



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Давор Рапић

Имплементација РВУ послужитеља и његова интеграција у Qt радни оквир

МАСТЕР РАД

Нови Сад, 2014.

	УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА МАСТЕР РАД	Датум:

(Податке уноси предметни наставник - ментор)

СТУДИЈСКИ ПРОГРАМ:	Рачунарство и аутоматика
РУКОВОДИЛАЦ СТУДИЈСКОГ ПРОГРАМА:	Никола Јорговановић

Студент:	Давор Рапић	Број индекса:	Е2 14/2013
Област:	Архитектуре и алгоритми ДСП-а 1		
Ментор:	др Јелена Ковачевић		
НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА: - проблем – тема рада; - начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;			

НАСЛОВ МАСТЕР РАДА:

Имплементација RBU послужитеља и његова интеграција у Qt радни оквир
--

ТЕКСТ ЗАДАТКА:

Потребно је реализовати RBU послужитеља и извршити његову интеграцију у Qt радни оквир путем графичког и мултимедијалног Qt додатка. Интеграцијом RBU концепта у Qt радни оквир омогућен је развој Qt апликација које се извршавају унутар кућне мреже. Апликација се чува и извршава на RBU послужитељу, док се исцртавање графичке корисничке спреге и репродукција мултимедијалних датотека обавља на клијентској страни.

Руководилац студијског програма:	Ментор рада:

Примерак за: ☐ - Студента; ☐ - Ментора
--



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР :	
Идентификациони број, ИБР :	
Тип документације, ТД :	Монографска документација
Тип записа, ТЗ :	Текстуални штампани материјал
Врста рада, ВР :	Дипломски – мастер рад
Аутор, АУ :	Давор Рапић
Ментор, МН :	проф. др. Јелена Ковачевић
Наслов рада, НР :	Имплементација РВУ послужитеља и његова интеграција у Qt радни оквир
Језик публикације, ЈП :	Српски / латиница
Језик извода, ЈИ :	Српски
Земља публиковања, ЗП :	Република Србија
Уже географско подручје, УГП :	Војводина
Година, ГО :	2014
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)	6/76/12/31/6/0/0
Научна област, НО :	Електротехника и рачунарство
Научна дисциплина, НД :	Рачунарска техника
Предметна одредница/Кључне речи, ПО :	РВУ послужитељ, РВУ протокол, ДЛНА, Qt радни оквир
УДК	
Чува се, ЧУ :	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, ВН :	
Извод, ИЗ :	У раду је представљена реализација РВУ послужитеља и његова интеграција у Qt радни оквир путем графичког и мултимедијалног Qt додатка. Коришћењем РВУ протокола омогућена је идентична корисничка спрега без обзира на тип платформе на којој се извршава РВУ клијент. Креиране Qt апликације се извршавају на РВУ послужитељу, док се исцртавање графичке корисничке спреге и репродукција мултимедијалних датотека обавља на РВУ клијенту.
Датум прихватања теме, ДП :	
Датум одбране, ДО :	
Чланови комисије, КО :	Председник: проф. др. Никола Теслић
	Члан: проф. др. Бранислав Тодоровић
	Члан, ментор: проф. др. Јелена Ковачевић
	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 :	Davor Rapić		
Mentor, MN :	PhD Jelena Kovačević		
Title, TI :	The implementation of RVU server and its integration into Qt framework		
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 :	2014		
Publisher, PB :	Author's reprint		
Publication place, PP :	Novi Sad, Dositeja Obradovica sq. 6		
Physical description, PD : (chapters/pages/ref./tables/pictures/graphs/appendixes)	6/76/12/31/6/0/0		
Scientific field, SF :	Electrical Engineering		
Scientific discipline, SD :	Computer Engineering, Engineering of Computer Based Systems		
Subject/Key words, S/KW :	RVU server, RVU protocol, DLNA, Qt framework		
UC			
Holding data, HD :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia		
Note, N :			
Abstract, AB :	This paper describes implementation of RVU server and its intergration into Qt framework through graphic and multimedia Qt plug-in. Identical graphical user interface on various platform is achieved by using RVU protocol. Qt application is executed on the RVU server while graphical user interface rendering and A/V content reproduction is performed on the RVU client.		
Accepted by the Scientific Board on, ASB :			
Defended on, DE :			
Defended Board, DB :	President:	PhD Nikola Teslić	Menthor's sign
	Member:	PhD Branislav Todorović	
	Member, Mentor:	PhD Jelena Kovačević	

Zahvalnost

Zahvaljujem se institutu RT-RK na pruženoj podršci u toku realizacije projekta, kao i stručnim saradnicima Nikoli Kuzmanoviću, Goranu Miljkoviću i Jeleni Kovačević na savetima i pomoći tokom izrade završnog rada.

SADRŽAJ

1. Uvod.....	3
2. Teorijske osnove	5
2.1 RVU protokol.....	5
2.1.1 RVU poslužitelj	6
2.1.1.1 DLNA	6
2.1.1.1.1 DMS, DMR, DMC.....	7
2.1.1.1.2 Modeli interakcije DLNA klasa	8
2.1.1.1.3 Poslužitelj udaljene korisničke sprege	8
2.1.1.2 RVU-RUI protokol.....	8
2.2 Qt radni okvir	10
2.2.1 Qt QPA apstrakcioni sloj	11
2.2.2 Qt medija plejer	12
3. Analiza zahteva i koncept rešenja.....	13
3.1 RVU poslužitelj.....	13
3.1.1 Adresiranje, pronalaženje i opis RVU uređaja	14
3.1.1.1 Gradivne komponente RVU poslužitelja.....	16
3.1.1.2 Gradivne komponente RVU klijenta	18
3.1.1.3 Koncept rešenja strukture RVU poslužitelja	19
3.1.2 Upravljanje sesijama.....	21
3.1.3 Udaljena grafička korisnička sprega.....	22
3.1.3.1 TCP tokovi - kanali	22
3.1.3.2 Vremenska ograničenja	24
3.1.3.3 RVU-RUI tipovi bafera	24

3.1.3.4	RVU-RUI komande	25
3.1.3.5	Struktura RVU-RUI poslužitelja	27
3.1.4	Prenos multimedijalnih datoteka	29
3.1.5	Kvalitet usluge i dijagnostika	30
3.1.6	Pribavljanje datoteka	30
3.1.7	Aplikativna programska sprega RVU poslužitelja	31
3.2	Qt dodaci	31
3.2.1	Qt RVU-RUI dodatak	31
3.2.2	Qt DLNA medija plejer dodatak	34
3.3	Qt radni okvir	36
4.	Programsko rešenje	37
4.1	RVU poslužitelj	38
4.1.1	DMS i RUI poslužitelj	38
4.1.2	DMC	44
4.1.3	RVU-RUI poslužitelj	48
4.1.3.1	Modul sa alatima	48
4.1.3.2	Komunikacioni modul	49
4.1.3.3	Modul za rukovanje kanalima	51
4.1.3.4	Modul za rukovanje RVU-RUI porukama	52
4.1.3.5	Modul za rukovanje RVU-RUI klijentima	54
4.1.3.6	Modul za rukovanje baferima RVU-RUI klijenta	54
4.1.3.7	Modul za rukovanje komandama	55
4.1.3.8	Centralni modul	58
4.1.3.9	Aplikativna programska sprega RVU-RUI poslužitelja	59
4.1.3.10	Interakcija između gradivnih komponenti RVU-RUI poslužitelja	62
4.2	Qt dodaci	66
4.2.1	Qt RVU-RUI dodatak	66
4.2.2	Qt DLNA medija plejer dodatak	69
4.3	Qt radni okvir	70
5.	Ispitivanje i verifikacija rešenja	72
6.	Zaključak	75
7.	Literatura	76

SPISAK SLIKA

Slika 1. Kućna mreža sa RVU kompatibilnim uređajima	4
Slika 2. Prikaz gradivnih komponenti RVU poslužitelja i RVU klijenta	6
Slika 3. Pregled modula Qt radnog okvira	10
Slika 4. Pregled klasa QPA sloja.....	11
Slika 5. Prikaz sekvenci neophodnih protokola po pokretanju platforme	14
Slika 6. Ključ za razumevanje strukture UPnP uređaja i servisa	14
Slika 7. Strukture RVU UPnP poslužitelja i klijenta	15
Slika 8. Struktura RVU poslužitelja i klijenta.....	15
Slika 9. Dijagram stanja DMR RVU klijenta	18
Slika 10. Usvojena struktura RVU poslužitelja	19
Slika 11. Uspostavljanje RVU-RUI sesije	21
Slika 12. Format poruke sa podacima	23
Slika 13. Struktura RVU-RUI poslužitelja organizovana kao programski stek.....	27
Slika 14. Ažuriranje RVU programske podrške klijenta	31
Slika 15. Prikaz strukture Qt RVU-RUI dodatka.....	33
Slika 16. Prikaz dodavanja novog klijenta i sprežnih objekata.....	33
Slika 17. Prikaz struktura Qt DLNA medija plejer dodatka	34
Slika 18. Prikaz scenarija za različite vrste događaja u komunikacionom modulu	50
Slika 19. Životni vek kanala	52
Slika 20. Proces obrade podataka.....	53
Slika 21. Algoritam zaključavanja bafera	56
Slika 22. Algoritam otključavanja bafera.....	57
Slika 23. Inicijalizacija sesije i konfiguracija klijenta	63

Slika 24. Prikaz karakteristične sekvence grafičkih komandi.....	64
Slika 25. Prikaz karakteristične sekvence audio komandi	64
Slika 26. Prikaz konfiguracije video prozora	65
Slika 27. Dijagram klasa Qt RVU-RUI dodatka.....	67
Slika 28. Proces dodavanja klijenta u Qt RVU-RUI dodatku.....	68
Slika 29. Dijagram klasa Qt DLNA medija plejer dodatka	69
Slika 30. Prikaz ispitne Qt aplikacije	73
Slika 31. Izgled DLNA pretraživač aplikacije na RVU klijentu.....	74

SPISAK TABELA

Tabela 1. Komande za uspostavu, raskid i informacije o RVU-RUI sesiji	25
Tabela 2. Komande za opis korisničkih događaja.....	25
Tabela 3. Grafičke komande	26
Tabela 4. Komande za opis lokalne GKS-e	26
Tabela 5. Komande za prikaz.....	27
Tabela 6. Audio komande	27

SKRAĆENICE

DLNA	- <i>Digital Live Network Alliance</i> , Digitalna mrežna alijansa
DTCP-IP	- <i>Digital Transmission Content Protection over IP</i> , Zaštita digitalnog sadržaja prilikom prenosa putem mreže zasnovane na IP protokolu
RUI	- <i>Remote User Interface</i> , Udaljena korisnička sprega
QPA	- <i>Qt Platform Abstraction</i> , Ku-te sloj za apstrakciju platforme
GKS	- Grafička korisnička sprega
DTV	- Digitalna televizija
CIA	- <i>Client Image Acquisition</i> , Akvizicija klijentske programske podrške
HTTP	- <i>Hypertext Transfer Protocol</i> , Protokol za prenos hiper-teksta
UPnP	- <i>Universal Plug and Play</i> , Grupa mrežnih protokola za lak i jednostavan pristup mreži
SSDP	- <i>Simple Service Discovery Protocol</i> , Protokol za otkrivanje uređaja na mreži
SOAP	- <i>Simple Object Access Protocol</i> , Protokol za pristup poljima objekta
GENA	- <i>General Event Notification Architecture</i> , Protokol za dostavu informacija o događajima
RTSP	- <i>Real Time Streaming Protocol</i> , Protokol za prenos podataka u realnom vremenu
DMS	- <i>Digital Media Server</i> , Digitalni medija poslužitelj
DMR	- <i>Digital Media Renderer</i> , Digitalni modul za vizualizaciju sadržaja
DMC	- <i>Digital Media Controller</i> , Digitalni medija kontroler
RUISrc	- <i>Remote User Interface Source</i> , Izvor udaljene korisničke sprege
RUIS	- <i>Remote User Interface Server</i> , Poslužitelj udaljene korisničke sprege

