaju poruke. Ovaj red se kreira posebnom operacijom koja pripada već pomenutom *CHAL-u.* Takodje, postoje posebne operacije iz *CHAL-a* kojima se poruka stavlja i uzima iz reda.
2. Ukoliko je poruka tipa *eDSMCC_GLUE_TYPE_LOAD_APPLICATION_BY_PID* poziva se operacija *DSMCC_GLUE_Task_LoadApplicationByPID()* koja vrši učitavanje čitave HbbTV aplikacije na lokalni sistem datoteka. Argumenti potrebni da bi se ova funkcija pozvala uzimaju se iz člana *applicationRequest* unije *msgData.* Pre i nakon ovog poziva se pozivaju operacije

wait i *signal* na semaforu koji služi za sinhronizaciju izvršavanja ove operacije, da se ne bi desilo da se operacija ponovo pozove, a da se prethodni poziv još uvek nije završio.

3. Ukoliko je poruka tipa *eDSMCC_GLUE_TYPE_TERMINATE* vrši se zaustavljanje niti. Naime nit se izvršava kao petlja koja uzima poruke iz reda i izvršava operacije. Uslov petlje je jedna promenljiva čija vrednost se postavlja na 0 ukoliko je poruka tipa *eDSMCC_GLUE_TYPE_TERMINATE* i time se petlja završava.

4.2.1.3. Realizacija funkcije DSMCC_GLUE_Task_Init()

Ova funkcija vrši kreiranje već pomenutog reda za poruke. Pravi novu nit od funkcije *DSMCC_GLUE_Task_Main.* Kreira semafore potrebne za sinhronizaciju asinhronih operacija. Dakle, ova funkcija vrši inicijalizaciju ovog dela *DSMCC Glue* podmodula. Sve to radi koristeći funkcionalnosti iz CHAL-a.

4.2.1.4. Realizacija funkcije DSMCC_GLUE_Task_Term()

Ova funkcija vrši obrnut proces u odnosu na funkciju DSMCC_GLUE_Task_Init(). Uklanja red kreiran ovom funkcijom, gasi nit koja služi za izvršavanje operacija i briše semafore kreirane u DSMCC_GLUE_Task_Init(). Sve to radi koristeći funkcionalnosti iz CHAL-a.

4.2.1.5. Realizacija funkcije DSMCC_GLUE_Task_PostMessage()

Ova funkcija vrši smeštanje poruke u red. Poruku koja joj se prosledi smešta u red koristeći funkcionalnosti iz CHAL-a.

4.2.1.6. Realizacija funkcije DSMCC_GLUE_Task_LoadApplicationByPID()

Ova funkcija vrši učitavanje *HbbTV* aplikacije na lokalni sistem datoteka na osnovu *PID-a*. Ovo čini po sledećem algoritmu:

1. Najpre poziva funkciju *DSMCC_Mount()* koja otpočinje smeštanje aplikacije na sistem datoteka organizovan u operativnoj memoriji. Argumente koji su potrebni za poziv ove funkcije dobija kao svoje argumente. Ovaj poziv može rezultovati i neuspehom te se pokušava ponovo sve dok poziv funkcije *DSMCC_Mount()* ne uspe. Između poziva se čeka jedna sekunda.
2. Poziva se funkcija *DSMCC_loadDSMCCFiles()*.

4.2.2. Deo *DSMCC Glue* podmodula koji služi kao nova sprega ka *DSMCC* funkcionalnostima

4.2.2.1. Realizacija funkcije DSMCC_GLUE_Init()

Ova funkcija inicijalizije *DSMCC Glue* podmodul. Inicijalizuje promenljive koje se koriste u pod modulu i poziva *DSMCC_GLUE_Task_Init()* da pokrene programsku nit koja služi za asinhrono izvršavanje operacija.