RUIC	- <i>Remote User Interface Controller</i> , Kontroler udaljene korisničke sprege
RUISink	- <i>Remote User Interface Sink</i> , Odredište udaljene korisničke sprege
HTML	- <i>HyperText Markup Language</i> , Jezik za označavanje hiper-teksta
EPG	- <i>Electronic Program Guide</i> , Elektronski programski vodič
PID	- <i>Packet Identifier</i> , Identifikator paketa
APS	- Aplikativna programska sprega
QoS	- <i>Quality of Service</i> , Kvalitet usluge
UDN	- <i>Unique Device Name</i> , Jedinstveno ime uređaja
URI	- <i>Uniform Resource Identifier</i> , Jedinstveni identifikator resursa
URL	- <i>Uniform Resource Locator</i> , Jedinstveni lokator resursa
MIME	- <i>Multipurpose Internet Mail Extensions</i> , Višenamenska proširenja internet pošte

1. Uvod

Cilj ovog rada je formiranje radnog okvira za razvoj udaljenih aplikacija, odnosno aplikacija koje se udaljeno iscertavaju na uređajima unutar kućne mreže. Ovaj cilj ostvaren je kreiranjem RVU poslužitelja definisanog RVU protokolom i njegovim sprežanjem sa Qt radnim okvirom. Qt aplikacija se izvršava na RVU poslužitelju, dok se iscertavanje grafičke korisničke sprege i reprodukcija multimedijalnog sadržaja obavlja na jednom ili više RVU klijenata. Kako bi se razvoj udaljenih aplikacija učinio što bržim i jednostavnijim iskorišten je Qt radni okvir. Sprega između Qt radnog okvira i RVU poslužitelja ostvarena je realizacijom Qt RVU-RUI i Qt DLNA medija plejer dodataka (engl. plug-in).

Rešenje je razvijeno i verifikovano na Linux i Android platformi, pri čemu je po potrebi moguće brzo i jednostavno izvršiti prilagođenje na bilo koju platformu podržanu od strane Qt radnog okvira.

U današnje vreme svedoci smo sve većeg i bržeg razvoja digitalnih uređaja (mobilni uređaji, tablet računari, televizori, itd.) koji su povezani unutar kućne mreže kako bi se omogućila međusobna interakcija i razmena multimedijalnog sadržaja. Kućna mreža predstavlja poseban tip lokalne mreže koja se razvila upravo iz potrebe da se omogući komunikacija između digitalnih uređaja prisutnih u kući ili njenoj neposrednoj blizini.

RVU alijansa definiše RVU protokol [1] koji se u velikoj meri zasniva na već postojećem DLNA standardu kako bi se omogućio prenos i reprodukciju multimedijalnog sadržaja [2]. Zaštita prilikom prenosa ostvarena je putem DTCP-IP protokola. Na ovaj način distributeri imaju mogućnost isporuke plaćenog sadržaja. Sa druge strane kako bi se omogućila identična grafička korisnička sprega na različitim platformama (Linux, Android, iOS, Windows Phone) RVU protokol definiše poseban RVU-RUI protokol. RVU-RUI protokol omogućuje udaljenu kontrolu grafičke korisničke sprege na nivou pojedinačnog piksela.



Slika 1. Kućna mreža sa RVU kompatibilnim uređajima

Qt radni okvir je široko korišten na različitim platformama za kreiranje aplikacija sa grafičkom korisničkom spregom [3]. Prilikom realizacije ovoga projekta korištena je Qt verzija 5 koja uvodi poseban sloj, tzv. QPA sloj [4], za apstrakciju konkretne hardverske platforme. Realizacijom QPA sloja omogućuje se izvršavanje Qt aplikacija na određenoj platformi.

Kreiranjem Qt RVU-RUI dodatka, zasnovanog na QPA sloju, kao veze između RVU poslužitelja i Qt radnog okvira omogućeno je udaljeno iscertavanje Qt aplikacija [5]. Kako bi se omogućila i reprodukcija multimedijalnog sadržaja realizovan je Qt DLNA medija plejer dodatak koji takođe predstavlja vezu između RVU poslužitelja i Qt radnog okvira i kojim se vrši kontrola udaljenog plejera.

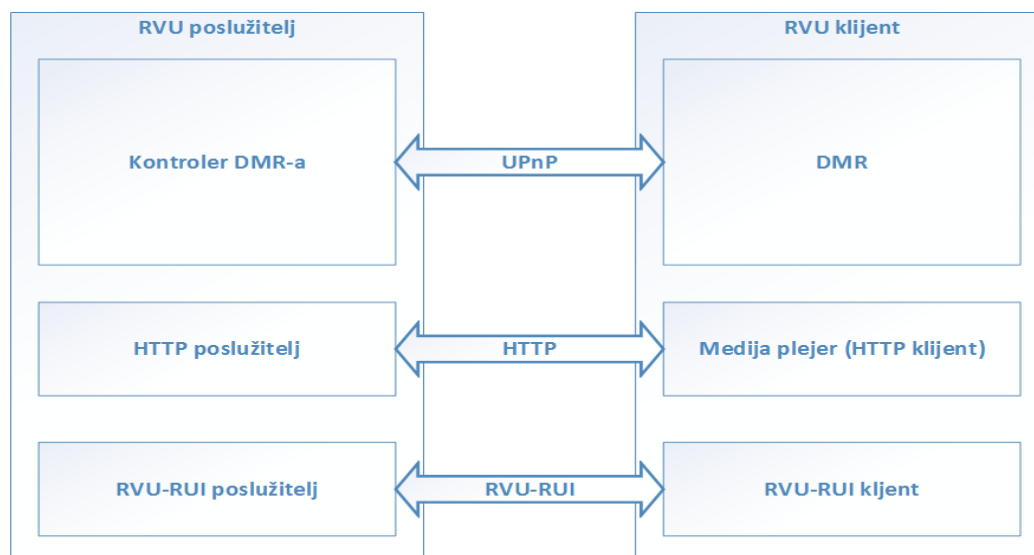
U drugom poglavlju dat je opis korištenih tehnologija. Treće poglavlje predstavlja koncept rešenja u kojem su opisane gradivne komponente RVU poslužitelja sa naglaskom na RVU-RUI poslužitelja. Definisana je veza između RVU poslužitelja i Qt radnog okvira, odnosno struktura Qt RVU-RUI i Qt DLNA medija plejer dodatka, nakon čega je opisana interakcija između ovih modula. U četvrtom poglavlju opisano je programsko rešenje. Peto poglavlje sadrži verifikacija realizovanog rešenja. Poslednje poglavlje je retrospektiva urađenog sa pravcima budućeg razvoja.

2. Teorijske osnove

U ovom poglavlju dat je opis korištenih tehnologija, odnosno RVU protokola i Qt radnog okvira. Prikazana je globalna struktura RVU poslužitelja i RVU klijenta, kao i protokoli putem kojih oni međusobno intereaguju. RVU protokol se u značajnoj meri oslanja na postojeći DLNA standard, zbog čega je dat pregled DLNA uređaja (engl. DLNA classes) koji su korišteni u ovom rešenju. Uveden je RVU-RUI protokol i izvršena njegova klasifikacija u okviru ostalih RUI protokola. U posljednjem poglavlju teorijskih osnova opisan je Qt radni okvir sa naglaskom na Qt QPA i Qt medija plejer dodatke koji su iskorišteni za spregu sa RVU poslužiteljem.

2.1 RVU protokol

Arhitektura sistema zasnovanog na RVU protokolu je tipa klijent-poslužitelj. RVU poslužitelj omogućuje distribuciju i upravljanje reprodukcijom multimedijalnog sadržaja, kao i dostavljanje grafičke korisničke sprege. Omogućena je zaštićena isporuka plaćenog sadržaja (engl. premium content) upotrebom DTCP-IP protokola, što je od posebne važnosti za distributere DTV servisa. RVU poslužitelj je centralizovani uređaj koji je najčešće isporučen od strane distributera sadržaja i u stanju je da opsluži jednog ili više RVU klijenata unutar kućne mreže. Jedna od glavnih prednosti RVU protokola je uvođenje RVU-RUI protokola koji omogućuje isporuku GKS-e do RVU klijenata, pri čemu je izgled GKS-e nezavisan od konkretne platforme. Na slici koja sledi je prikaz strukture RVU poslužitelja i RVU klijenta.



Slika 2. Prikaz gradivnih komponenti RVU poslužitelja i RVU klijenta

2.1.1 RVU poslužitelj

RVU poslužitelj treba da omogući lako otkrivanje od strane RVU klijenata i povezivanje sa njima u uslovima kućne mreže u kojoj u svakom trenutku može da postoji više različitih tipova uređaja, pri čemu svaki od njih može da ima različitu funkcionalnost. Kako bi ispunili ove zahteve RVU poslužitelj i klijent se baziraju na DLNA standardu, koji takođe specificira i način isporuke multimedijalnog sadržaja. Sa druge strane RVU poslužitelj treba da omogući i isporuku grafičke korisničke sprege. Ovaj zahtev ispunjen je uvođenjem ugrađenog RUI poslužitelja [6] koji je definisan od strane DLNA standarda, dok je tip RUI protokola potpuno nov u odnosu na prethodna rešenja. RVU alijansa uvodi novi RVU-RUI protokol. Poslednja bitna funkcionalnost RVU poslužitelja je mogućnost dobavljanja ili ažuriranja neke datoteke na RVU klijentu putem CIA protokola definisanog od strane RVU alijanse.

2.1.1.1 DLNA

DLNA alijansa definisala je grupu standarda i direktiva kako bi obezbedila interoperabilnost između multimedijalnih uređaja unutar kućne mreže. Ovi standardi, odnosno direktive su zasnovane na već postojećim i široko korištenim standardima kao što su UPnP i HTTP. UPnP [7] predstavlja grupu mrežnih protokola koji se koriste od strane DLNA standarda kako bi se obezbedilo otkrivanje i kontrola uređaja dostupnih unutar kućne mreže. Upravljanje multimedijalnim datotekama, odnosno upravljanje reprodukcijom multimedijalnih datoteka takođe je bazirano na UPnP protokolu.

UPnP se sastoji iz sledećih protokola:

- **SSDP** protokol omogućuje reklamiranje uređaja na mreži, odnosno sa druge strane omogućuje otkrivanje dostupnih uređaja.
- **SOAP** protokol omogućuje razmenu poruka (UPnP akcija)
- **GENA** protokol omogućuje razmenu informacija o promeni stanja uređaja, tj. omogućuje isporuku informacija o promeni stanja nekog uređaja ostalim uređajima u mreži.
- **HTTP** protokol omogućuje preuzimanje dokumenata sa opisom uređaja ili servisa koji definišu njihovu funkcionalnost.

Prenos multimedijalnih datoteka ostvaren je putem HTTP protokola koji je obavezan ili opciono putem RTSP protokola. Prilikom prenosa sadržaj je moguće zaštititi korištenjem pomenutog DTCP-IP protokola.

2.1.1.1.1 DMS, DMR, DMC

Uređaji unutar kućne mreže se mogu podeliti prema klasi (engl. device classes). Detaljnije uputstvo o kategorizaciji uređaja se nalazi u [8]. Svaka od ovih klasa realizuje određenu DLNA funkcionalnost (engl. Device Capabilities), pri čemu su neke od njih obavezne, a neke opcione za određene klase uređaja. Opisi uređaja definisani su od strane UPnP-a u dokumentu [9].

Klase uređaja koje su od značaja za ovaj projekat su:

- **DMS** predstavlja klasu koja omogućuje korisnicima deljenje multimedijalnih datoteka (videa, muzike, slika) sa drugima. Pored datoteka omogućeno je i deljenje objekata frekvencijskih odabirača (engl. tuner object) koji omogućuje modelovanje frekvencijskih odabirača digitalnog televizijskog prijemnika. Na ovaj način DTV servise je moguće distribuirati do uređaja unutar kućne mreže. Sadržaj koji se deli može biti zaštićen, pri čemu se zaštita sprovodi upotrebom DTCP-IP protokola. Ovaj princip omogućuje zaštitu plaćenog sadržaja, tj. omogućuje pristup samo onima koji su za to platili određenu novčanu nadoknadu.
- **DMR** je klasa koja pruža podršku za kontrolu reprodukcije multimedijalnih datoteka. Svakom pojedinačnom DMR-u može se pridružiti više različitih modula za reprodukciju (engl. User-Agent), npr. modul za reprodukciju videa, audia, slika. Na ovaj način kada korisnik želi da reprodukuje određeni tip multimedijalne datoteke automatski će biti izabran adekvatan modul.

- **DMC** je klasa koju nije moguće otkriti na mreži već služi za kontrolu prethodne dve, odnosno uz pomoć DMC-a moguće je odabrati određenu datoteku na DMS-u i reprodukovati je na DMR-u.

2.1.1.1.2 Modeli interakcije DLNA klasa

Modeli interakcije DLNA klasa (engl. DLNA system usage) definišu koje DLNA klase učestvuju u interakciji kao i ulogu svake od njih. Svi modeli predstavljeni su u dokumentu [8], pri čemu su za realizaciju ovog rešenja od značaja sledeće dve:

- Model dvostruke kutije (engl. 2-box push system usage) podrazumeva upotrebu kontrolera DMR-a koji dostavlja multimedijalni sadržaj do DMR-a u cilju njegove reprodukcije.
- Model trostruke kutije (engl. 3-box system usage) podrazumeva postojanje DMC-a koji vrši pretragu datoteka koje se nalaze na DMS-u i njihovu reprodukciju na DMR-u.

2.1.1.1.3 Poslužitelj udaljene korisničke sprege

RUI predstavlja koncept u kome centralizovani poslužitelj kontroliše fizički udaljene uređaje-klijente. Poslužitelj prima događaje od strane klijenata, nakon čega prevodi klijenta u sledeće adekvatno stanje. DLNA alijansa definiše kategorije i funkcionalnosti kako bi standardizovala ovaj vid upravljanja.

Izvor udaljene korisničke sprege - RUISrc predstavlja DLNA funkcionalnost ugrađenu u standardizovani UPnP uređaj RUI poslužitelj - RUIS. RUIS ima zadatak da omogući deljenje GKS-e. On se može ugraditi u neku od DLNA klasa, pri čemu je najprirodnije ugraditi ovu funkcionalnost u okviru DMS-a. Koristeći RUI kontroler (RUIC) moguće je prikazati odabranu GKS-u na nekom od RUI odredišta - RUISink. RUISink-u je pridružen korisnički agent (engl. User-Agent) koji vrši prikaz GKS-e. U zavisnosti od tipa GKS-e zavisi i tip korisničkog agenta (webkit, RVU-RUI klijent, ...). Neke od GKS-a koje je moguće deliti putem RUIS-a su: HTML, HTML5, CE-HTML, RVU-RUI (podprotokol RVU protokola).

2.1.1.2 RVU-RUI protokol

Dokument [10] analizira postojeće RUI protokole prema kojem se oni mogu podeliti u dve grupe:

- primitivni transfer RUI-a (engl. native RUI) - Kod ovog tipa RUI-ja poslužitelj skladišti, dostavlja grafičke komponente i određuje izgled GKS-e. Klijentska

programska podrška je “tanka” i njen jedini zadatak je iscrtavanje GKS-e i reagovanje na akcije korisnika koje se prosleđuju do poslužitelja.

- Prednosti:
 - isti izgled GKS-e na različitim tipovima klijentima
 - jednostavna programska podrška na klijentskoj strani (engl. “light-weight” client)
 - lakše održavanje i nadogradnja
 - smanjenje potrošnje energije i prostora na disku
- Mane:
 - opterećenost mreže
 - kašnjenje u toku dostavljanja grafičkih komponenti
 - prilagođavanje različitim rezolucijama

Najpoznatija rešenja iz ove grupe protokola su RFB (engl. Remote Frame Buffer), VNC (engl. Virtual Network Computing), RDP(engl Remote Desktop Protocol), itd.

- označeni RUI (engl. markup RUI) – Za razliku od prethodnog tipa RUI protokola, ovde je pristup problemu RUI-ja potpuno drugačiji. Ovi protokoli ne opslužuju klijente grafičkim komponentama, već im prosleđuju globalni opis i radni okvir GKS-e. Putem ovog radnog okvira, označeni protokoli opisuju koje oblike GKS-e bi klijentska programska podrška trebala da podrži i na koji način bi se oni trebali ponašati.

- Prednosti:
 - eliminacija opterećenja mreže
 - eliminacija kašnjenja prilikom prenosa grafičkih komponenti
- Mane:
 - klijentska programska podrška je značajno kompleksnija, prostorno i procesorski
 - izgled GKS-e ne mora biti isti na svim klijentima, već zavisi od realizacije programske podrške.

Najpoznatija rešenja iz ove grupe protokola su UPnP RUI protokoli (HTML, HTML5, CE-HTML).

Kako bi obezbedili udaljenu GKS-u na RVU klijentima RVU alijansa definisala je RVU-RUI protokol [1]. Prema načinu funkcionisanja ovaj protokol se nalazi u grupi primitivnih RUI-ja, dok se RVU klijent obaveštava o deljenim GKS-a na isti način kao kod UPnP RUI protokola.

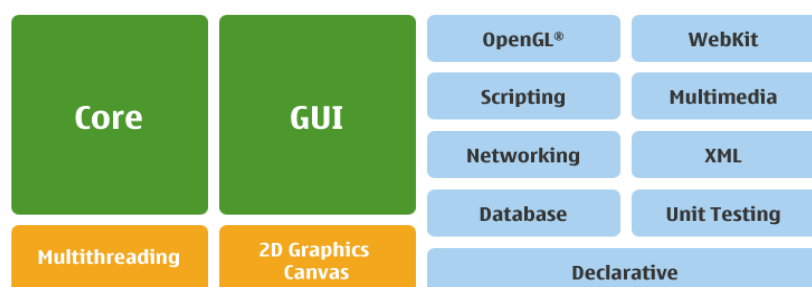
Razlozi za uvođenje novog tipa protokola su nemogućnost postojećih da ispune neophodne zahteve koji su postavljeni od strane korisnika, a samim tim i RVU alijanse [11], a to su:

- efikasan način opsluživanja više klijenata istovremeno
- identična GKS-a bez obzira na tip platforme
- potreba da klijent bude što jednostavniji kako bi samim tim bio i što jeftiniji

RVU-RUI protokol pored prenosa grafičkih komponenti, omogućuje i prenos audio efekata unutar GKS-e, pozicioniranje i konfiguraciju video prozora, a takođe pruža podršku i za konfiguraciju 3D videa. RVU-RUI protokol rešava ove zahteve tako što većinu posla obavlja direktno na poslužitelju, odnosno kompleksne grafičke operacije se obavljaju na poslužitelju, a zatim se vrši samo ažuriranje GKS-e na klijentu. Prenos multimedijalnih datoteka se vrši nezavisno od GKS-e, što je jedna od prednosti ovog rešenja, upotrebom modela dvostruke kutije. Konačan izgled GKS-e dobija se spajanjem grafičke ravni i jedne ili više video ravni. Efikasno opsluživanje više klijenata istovremeno omogućeno je zahvaljujući mogućnosti svakog od klijenata da uspostavi više TCP veza, tzv. kanala sa poslužiteljem, pri čemu je prva veza otvorena od strane klijenta, a ostale u zavisnosti od potreba otvara poslužitelj ili klijent. Kanali se mogu podeliti na dva tipa komandne kanale i kanale za podatke. Ovaj princip omogućuje minimalno kašnjenje komandi prilikom prenosa, zbog toga što se grafičke komponente prenose posebnim putem. Usled razdvojenosti podataka i komandi potrebno je definisati vremenske zahteve, tj. definisati maksimalna dozvoljena kašnjenja, što je urađeno u RVU specifikaciji.

2.2 Qt radni okvir

Qt radni okvir omogućuje razvoj grafičkih i konzolnih aplikacija za personalne računare, ugrađene i mobilne platforme (Linux, Windows, Android, iOS, itd.). Ovaj radni okvir je vođen događajima (engl. event driven framework). Programiranje vođeno događajima omogućeno je zahvaljujući posebnom kod generatoru (engl. meta object compiler - .moc) i makroima koji proširuju standardni C++.



Slika 3. Pregled modula Qt radnog okvira

U zavisnosti od parametara koji se proslede prilikom pokretanja, ista Qt aplikacija se može izvršavati na različitim platformama. Ovi parametri određuju koji dodaci (engl. plug-in) će se učitati prilikom pokretanja i na taj način omogućiti spregu sa platformom. Uz radni okvir dolaze standardni dodaci za Linux, Windows, Android, iOS, itd., pri čemu je po potrebi moguće realizovati sopstvene dodatke. Ovaj princip korišten je kako bi se omogućilo izvršavanje Qt radnog okvira na RVU poslužitelju, odnosno kako bi se formirala veza između njih. U ovom projektu su razvijena dva dodatka:

- Qt RVU-RUI dodatak – apstrakcija sistema prozora (engl. window system integration)
- Qt DLNA medija plejer dodatak – apstrakcija DLNA modela trostruke kutije

2.2.1 Qt QPA apstrakcioni sloj

Proces prenošenja (engl. porting) Qt radnog okvira na druge platforme u verziji četiri je bio veoma složen i dugotrajan proces. Kako bi se ovaj proces ubrzao QWS (engl. Qt Window System) sloj je u verziji pet, koja je korištena u realizaciji ovog projekta, zamenjen sa QPA slojem. Realizacijom QPA sloja za određenu platformu omogućeno je izvršavanje Qt radnog okvira na toj platformi, odnosno Qt aplikacija.

Qt QPA apstrakcioni sloj se sastoji iz različitih QPlatform apstraktnih klasa. Ove klase se mogu podeliti u dve grupe, prva koja služi za integraciju sistema prozora i druge koja omogućuje odabir teme i stilova grafičke korisničke sprege na konkretnoj platformi. Modul za rukovanje sistemom prozora omogućuje pozicioniranje i promenu veličine prozora, kao i iscrtavanje bafera okvira (engl. frame buffer), odnosno jednog njegovog dela na platformi. Ukoliko je potrebno moguće je realizovati i sopstveni modul za crtanje (engl. paint engine) i na taj način po potrebi nezavisno iscrtavati svaku od komponenti GKS-e. Modul za rukovanje sistemom prozora zadužen je i za prosleđivanje događaja generisanih od strane platforme do Qt radnog okvira. QPA sloj omogućuje i stilizovanje komponenti grafičke korisničke sprege, npr. na Android platformama korišćenje internih tema kao podrazumevanih.

QPlatformIntegration	QPlatformTheme
QPlatformWindow	QPlatformMenu
QPlatformBackingStore	QPlatformMenuBar
QPlatformOpenGLContext	QPlatformDialogHelper
QPlatformSharedGraphicsCache	platform palettes
QPlatformFontDatabase	platform fonts
QPlatformClipboard	theme hints
QPlatformDrag	
QAbstractEventDispatcher	
QPlatformInputContext	
QPlatformAccessibility	
QPlatformNativeInterface	
QPlatformServices	

Slika 4. Pregled klasa QPA sloja

2.2.2 Qt medija plejer

Qt radni okvir koristi poseban modul - Qt medija plejer, kako bi reprodukovao audio i video datoteke unutar aplikacije. Sastoji se iz apstraktnih klasa čijom se realizacijom formira veza sa konkretnom bibliotekom koja vrši reprodukciju audia, odnosno videa. Putem ovog modula moguće je radnom okviru proslediti i meta informacije o datotekama koje se reprodukuju. Qt medija plejer se sastoji iz više različitih kontrola koje realizuju njegovu funkcionalnost. Analogno QPA sloju realizacija Qt medija plejer modula je zavisna od platforme na kojoj se radni okvir izvršava, npr. Linux verzija Qt radnog okvira koristi “dži-strimer” (engl. GStreamer) biblioteku.