4.2.2.2. Realizacija funkcije DSMCC_GLUE_Term()

Funkcija postavlja promenljive koje se koriste u *DSMCC Glue* podmodulu i gasi programsku nit koja služi za asinhrono izvršavanje operacija pozivajući funkciju *DSMCC_GLUE_Task_Term().*

4.2.2.3. Realizacija funkcije DSMCC_GLUE_PrintListOfHbbTVApplications()

Funkcija na konzoli štampa listu *HbbTV* aplikacija koje su prisutne na trenutnom servisu. Informacije o aplikacijama uzima iz niza koji popunjava već pomenuta *DSMCC_GetDataBroadcastIdDescriptors()* funkcija.

4.2.2.4. Realizacija funkcije DSMCC_GLUE_LoadApplicationByPID()

Ova funkcija pravi poruku koja služi da bi se pokrenulo učitavanje čitave *HbbTV* aplikacije na lokalni sistem datoteka. U tom cilju formira poruku i poruku smešta u već pomenuti red. Ovo čini pozivajući funkcionalnosti iz dela *DSMCC Glue-a* koji služi za asinhroni poziv funkcija, a koji je pomenut u prethodnom delu. Poruka je tipa *eDSMCC_GLUE_TYPE_LOAD_APPLICATION_BY_PID*, a argumenti potrebni za poziv se upisuju u polja *applicationRequest-a* poruke.

5. TESTIRANJE I REZULTATI

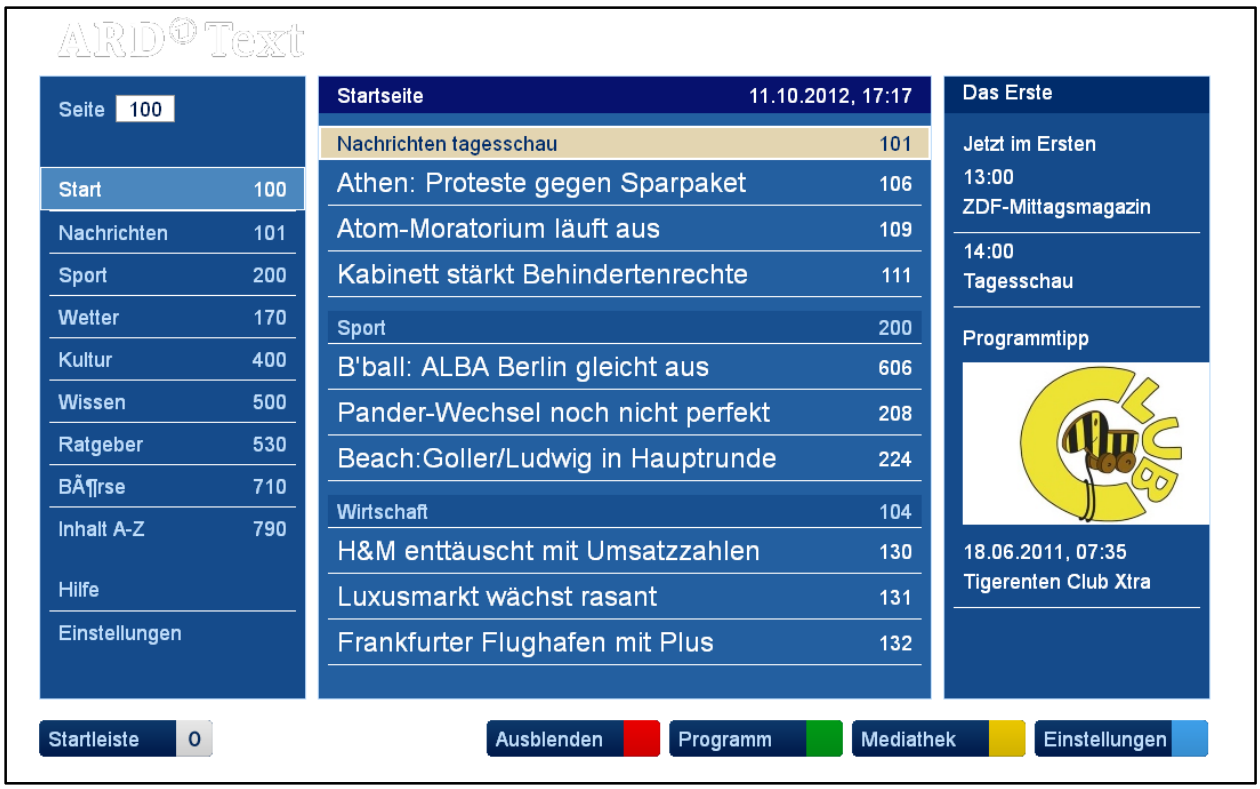
Testiranje ovog rešenja je rađeno na dve platforme. Prva platforma je bila *PC* računar sa linux operativnim sistemom. Ovom prilikom je korišćen tok podatka koji se nalazi u datoteci koja se nalazi na lokalnom sistemu datoteka. Druga platforma je bila *Marvell* ploča. (*Marvell* je firma iz Sjedinjenih Američkih Država koja se, izmedju ostalog, bavi i proizvodnjom uređaja iz oblasti digitalne televizije. Pod terminom *Marvell* ploča se misli na sistem na čipu koji ima neophodne komponente koje su potrebne za reprodukciju digitalne televizije. Neke od ovih komponenti su TV tjuner, mikroprocesor, operativna memorija, stalna memorija itd.) Na *Marvell* platformi kao tok podataka koriščen je tok koji dolazi iz kabla.