3. Analiza zahteva i koncept rešenja

U cilju kreiranja koncepta rešenja na početku ovog poglavlja izvršena je analiza zahteva. Nakon izvršene analize, definisani su neophodni gradivni blokovi kao i pravila njihove međusobne interakcije. Struktura rešenja podeljena je u tri celine: RVU poslužitelj, Qt dodaci i Qt radni okvir.

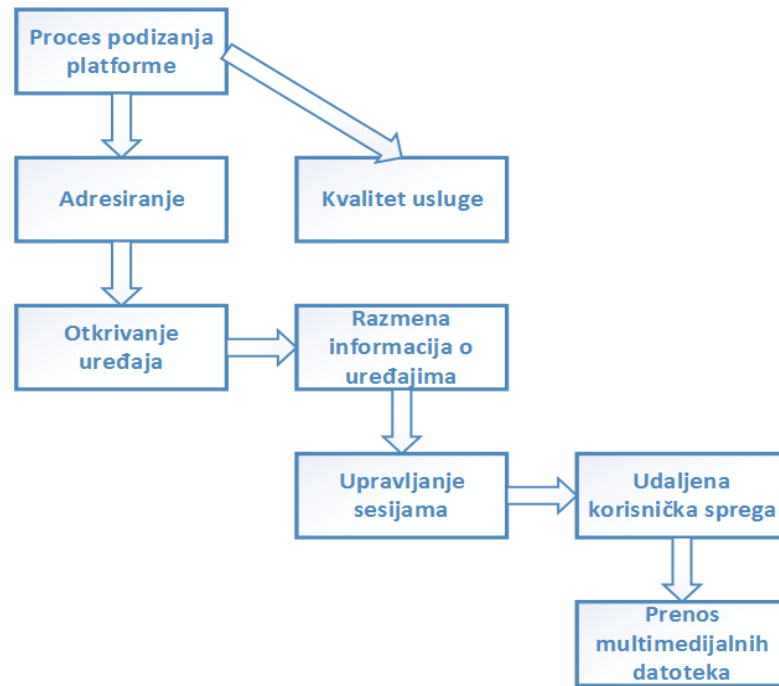
3.1 RVU poslužitelj

RVU poslužitelj detaljno je definisan RVU specifikacijom, u kojoj su navedene njegove obavezne i opcione funkcionalnosti. U sledećih nekoliko poglavlja opisani su zahtevi koje je potrebno ispuniti, kao i način izvedbe rešenja. Neki od zahteva nisu detaljno specificirani već postoji predlog mogućeg rešenja. Ukoliko je od predloga odstupljeno definisan je razlog zbog kojeg je to urađeno. Kako bi se razumelo funkcionisanje sistema u celini, tj. interakcija između RVU poslužitelja i RVU klijenata, u poglavljima u kojima je bilo potrebno opisana je i struktura RVU klijenta. Realizovani koncept rešenja je u potpunosti saglasan sa specifikacijom, kako bi se omogućila sertifikacija uređaja.

RVU protokol može se podeliti u šest podprotokola:

- Adresiranje, pronalaženje i opis RVU uređaja
- Upravljanje sesijama
- Udaljena korisnička sprega
- Prenos multimedijalnih datoteka
- Kvalitet usluge i dijagnostika (engl. QoS and diagnostic)
- Pribavljanje datoteka (statičkih, deljenih, izvršnih)

Na sledećoj slici je prikazana sekvenca neophodnih protokola po pokretanju platforme:



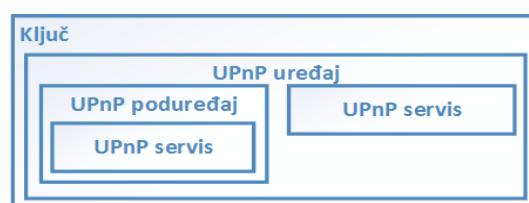
Slika 5. Prikaz sekvenci neophodnih protokola po pokretanju platforme

3.1.1 Adresiranje, pronalaženje i opis RVU uređaja

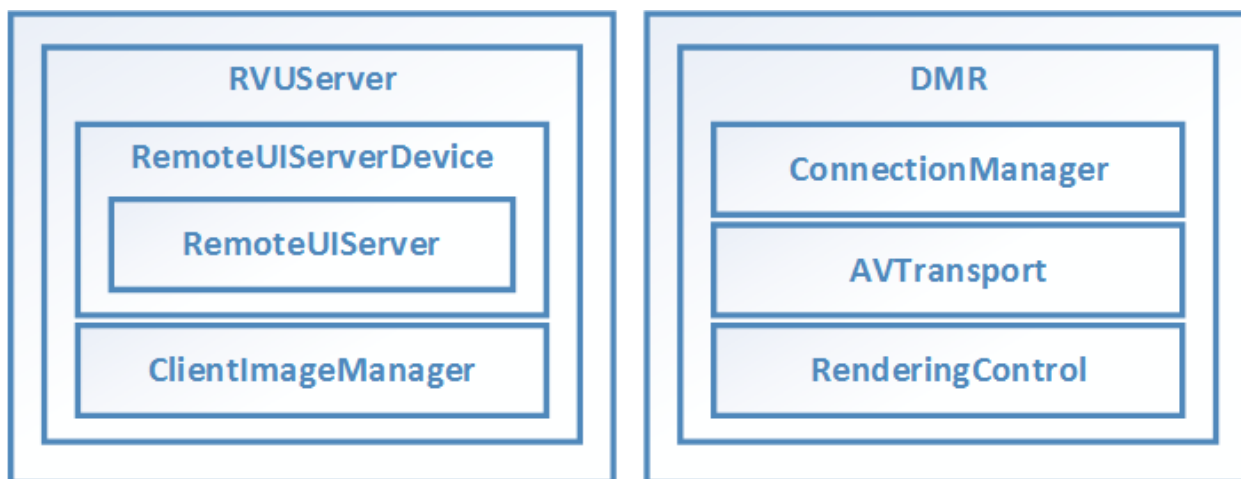
Adresiranje, pronalaženje i opis RVU uređaja zasniva se na UPnP protokolu, odnosno UPnP arhitekturi uređaja [9] i DLNA direktivama [8].

- Adresiranje se odnosi na dodeljivanje IP adresa svakom od čvorova unutar kućne mreže.
- Pronalaženje predstavlja inicijalni proces interakcije između UPnP uređaja i kontrolnih tačaka u cilju dobavljanja informacija o raspoloživosti uređaja i servisa koje ti uređaji pružaju.
- Opis RVU uređaja omogućuje kontrolnim tačkama pribavljanje detaljnih informacija o dostupnim uređajima i njihovim servisima. Format opisa definisan je XML šema dokumentima.

RVU protokol ne propisuje tačnu strukturu RVU poslužitelja i RVU klijenta već samo njihove neophodne elemente. Na slikama koje slede je prikaz ključa i struktura predloženih RVU UPnP uređaja.



Slika 6. Ključ za razumevanje strukture UPnP uređaja i servisa



Slika 7. Strukture RVU UPnP poslužitelja i klijenta

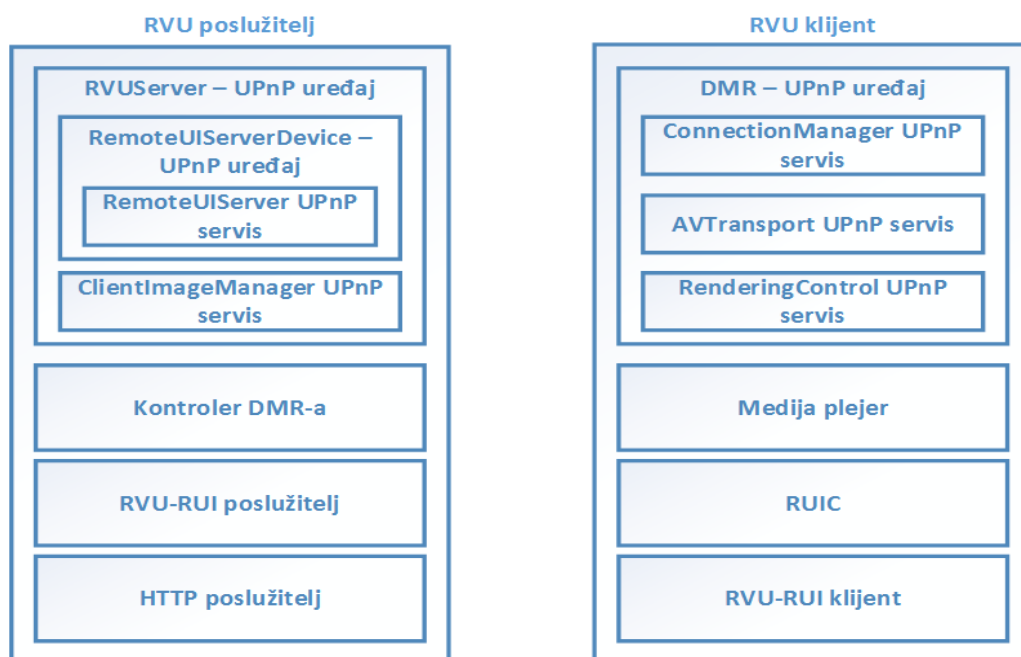
RVU UPnP poslužitelj i klijent su UPnP uređaji definisani od strane RVU protokola uz pomoć kojih se RVU poslužitelj i klijent reklamiraju na mreži. Struktura RVU poslužitelja i klijenta je znatno složenija i oni se sastoje od:

RVU poslužitelj:

- RVU UPnP korenskog uređaja
- Ugrađenog RUI poslužitelja
- RVU-RUI poslužitelja
- "ClientImageManager" servisa
- Kontroler DMR-a
- HTTP poslužitelja

RVU klijent:

- DMR-a
- Medija plejera
- RUIC-a
- RVU-RUI klijenta



Slika 8. Struktura RVU poslužitelja i klijenta

3.1.1.1 Gradivne komponente RVU poslužitelja

RVU UPnP poslužitelj je korenski UPnP uređaj. Njegova glavna funkcionalnost je agregacija ostalih UPnP uređaja i servisa uz pomoć kojih se RVU poslužitelj reklamira na mreži, kao što je ranije navedeno.

Ugrađeni RUI poslužitelj (UPnP uređaj - RemoteUIServerDevice) je poduređaj RVU UPnP poslužitelja koji omogućuje deljenje RVU-RUI GKS-e putem "RemoteUIServer" UPnP servisa. Pored RVU-RUI-ja moguće je deliti i druge korisničke sprege zasnovane na nekom drugom protokolu, kao što su HTML, HTML5, CE-HTML, međutim one nisu razmatrane u ovom radu.

RVU-RUI poslužitelj omogućuje distribuciju i upravljanje komponentama GKS-e. Funkcionalnost ovog modula u potpunosti je definisana RVU specifikacijom i detaljno opisana u poglavlju udaljene korisničke sprege.

"ClientImageManager" je opcioni UPnP servis i omogućuje dobavljanje datoteka. Ovaj tip servisa je nov i definisan je od strane RVU alijanse. Putem servisa vrši se razmena informacija o verzijama datoteka, dok se prenos datoteka vrši pomoću TFTP protokola. Važno je naglasiti da CIA protokol nije predviđen za prenos datoteka koje nisu povezane sa RVU protokolom, već je predstavlja funkcionalnost putem koje se ažurira RVU programski stek.

HTTP poslužitelj omogućuje distribuciju sadržaja do klijenta. Ovaj poslužitelj mora podržavati DLNA proširenja zaglavlja HTTP GET zahteva. Ova proširenja omogućuju upravljanje sesijom isporuke podataka.

Kontroler DMR (engl. push controller), odnosno njegovih servisa, koristi se kako bi se omogućila kontrola reprodukcije multimedijalnih datoteka. Ovaj kontroler čine kontrolne tačke sledećih servisa koji su dodatno prošireni akcijama i varijablama stanja od strane RVU protokola u cilju bolje kontrole reprodukcije:

- "ConnectionManager" - UPnP servis omogućuje modelovanje distributivnih sposobnosti (engl. streaming capabilities) audio i video uređaja, kao i njihovo povezivanje ukoliko se sposobnosti poklapaju, npr. izvor (DMS) omogućuje distribuciju H264 enkodovanog videa, a odredište (DMR) ima podršku za reprodukciju istog tog video formata. Servis takođe pruža informacije o nekoj od trenutnih sesija, kao i listu svih aktivnih sesija. Kako bi se RVU poslužitelju omogućila apsolutna kontrola nad uspostavljanjem veza sa RVU klijentima uvode se dve nove obavezne akcije:
 - UPnP akcija "PrepareForConnection" - Priprema RVU klijenta u cilju uspostavljanja veze sa poslužiteljem. Razmenjuju se informacije o mreži, protokolu kojim će biti vršen prenos podataka, kao i formatu podataka.

- UPnP akcija "ConnectionComplete" - Nakon završetka prenosa podataka RVU poslužitelj koristi ovu akciju kako bi oslobodio resurse koji su bili vezani za sesiju.
- "RenderingControl" - UPnP servis omogućuje dinamičko podešavanje različitih atributa (jačina tona, osvetljenost ekrana, kontrast ekrana, itd.) uređaja za reprodukciju multimedijalnih datoteka. Koristeći kontrolnu tačku ovog servisa moguće je dobiti listu parametara čiju konfiguraciju je moguće izvršiti, kao i očitati njihovo trenutno stanje. Kako bi se korisniku olakšalo podešavanje uređaja moguće je formirati listu predefinisanih podešavanja (engl. preset). Jedan od glavnih zadataka RVU poslužitelja je distribucija DTV servisa. Sadržaj koji se prenosi je enkodovan u okviru transportnog toka (engl. transport stream – TS) u kojem se može nalaziti jedan ili više audio i video elementarnih tokova, kao i drugih tipova podataka poput teleteksta, EPG-a. Unutar transportnog toka svaki od paketa ima svoj identifikator paketa - PID, uz pomoć kojeg je moguće razlikovati podatke. RVU protokol uvodi posebne akcije i promenljive stanja (proširenja servisa) kako bi omogućio rukovanje različitim video i audio tokovima unutar transportnog toka.

Promenljive stanja:

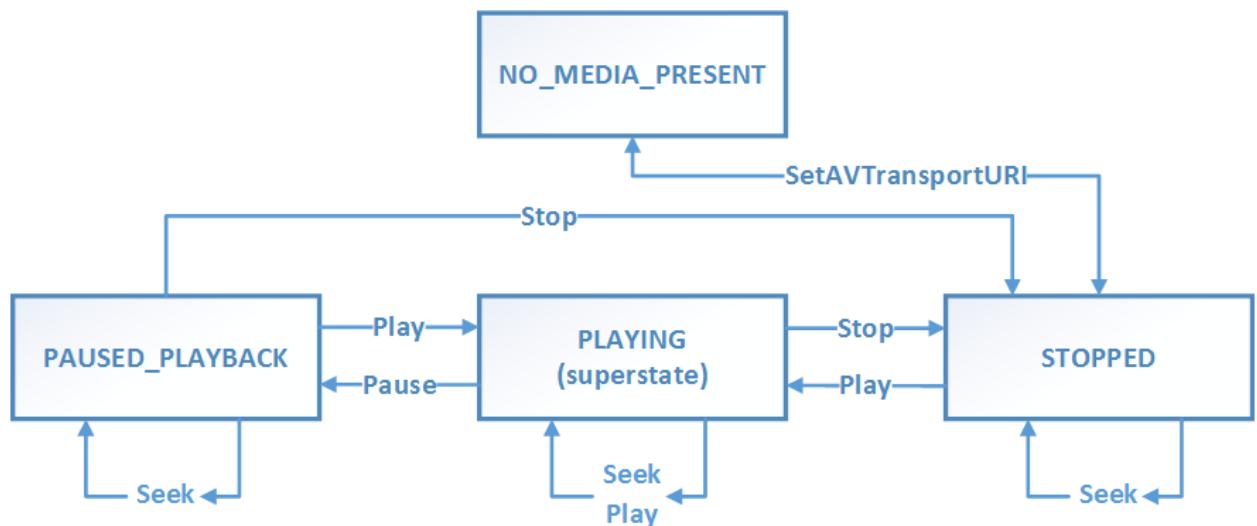
- X_AudioPID - audio PID
- X_AudioEncoding - format audia koji odgovara X_AudioPID-u (aac, ac3, pcm, mp3, ...)
- X_VideoPID - video PID
- X_VideoEncoding - format videa koji odgovara X_VideoPID-u (h264, mpeg2, ...)

Akcije:

- X_UpdateAudioSelection – podešavanje audio PID-a elementarnog toka koji se želi reprodukovati
- X_GetAudioSelection – dobavljanje trenutnog audio PID-a
- X_UpdateVideoSelection – podešavanje video PID-a elementarnog toka koji se želi reprodukovati
- X_GetVideoSelection – dobavljanje trenutnog video PID-a
- "AVTransport" - UPnP servis omogućuje kontrolu distribucije i reprodukcije multimedijalnog sadržaja. Stanja DMR modula takođe su propisana specifikacijom AVTransport servisa, pri čemu "AVTransport" kontrolna tačka akcijama ovog servisa prevodi DMR iz jednog stanja u drugo. RVU protokol ograničava svoja stanja i prelaze prema slici 8.

Stanje "PLAYING" moguće je dekomponovati na podstanja u zavisnosti od brzine reprodukcije sadržaja. Veća sloboda prilikom upravljanja reprodukcijom sadržaja omogućena je uvođenjem novih prelaza između stanja od strane RVU protokola:

- proizvoljno pozicioniranje (engl. seek) u stanju "PAUSED_PLAYBACK"
- proizvoljno pozicioniranje u stanju "PLAYING"
- proizvoljno pozicioniranje u stanju "STOPPED"



Slika 9. Dijagram stanja DMR RVU klijenta

3.1.1.2 Gradivne komponente RVU klijenta

U ovom poglavlju opisane su gradivne komponente RVU klijenta sa slike 8.

DMR se po DLNA specifikaciji sastoji od "ConnectionManager", "RenderingControl" i "AVTransport" UPnP servisa, čije kontrolne tačke čine prethodno opisani kontroler DMR-a na RVU poslužitelju.

Medija plejer je spregnut sa DMR-om i ujedno predstavlja njegovog korisničkog agenta. Korišćenjem servisa DMR-a vrši se kontrola medija plejera, koji takođe predstavlja i HTTP klijenta jer vrši reprodukciju sadržaja dostupnog putem mreže.

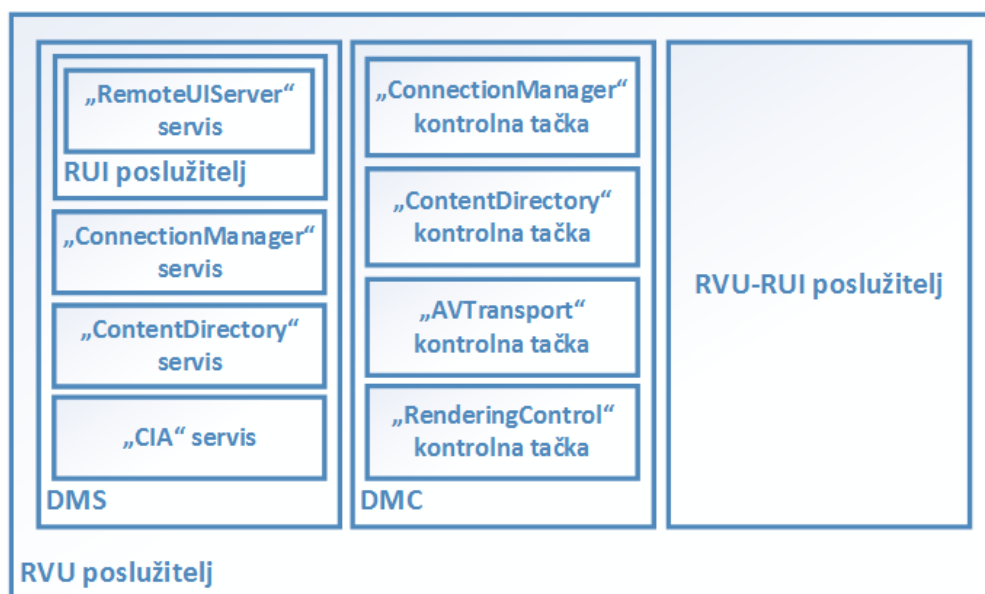
RUIC omogućuje pronalaženje RUISrc-a i RUISink-a i njihovo uparivanje, kao i odabir adekvatne GKS-e. U ovom rešenju RUIC vrši pretragu RUISrc kako bi pronašao RVU-RUI GKS-u nakon čega može da se poveže na RVU-RUI poslužitelja.

RVU-RUI klijent omogućuje spregu, konfigurisanje i upravljanje modulom za crtanje na klijentskoj strani (SDL, OpenGL, itd.). Konfigurisanje podrazumeva obaveštavanje poslužitelja o podržanim rezolucijama iscertavanja i dekodovanja, vrstama generisanih događaja, kao i grafičkim operacijama koje je moguće koristiti u cilju formiranja grafičke korisničke sprege. Pozicioniranje prozora u kojem se iscertava reprodukovani video je takođe jedan od zadataka

ovog modula. Pored toga, zadužen je i za akviziciju događaja generisanih od strane klijenta i njihovu propagaciju do RVU poslužitelja.

3.1.1.3 Koncept rešenja strukture RVU poslužitelja

Specificirani model RVU poslužitelja omogućuje kontrolu DMR-a na RVU klijentu. Na ovaj način stvara se doživljaj da ugrađeni kontroler DMR-a "gura" sadržaj od ugrađenog HTTP poslužitelja do RVU klijenta i upravlja njegovom reprodukcijom. U teorijskim osnovama ovaj tip interakcije između RVU poslužitelja i klijenta definisan je kao model dvostruke kutije. Ovaj model zamenjen je modelom trostruke kutije. Zamena je sprovedena ugradnjom kontrolne tačke "ContentDirectory" UPnP servisa u kontroler DMR-a, odnosno kontroler DMR-a zamenjen je DMC-om, a korenski RVU UPnP uređaj zamenjen DMS-om koji u sebi sadrži HTTP poslužitelja. Na ovaj način predložena struktura RVU poslužitelja je unapređena kako bi se omogućila razmena sadržaja i sa uređajima koji ne realizuju RVU klijenta, odnosno omogućena je razmena sadržaja sa uređajima koji imaju sposobnost interakcije sa DMS-om. Interakcija između DMC-a, DMS-a i DMR-a predstavlja model trostruke kutije. Slika 9. opisuje usvojenu strukturu RVU poslužitelja.



Slika 10. Usvojena struktura RVU poslužitelja

Ugrađeni DMS u sebi pored svojih obaveznih servisa sadrži sve obavezne servise RVU poslužitelja, a takođe i ugrađeni RUIS UPnP uređaj. DMS je proširen i mogućnošću deljenja sadržaja od strane spoljašnjih poslužitelja, npr. putem spoljašnjeg RTSP poslužitelja koji omogućuje distribuciju DTV servisa. RTSP poslužitelj prikazan je putem ranije spomenutog objekta frekvencijskog odabirača. Ovaj objekat u sebi sadrži opise DTV servisa koje je moguće dobiti putem RTSP poslužitelja.

Servisi DMS-a:

- "ConnectionManager" servis koji ima recipročnu ulogu u odnosu na onu kod DMR-a. U samom servisu realizovane su sve potrebne akcije i promenljive stanja propisane kao obavezne po DLNA i RVU specifikaciji.
- "ContentDirectory" servis omogućuje deljenje i pretragu datoteka. Dodavanjem ovog servisa omogućena je prethodno pomenuta proširena funkcionalnost RVU poslužitelja.
- "ClientImageAcquisition" servis je preuzet iz RVU poslužitelja kako bi se realizovao korenski UPnP uređaj sa svim potrebnim servisima prema RVU specifikaciji. Više informacija o ovom servisu će biti u poglavlju namenjenom opisu CIA podprotokola. Važno je napomenuti da je ovo opcioni servis i da je ovde naveden u cilju sagledavanja kompletnog rešenja, premda je njegova programska realizacija izostavljena.