Tok na kome je vršeno testiranje je sadržao servis koji je imao dve *HbbTV* aplikacije. Prva aplikacija je bila početna aplikacija (eng. “start up”). Ova aplikacija je relativno jednostavne strukture. Ima nekoliko datoteka u korenom direktorijumu i pored ovih dateteka još jedan direktorijum. Ovaj direktorijum koji se nalazi u korenom direktorijumu sadrži nekoliko datoteka. Učitavanje aplikacije je trajalo od 5 do 8 sekundi na *PC* računaru, dok je na *Marvell* ploči trajalo 3 do 4 sekunde. Treba napomenuti da vreme učitavanja aplikacije zavisi od trenutka u kome *DSMCC* modul počne smeštanje aplikacije. Ukoliko smeštanje počne u trenutku kada se emituje prvi modul toka cikličnih objekata učitavanje traje najkraće. Ako smeštanje počne u trenutku kada se emituje drugi modul toka cikličnih objekata učitavanje traje najduže, jer se mora sačekati da prođe pun krug dok se ponovo ne emituje prvi modul. Dakle, vreme zavisi od trenutka u kom se počne sa smeštanjem aplikacije u operativnu memoriju. Primer učitane početne aplikacije je prikazan na sledećoj slici:

Slika 9: Početna (eng. "star up") HbbTV aplikacija

Druga aplikacija koja je učitana je teletext. Ova aplikacija ima kompleksnu strukturu direktorijuma i poddirektorijuma u kojima se nalazi preko hiljadu datoteka. Zbog svoje kompleksnosti i veličine bila je dobar test primer za testiranje *DSMCC* modula. Vreme potrebno za učitavanje ove aplikacije iznosilo je na *PC* računaru od 3 minuta i 20 sekundi do 5 minuta. Dok je na *Marvell* ploči učitavanje trajalo od 25 sekundi do 40 sekundi. Razlika u vremenu neophodnom da se učita aplikacija

na *PC* računaru i *Marvell* ploči nastala je, jer linux nije bio operativni sistem domaćin na računaru na kome je testiranje vršeno već je korišćena virtuelna mašina. Operativni sistem domaćin je bio *Windows*. Iz tog razloga dosta vremena se trošilo na komunikaciju između operativnog sistema domaćina i virtuelne mašine. Ova aplikacija je prikazana na sledećoj slici:



Slika 10: HbbTV teletext aplikacija

6. ZALJUČAK

U ovom radu je realizovana nova sprega između korisnika i *DSMCC* modula. Funkcionalnosti nove sprege omogućavaju korisniku da zahteva operacije koje njemu odgovaraju. Korisnik sada može da dobavi informacije o tome koje se aplikacije nalaze u okviru televizijskog servisa tj. da postane "svestan" sadržaja servisa u njegovom interaktivnom delu. Nakon toga može da zahteva učitavanje aplikacija na lokalni sistem datoteka na putanju na koju želi. Upotreba ovako realizovane sprege je vrlo jednostavna. Ukoliko bude bilo neophodno nova sprega se može vrlo lako proširiti novim funkcionalnostima da bi zadovoljila neke nove potrebe korisnika. Na taj način omogućena je velika fleksibilnost sprege, a samim tim i načina na koji se DSM-CC modul koristi. Ono sto je vrlo bitno jeste da su pozivi ka vremenski dugotrajnim operacijama iz nove sprege asinhroni, tako da se korisnik neće blokirati na vremenski dugom pozivu iz sprege već će moći da vrši neku drugu obradu dok operacija iz sprege traje. U potpunosti je eliminisana potreba da korisnik unapred poznaje strukturu direktorijuma i datoteka aplikacije, već samo treba da uputi zahtev za učitavanjem aplikacije ka sprezi i funkcije sprege će aplikaciju učitati na lokalni sistem datoteka. Hijerarhija direktorijuma i raspored datoteka po direktorijumima će biti isti kao kada je aplikacija poslata. Održavanje iste hijerarhije je neophodno iz razloga što datoteke aplikacije mogu referencirati jedna drugu relativno u odnosu na koreni direktorijum aplikacije. Ovo referenciranje koristi *WebKit* kada izvršava aplikaciju te je bitno održati hijerarhiju. Takođe, izbegnuta je potreba da se menja izvorni kod *WebKit-a* na način da *WebKit* poziva funkcije iz *DSMCC* modula, već se aplikacija učita na lokalni sistem datoteka. *WebKit* već ima mogućnost da izvršava aplikaciju ako se ona nalazi na lokalnom sistemu datoteka, tako da nije bilo potrebno modifikovati *WebKit.* Ovako realizovana sprega se može vrlo lako iskoristiti od strane drugih modula koji bi koristili funkcionalnosti *DSMCC* modula. Postoje i druge vrste aplikacija koje se mogu slati putem toka cikličnih objekata (ove aplikacije mogu biti npr. *MHEG* ili *MHP* aplikacije). Ove aplikacije imaju svoje module koji služe za njihovo prikazivanje, tako da i ovi moduli mogu koristiti funkcionalnosti nove sprege za svoje potrebe. Time je osnovni cilj kod realizacije sprege ispunjen. Realizovana je asinhrona, nezavisna, funkcionalna i lako proširiva sprega koja se može koristiti od strane raznih programskih modula.

7. LITERATURA

[1] DVB standard : ETSI EN 300 468 V1.5.1

[2] MPEG-2 : ISO/IEC 13818

[3] HBBTV standard : HbbTV statndard revision : ETSI TS 102 796 v1.1.1

[4] DSM-CC standard : ISO/IEC 13818-10:1999

[5] Hongguang Zhang; Tianpu Jiang; Zhiqi Gu; Shibao Zheng; , *"Design and implementation of broadcast file system based on DSM-CC data carousel protocol,"* Consumer Electronics, IEEE Transactions on , vol.50, no.3, pp. 929- 933, Aug. 2004

[6] Lukac, Z.; Zlokolica, V.; Mlikota, B.; Radonjic, M.; Velikic, I.; , *"A testing methodology and system for functional verification of general HbbTV device,"* Consumer Electronics (ICCE), 2012 IEEE International Conference on , vol., no., pp.325-326, 13-16 Jan. 2012

[7] W. Fischer, "*Digital Video and Audio Broadcasting Technology,"* Springer Heidelberg Dordrecht London New York 2010