Ovako formirani hibridni UPnP uređaj predstavlja uniju servisa DMS-a i RVU poslužitelja. Sadržaj koji RVU poslužitelj stavlja na raspolaganje putem DMS-a predstavljen je preko DLNA objekata, koji sadrže informacije o datotekama. Ukoliko je sadržaj zaštićen DTCP-IP protokolom DLNA objekti sadrže IP adresu i port DTCP-IP poslužitelja, što važi i za objekte frekvencijskih odabirača. Klijent mora da se poveže na DTCP-IP poslužitelja i izvrši autentikaciju kako bi mogao da prihvati i dekriptuje primljeni sadržaj.

DMC sadrži kontrolne tačke sledećih UPnP servisa koji realizuju sve akcije i promenljive stanja obavezne po DLNA i RVU specifikaciji:

- "ConnectionManager"
- "ContentDirectory"
- "AVTransport"
- "RenderingControl"

Prethodno spomenuti RUI poslužitelj koji po specifikaciji predstavlja ugrađeni UPnP uređaj RVU poslužitelja, je ugrađen u DMS. RUI poslužitelj je realizovan po specifikaciji i uz pomoć njega se RVU klijent spreže sa RVU-RUI poslužiteljem. Ovaj modul kao što se da videti sa prethodno priložene slike takođe predstavlja gradivnu komponentu RVU poslužitelja, a njegova struktura detaljno će biti izložena u analizi RVU-RUI podprotokola. Uvođenjem RUI-ja DLNA model "trostruke kutije" proširen je na DLNA model "trostruke kutije sa GKS-om" (engl. 3-box UI with A/V system usage).

3.1.2 Upravljanje sesijama

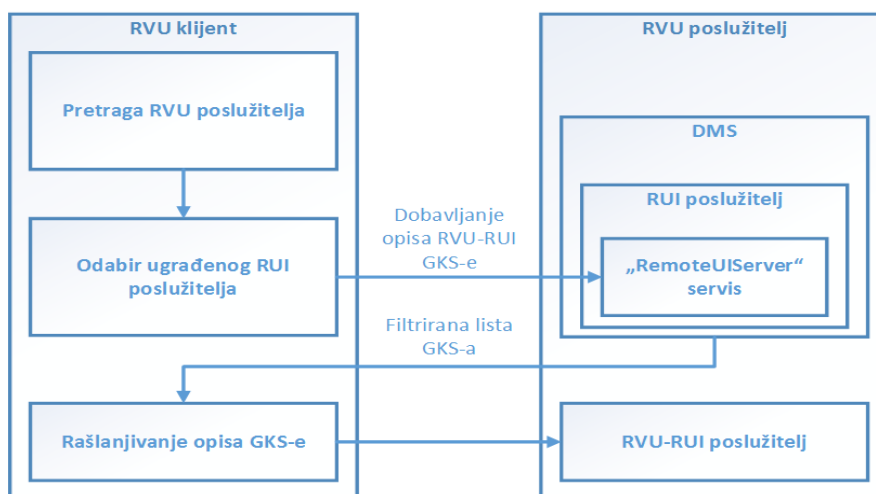
U prethodno opisanoj fazi adresiranja i pretraživanja RVU klijent je dobio informacije o UPnP uređajima koji su dostupni u kućnoj mreži. Nakon ove faze klijent filtrira RUI poslužitelje među dostupnim UPnP uređajima (RUI poslužitelj je ugrađen u DMS) i omogućuje klijentu izbor željenog RUI poslužitelja. Na odabranom poslužitelju pronalazi RVU-RUI korisničku spregu i uspostavlja vezu sa RVU-RUI poslužiteljem. Kada je ova veza uspostavljena, može da prepozna DMR klijenta putem IP adrese i poveže se sa njim, čime je faza uspostavljanja RVU sesije završena. Kompletan proces uspostavljanja veze je definisan UPnP standardom, dok je opis RVU-RUI-ja i identifikacija između RVU-RUI poslužitelja i DMR-a definisana RVU protokolom.

Usvojeni koncept rešenja ovog podprotokola je u potpunosti u skladu sa UPnP, DLNA i RVU specifikacijama. DMC RVU poslužitelja se pretplaćuje (engl. subscribe) prilikom uspostave veze na sve servise DMR-a RVU klijenta, čime se u toku sesije omogućuje trenutna kontrola servisa bez potrebe za naknadnom pretplatom.

Opis RVU-RUI GKS-e koji omogućuje povezivanje sa RVU-RUI poslužiteljem:

```
<name>RVU Remote UI</name>
<protocol shortName="RVU-RUI">
    <uri>rvurui://[ip]:[port]</uri>
</protocol>
```

- name - deskriptivno ime GKS-e
- shortName - identifikator protokola
- uri - adresa i prolaz RVU-RUI poslužitelja na kojem prima zahteve RVU-RUI klijenata



Slika 11. Uspostavljanje RVU-RUI sesije

3.1.3 Udaljena grafička korisnička sprega

RVU-RUI podprotokol uveden je u poglavlju teorijskih osnova, gde je dat i njegov kratak pregled. U ovom poglavlju biće izvršena detaljna analiza u cilju formiranja RVU-RUI poslužitelja. Putem RVU-RUI protokola omogućeno je slanje događaja (engl. events) od klijenata do poslužitelja, kao i grafičkih i audio komponenti u suprotnom smeru u cilju formiranja GKS-e.

3.1.3.1 TCP tokovi - kanali

RVU klijent započinje RVU-RUI sesiju tako što otvara prvi komandni kanal, na IP adresi i prolazu (engl. port) koje je dobio od ugrađenog RUI poslužitelja (IP adresa i prolaz se nalaze u opisu RVU-RUI GKS-e). Nakon otvaranja prvog kanala po potrebi može biti otvoren proizvoljan broj dodatnih kanala između RVU-RUI poslužitelja i klijenta. Dodatni kanali mogu biti otvoreni od strane klijenta ili poslužitelja. Iz ovoga sledi da svi RVU uređaji moraju u sebi imati ugrađenog TCP poslužitelja koji će primati zahteve za otvaranje kanala, odnosno TCP klijenta koji će slati zahteve.

S obzirom na vrstu podataka koja se prenosi putem kanala oni se mogu podeliti na komandne i kanale za podatke. U zavisnosti od prvog primljenog bajta na kanalu zavisi njegov tip. Kada se odredi tip kanala dodeljuje mu se jedinstveni identifikator i za vreme njegovog životnog veka preko njega se mogu prenositi podaci samo tog tipa. U svakom vremenskom trenutku RVU uređaji mogu zatvoriti kanal. Kada se zatvori poslednji komandni kanal, završava se i RVU-RUI sesija.

Vrste kanala:

- Komandni kanal - Preko komandnog kanala se prenose XML nizovi znakova (engl. string) (UTF-8 enkodovane XML poruke). Prvi bajt koji mora biti primljen na komandnom kanalu je karakter '<'. Na svaku primljenu komandu RVU uređaji šalju odgovore, pri čemu je format odgovora zavisao od tipa komande. Obavezno polje svakog odgovora je indikacija uspešnosti izvršavanja. Prva poruka koja se razmeni preko svakog komandnog kanala je "Helo" poruka, koja sa sobom nosi informaciju o prolazu ugrađenog TCP poslužitelja na kojem suprotna strana može po potrebi zatražiti otvaranje dodatnog kanala. Komande se na kanalu obrađuju sekvencijalno u redosledu pristizanja, a nepotpune komande se odbacuju. Ukoliko je potrebno moguće je obustaviti izvršavanje neke komande, što se može ostvariti upotrebom drugog komandnog kanala. Kako bi se komande međusobno razlikovale

svaka od njih pored imena i neophodnih argumenata sadrži i identifikator koji mora biti jedinstven u definisanom vremenskom intervalu.

Format komande:

```
<commandName commandToken="commandTokenValue"
```

```
attributeName1="attributeValue1" ... attributeNameN="attributeValueN"/>
```

- commandName - ime komande (lista svih dostupnih RVU-RUI komandi je data na kraju ovog poglavlja)
- commandToken - identifikator komande
- attributeN - atribut komande

Format odgovora na komandu:

```
<Response errCode="errorCodeValue" commandToken="commandTokenValue"
```

```
attributeName1="attributeValue1" ..attributeNameN="attributeValueN"/>
```

- errCode - indikacija greške (svaki podržani tip komande ima listu mogućih vrednosti greške)
- commandToken - identifikator komande (na osnovu ovog identifikatora vrši se identifikacija komande za koju je stigao odgovor)
- attributeN - atribut odgovora

• Kanal za podatke - Kanalima za podatke se prenose binarni ili ASCII podaci. Prvi bajt na ovom tipu kanala je ASCII decimalna cifra od 0 do 9. Analogno komandom kanalu prva poruka koja se razmeni je "Helo data" poruka, nakon čega je moguće slati ostale poruke koje sadrže podatke. Ove poruke povezuje se sa odgovarajućom komandom putem identifikatora poruke (engl. commandToken), odnosno one imaju isti identifikator poruke. Format poruke je prikazan na slici koja sledi.

Syntax	Bits	Format	Comment
RVU-RUI-Data-Frame() {			
frame-header {			
frame-header-len	32	ASCII-coded decimal	4-digits, 0 padded
separator1	8	0x20	
frame-type-id	8*number of chars	ASCII characters	
separator2	8	0x20	
command-token	8*number of digits	ASCII-coded decimal	
separator3	8	0x20	
frame-data-len	8*number of digits	ASCII-coded decimal	
CRLF1	16	0x0D0A	
}			
frame-data	8*frame-data-len	uimbsbf	
CRLF2	16	0x0D0A	
}			

Slika 12. Format poruke sa podacima

Podržani tipovi podataka:

- ASCII cifre - podskup ASCII znakova
- PixelData - nekompresovana slika
- CompressedPixelData - slika enkodovana u “zlib” format
- JPEGPixelData - slika enkodovana u JPEG format
- PNGPixelData - slika enkodovana u PNG format
- CLUT - binarni CLUT podaci (CLUT tabela)
- AudioData - audio podaci

Iz prethodno izlaganja proizlazi da je optimalan broj komandnih kanala tri. Prvi komandni kanal se koristi za sekvencijalni prenos komandi, drugi služi kako bi se asinhrono mogao promeniti redosled izvršavanja komandi, dok je treći enkriptovan i služi za prenos događaja. Sa druge strane optimalan broj kanala za podatke zavisi od količine podataka koje je potrebno preneti.

3.1.3.2 Vremenska ograničenja

Komande i podaci se prenose različitim kanalima, usled čega oni dolaze na odredište u različitim vremenskim trenucima. Kako bi se izbegla velika vremenska kašnjenja i nagomilavanja komandi odnosno podataka, RVU specifikacija je definisala vremenske okvire u kojima se navodi koliko podaci mogu da kasne za komandom, ili obrnuto. Preporuka specifikacije koja je usvojena u ovom rešenju, je da se uvek prvo šalje komanda nakon čega se šalju njoj pridruženi podaci. Vremenska kašnjenja regulisana su od strane RVU-RUI klijenta, dok RVU-RUI poslužitelj prati indikacije o vremenskim kašnjenjima putem odgovora na komande koje mu šalje klijent i na taj način po potrebi može ponoviti slanje nekih podataka ukoliko je došlo do njihovih gubitaka.

3.1.3.3 RVU-RUI tipovi bafera

Na klijentskoj strani podaci se čuvaju u baferima. U zavisnosti od vrste podataka koja se čuva u baferu zavisi i njegov tip. RVU-RUI poslužitelj po potrebi zauzima i oslobađa bafere na klijentu. Prilikom uspostavljanja RVU-RUI sesije klijent poslužitelju prosleđuje informacije o količini dostupne memorije, kao i broju bafera koji se mogu zauzeti na klijentu. U ovom rešenju omogućeno je rukovanje svim tipovima bafera definisanim RVU specifikacijom.

Tipovi bafera:

- grafički bafer (engl. “off-screen” buffer) - omogućuje skladištenje komponenti GKS-e
- bafer za prikaz (engl. display buffer) - bafer čiji se sadržaj prikazuje korisniku
- video bafer - logički bafer koji služi za podešavanje video prozora

- audio bafer - omogućuje skladištenje audio podataka

3.1.3.4 RVU-RUI komande

RVU-RUI komande se mogu podeliti u grupe prema funkcionalnosti koju realizuju. Putem ovih komandi omogućeno je otvaranje i zatvaranje kanala, kao i dobavljanje informacija o memorijskoj slici klijenta. Poseban podskup komandi zadužen je za rukovanje GKS-om. Komande za prikaz upravljaju pozicioniranjem video prozora i prozora za prikaz. Po potrebi klijent može preuzeti upravljanje iscrtavanjem GKS-e od poslužitelja slanjem sopstvenog zahteva ili na osnovu zahteva poslužitelja. RVU klijenti prilikom prikaza GKS-e mogu reprodukovati različite audio efekte, zahvaljujući posebnom podskupu audio komandi.

U procesu inicijalizacije RVU-RUI poslužitelj dobavlja informacije o događajima podržanim od strane klijenta. Generisani događaji se u određenom formatu (HDMI ili CDI) prenose komandnim kanalom do poslužitelja. Oni mogu da sadrže poverljive informacije poput šifre korisnika. Kako bi se omogućila zaštita ovih podataka RVU specifikacija definiše opcionu podršku TLS (Transport Layer Security) protokola. Ukoliko RVU poslužitelj podržava ovaj protokol događaji generisani od strane korisnika će biti enkriptovani prilikom prenosa.

U sledećih nekoliko tabela dat je pregled komandi po grupama koje je potrebno realizovati:

Ime komande	Opis	Tip podataka
Hello	Informacije neophodne za uspostavljanje kanala	-
Shutdown	Zatvaranje kanala	-
GetMemInfo	Dobavljanje informacija o memorijskim resursima	-

Tabela 1. Komande za uspostavu, raskid i informacije o RVU-RUI sesiji

Ime komande	Opis	Tip podataka
HDMIKeyEvent	Događaj opisan HDMI formatom (obavezan format)	-
CDIKeyEvent	Događaj opisan CDI formatom (opciona format)	-
GetKeyList	Dobavljanje informacija o listi podržanih događaja	-

Tabela 2. Komande za opis korisničkih događaja

Ime komande	Opis	Tip podataka
AllocateBuffer	Zauzimanje grafičkog bafera	-
DeallocateBuffer	Oslobađanje grafičkog bafera	-
Write	Upis podataka u grafički bafer	GraphicsData
Read	Čitanje podataka iz grafičkog bafera	GraphicsData
BlitQueue	Komanda za grupisanje komandi spajanja	-
Dispatch	Izvršavanje akumuliranih komandi spajanja	-
EmptyQueue	Brisanje akumuliranih komandi	-

WaitVSync	Čekanje da klijent uđe u Vsync period	-
CopyBlit	Kopiranje piksel podataka	-
FillBlit	Popunjavanje regiona zadatom bojom	-
ResizeBlit	Skalirano kopiranje piksel podataka	-
ShadeBlit	Porter-Duff stapanje konstantom bojom	-
BlendBlit	Porter-Duff stapanje dve grupe piksel podataka	-
MultiSourceBlendBlit	Porter-Duff stapanje dva grafička bafera	-
ResizeAndBlendBlit	Skalirano Porter-Duff stapanje dve grupe piksel podataka	-
ColorKeyResizeBlit	Skalirano kopiranje piksel podataka uz poređenje sa bojom	-
GetGraphicsCaps	Informacija o podržanim grafičkim operacijama i parametrima	-
GetZList	Dobavljanje rasporeda bafera po Z osi	-
SetZList	Postavljanje rasporeda bafera po Z osi	-
SetCLUT	Upis CLUT tabele u grafički bafer	CLUT

Tabela 3. Grafičke komande

Ime komande	Opis	Tip podataka
ListLocalUIElements	Lista elemenata GKS-e dostupnih na klijentu	-
RequestLocalUI	Zahtev za generisanje lokalne GKS-e	-
LocalUIEvent	Ulaz ili izlaz iz lokalne grafičke operacije	-
ClientRequestLocalUI	Zahtev klijenta za preuzimanjem kontrole nad GKS-om	-

Tabela 4. Komande za opis lokalne GKS-e

Ime komande	Opis	Tip podataka
GetVideoBuffer	Povezivanje AVTransport sesije sa video baferom	-
ReleaseVideoBuffer	Otpuštanje kontrole nad video baferom	-
SetBackgroundColor	Postavljanje pozadinske boje	-
ConfigureDisplayBuffer	Konfigurisanje bafera za prikaz	-
ReconfigureDisplayBuffer	Rekonfigurisanje bafera za prikaz	-
ConfigureVideoFullscreen	Konfigurisanje video bafera preko celog ekrana	-
ConfigureVideoWindow	Konfigurisanje video bafera za prikaz unutar GKS-e	-
ConfigureWindowedVideoWindow	Konfigurisanje dela video bafera za prikaz unutar GKS-e	-
ConfigureVideoDecoderResolution	Konfigurisanje rezolucije za dekodovanje videa	-
BlindVideo	Prikrivanje video bafera	-
GetOutputSettings	Dobavljanje rezolucije ekrana	-
OutputSettingsChanged	Promena rezolucije ekrana	-
GetVideoDisplaySettings	Dobavljanje podešavanja video bafera	-
VideoDisplaySettingsChanged	Promena podešavanja video bafera	-
AllowedClosedCaptioning	Dozvola prikaza prevoda	-

EnableClosedCaptioning	Uključivanje/Isključivanje prevoda	-
GetCloseCaptioningState	Dobavljanje podešavanja za prevode	-
ReconfigureDisplayBuffer3DTV	Rekonfigurisanje bafera u 3DTV modu	-
Set3DTVFlattenStructure	Podešavanje 3DTV parametara	-

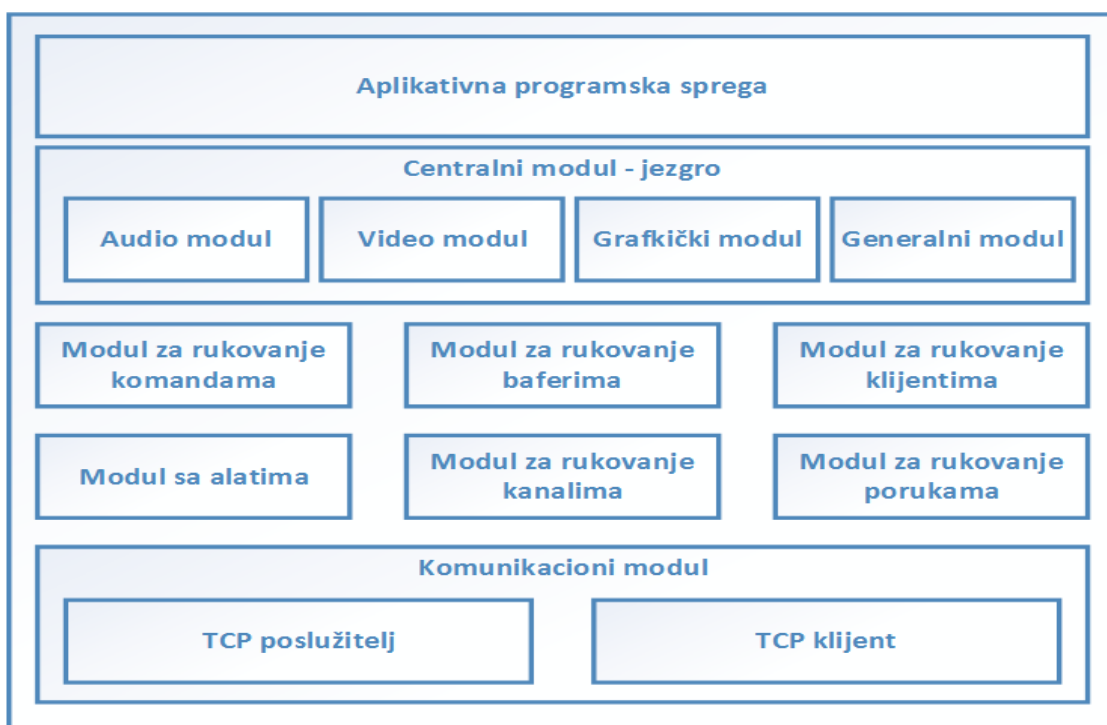
Tabela 5. Komande za prikaz

Ime komande	Opis	Tip podataka
OpenAudioDecoder	Otvaranje audio dekodera	-
CloseAudioDecoder	Zatvaranje audio dekodera	-
GetNumAudioDecoders	Dobavljanje broja audio dekodera	-
GetAudioDecoderCaps	Dobavljanje informacija o audio dekoderima	-
AllocateAudioBuffer	Zauzimanje audio bafera	-
DeallocateAudioBuffer	Oslobađanje audio bafera	-
WriteAudioData	Upis audio podataka u zadati audio bafer	AudioData
Play	Reprodukcija audio podataka dostupnih na kanalu za podatke	AudioData
PlayBuffer	Reprodukcija audio podataka dostupnih u audio baferu	-
PlayStatus	Status reprodukcije audio podataka	-
Stop	Zaustavljanje reprodukcije audio podataka	-

Tabela 6. Audio komande

3.1.3.5 Struktura RVU-RUI poslužitelja

Koncept rešenja strukture RVU-RUI poslužitelja je prikazana na sledećoj slici:



Slika 13. Struktura RVU-RUI poslužitelja organizovana kao programski stek

Na dnu programskog steka nalazi se komunikacioni modul koji omogućuje uspostavu kanala između RVU-RUI poslužitelja i klijenta. Ovaj komunikacioni modul se sastoji iz TCP poslužitelja koji će prihvatati zahteve i TCP klijenta koji će slati zahteve za otvaranje kanala RVU-RUI klijentu. Modul sa alatima sadrži realizaciju neophodnih struktura podataka koje će se koristiti u ovom rešenju. Kako bi se eliminisao uticaj konkurentnog pristupa strukturama potrebno je obezbediti sinhronizaciju pristupa (engl. “thread-safe”). Strukture koje su identifikovane kao neophodne u ovom rešenju su lista i red.

Modul za rukovanje kanalima je uveden kako bi se omogućilo upravljanje TCP tokovima, dok modul za rukovanje porukama omogućuje kreiranje, slanje i uništavanje RVU-RUI poruka (komande, podaci). Ova dva bloka su međusobno spregnuta čime se omogućuje slanje poruke preko željenog kanala.

Na sledećem nivou apstrakcije se nalazi modul za rukovanje klijentima. Putem njega omogućeno je kreiranje novog klijenta i brisanje postojećeg. Sve informacije o RVU-RUI klijentu se čuvaju u ovom modulu. Informacije su podeljene na dva nivoa apstrakcije. Veza između ova dva nivoa ostvarena je putem identifikatora klijenta, koji je isti na oba nivoa.

- komunikacioni nivo:
 - identifikator klijenta
 - informacije o kanalima pridruženim klijentu
 - IP adresa
- deskriptivni nivo:
 - identifikator klijenta
 - informacije o baferima zauzetim na klijentu
 - memorijska slika klijenta
 - podržane grafičke operacije
 - podržane rezolucije prikaza i dekodovanje videa
 - informacije o audio dekođerima
 - spisak događaja koje klijent može generisati

Modul za rukovanje komandama omogućuje generisanje podržanih RVU-RUI komandi. Usled vremenske zavisnosti između njih potrebno je obezbediti njihovo pravilno izvršavanje. Kako bi se rešio ovaj problem razvijen je algoritam koji vodi evidenciju o redosledu izvršavanja.

Kao argumenti komandi javljaju se identifikatori bafera zauzetih na klijentskoj strani. Ovim baferima rukuje poseban modul koji vodi evidenciju zauzetosti bafera na klijentskoj strani, a takođe omogućuje i oslobađanje kao i pretragu na osnovu identifikatora bafera. Potrebno je izvršiti sprezanje ovog modula sa komandnim modulom kako bi se obezbedilo pravilno funkcionisanje prethodno spomenutog algoritma.

Centralni modul, odnosno jezgro RVU-RUI poslužitelja koristi sve prethodno navedene module kako bi realizovao funkcionalnost definisanu specifikacijom. Funkcionalnost poslužitelja je moguće podeliti u četiri celine koje čine četiri podmodula:

- generalni modul - omogućuje dodavanje, uklanjanje klijenata, rukovanje informacijama o klijentima
- grafički modul - omogućuje slanje elemenata GKS-e i upravljanje njihovim iscertavanjem na klijentu
- video modul - omogućuje pozicioniranje video prozora unutar GKS-e, kao i podešavanje rezolucije dekodovanja i Z ose koja definiše poziciju različitih slojeva (sloj za prikaz, video sloj)
- audio modul - omogućuje slanje audio podataka kao i upravljanje njihovom reprodukcijom

Svoju funkcionalnost centralni modul stavlja na raspolaganje ostatku sistema putem aplikacione programske sprege (APS) RVU-RUI poslužitelja. Korištenjem ove sprege moguće je kreirati RVU aplikacije. Proces kreiranja aplikacije korištenjem APS-e RVU-RUI poslužitelja bi bio veoma spor i složen zadatak, kako bi se ovaj proces olakšao i ubrzao iskorišten je Qt radni okvir o kojem će biti reči u narednim poglavljima.

3.1.4 Prenos multimedijalnih datoteka

Glavni zadatak ovog podprotokola je isporuka multimedijalnog sadržaja RVU klijentima. Prenos multimedijalnih datoteka se zasniva na DLNA medija transfer zahtevima, koji se dalje oslanjaju na HTTP protokol ili opciono RTSP protokol. Pored ovoga omogućeni su različiti modovi reprodukcije (ubrzano, usporeno, ...), izbor audio elementarnog toka koji se želi reprodukovati, kao i zaštita sadržaja koji se prenosi putem DTCP-IP protokola.

Prenos multimedijalnih datoteka započinje tako što RVU poslužitelj korištenjem kontrolera DMR-a, ili u ovom rešenju putem DMC-a proziva akcije DMR servisa. Akcijama "ConnectionManager" UPnP servisa formira se veza između RVU poslužitelja i RVU klijenta. Nakon toga DMC koristeći akcije AVTransport servisa postavlja putanju do datoteke (engl. Uniform Resource Identifier - URI) i upravlja njenom reprodukcijom. Ove akcije se prosleđuju do medija plejera koji predstavlja i HTTP klijenta. Nakon startovanja reprodukcije RVU poslužitelj vrši prenos podataka od ugrađenog HTTP poslužitelja do RVU klijenta na kojem se obavlja sama reprodukcija.

DLNA specifikacijom predviđena su proširenja HTTP GET zahteva koja je potrebno realizovati na HTTP poslužitelju i HTTP klijentu. Akcijama "AVTranpost" servisa modul za reprodukciju sadržaja se prevodi između stanja definisanih RVU specifikacijom.

U ovom rešenju HTTP poslužitelj (realizovana verzija 1.1 HTTP protokola) će biti ugrađen u DMS. Zahvaljujući ovome sadržaj deljen korištenjem DMS-a je moguće isporučiti do klijenta putem ovog poslužitelja. Važno je još jednom napomenuti da DMS ima mogućnost deljenja sadržaja i od strane spoljašnjih poslužitelja kada se ugrađeni HTTP poslužitelj ne koristi. Kako bi se ispoštovala RVU i DLNA specifikacija potrebno je implementirati sva obavezna proširenja HTTP zahteva.

3.1.5 Kvalitet usluge i dijagnostika

Podprotokol za kontrolu kvaliteta usluge i dijagnostiku se koristi kako bi se obezbedio konzistentno visok kvalitet usluge unutar kućne mreže. RVU protokol rešava ovaj problem prioritizacijom paketa zasnovanom na DLNAQOS modelu. Iako je kvalitet usluge kontrolisan od strane poslužitelja, i poslužitelj i klijent su zaduženi za označavanje paketa prema DLNAQOS specifikaciji. DLNAQOS definiše četiri nivoa prioriteta koja su ovom rešenju realizovana podešavanjem prioriteta paketa ugrađenog HTTP poslužitelja.

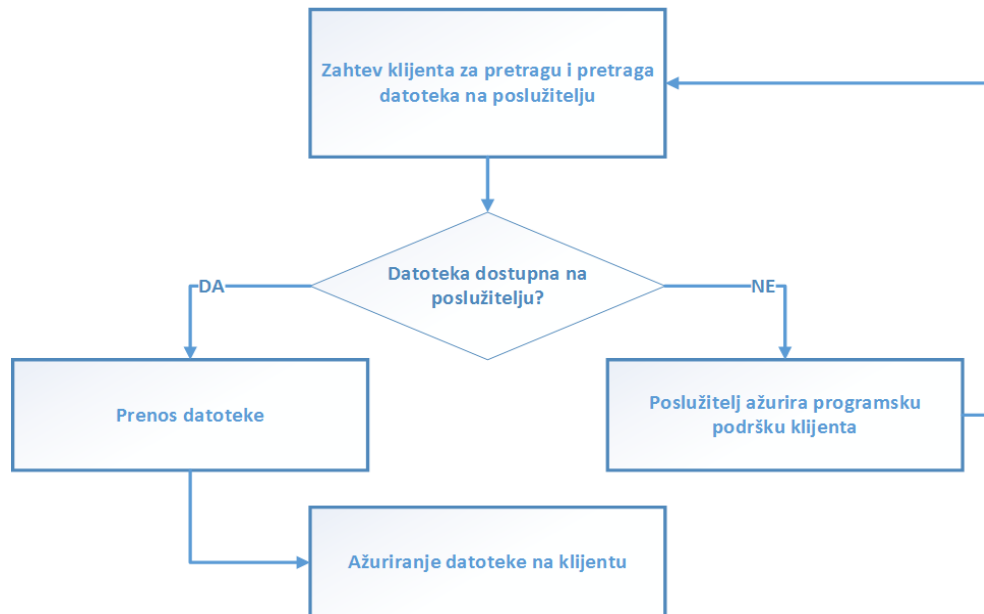
3.1.6 Pribavljanje datoteka

Kao što je ranije navedeno RVU protokol definiše novi CIA podprotokol kako bi omogućio dobavljanje izvršnih, deljenih ili statičkih datoteka, i na taj način dinamički ažurirao RVU programsku podršku klijenta. Prilikom pokretanja klijent upoređuje verziju svoje programske podrške i one dostupne na poslužitelju. Ukoliko poslužitelj poseduje noviju verziju putem TFTP protokola omogućeno je ažuriranje programske podrške.

CIA podprotokol je opcionalna funkcionalnost RVU poslužitelja i sastoji se iz dva dela:

- CIA servis - Korištenjem ovog servisa vrši se razmena informacija o verzijama datoteka. Ovaj servis je veoma jednostavan i sastoji se od promenljive stanja koja opisuje verziju datoteke. Kada se ova promenljiva promeni klijent biva obavešten o promeni kako bi eventualno dobio noviju verziju datoteke. TFTP putanja do nove datoteke se dobavlja kao izlazni argument "CheckImage" UPnP akcije.
- Protokol kojim se vrši prenos datoteke - Prenos datoteke od RVU poslužitelja do RVU klijenta se vrši putem ranije navedenog TFTP protokola, dok proces ažuriranja datoteke na strani poslužitelja nije definisan specifikacijom i prepušten je proizvoljnoj realizaciji.

Sekvenca ažuriranja RVU programske podrške klijenta je prikazana na slici:



Slika 14. Ažuriranje RVU programske podrške klijenta

Na strani poslužitelja potrebno je ugraditi CIA servis u DMS uređaj kako bi klijent mogao da proverava verzije datoteka (Slika 9.), a takođe je potrebno implementirati TFTP poslužitelja putem kojeg će biti vršen prenos datoteka.

3.1.7 Aplikativna programska sprega RVU poslužitelja

Prethodno navedena funkcionalnost RVU poslužitelja je stavljena na raspolaganje ostatku sistema kroz APS-u RVU poslužitelja. APS-u je moguće podeliti prema gradivnim blokovima RVU poslužitelja:

- APS-a DMS-a
- APS-a ugrađenog RUI poslužitelja
- APS-a DMC-a
- APS-a RVU-RUI poslužitelja

3.2 Qt dodaci

Kako bi se formirala veza između RVU poslužitelja i Qt radnog okvira potrebno je realizovati sledeće dodatke.

3.2.1 Qt RVU-RUI dodatak

Qt RVU-RUI dodatak predstavlja realizaciju klasa QPA sloja kojom se formira veza između Qt radnog okvira i RVU poslužitelja, odnosno RVU-RUI poslužitelja. Proces realizacije

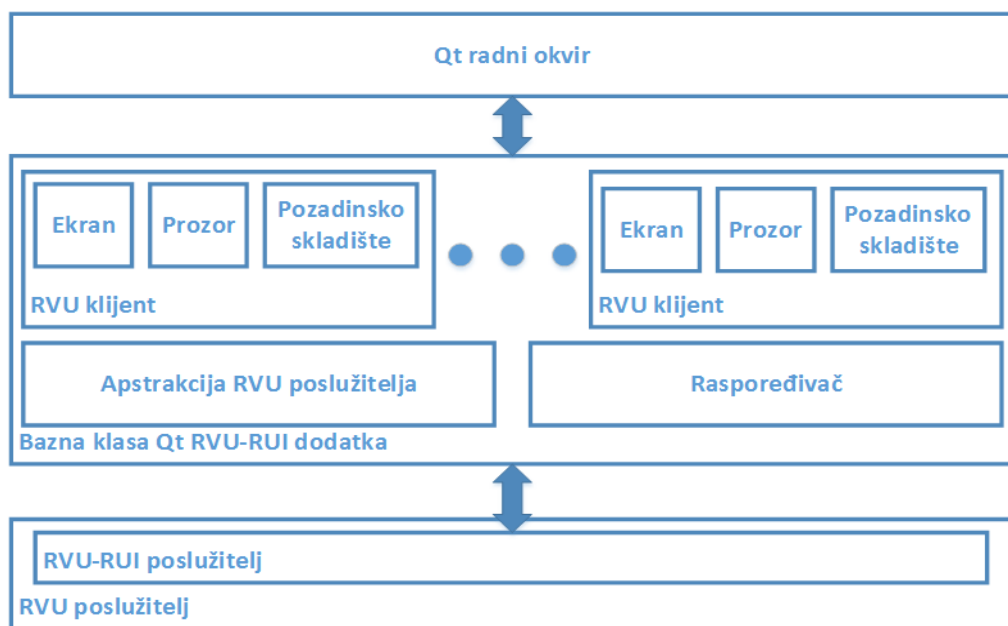
QPA sloja kratko je prikazan u [4]. Potrebno je kreirati modul za upravljanjem sistemom prozora i u zavisnosti od platforme dodati spregu prema temama koje će određivati izgled GKS-e.

Proces kreiranja dodatka započinje realizacijom fabričke klase čiji objekat će biti kreiran prilikom pokretanja aplikacije i na taj način omogućiti učitavanje adekvatnog dodatka. Prilikom učitavanja fabrika formira objekat bazne klase. Potrebno je realizovati baznu klasu putem koje će se pristupati resursima platforme. Objekat bazne klase se nalazi u vlasništvu aplikacije, kako bi aplikacija po potrebi mogla da pristupa resursima platforme. Konkretno u ovom projektu ovaj objekat omogućuje pristup RVU-RUI poslužitelju.

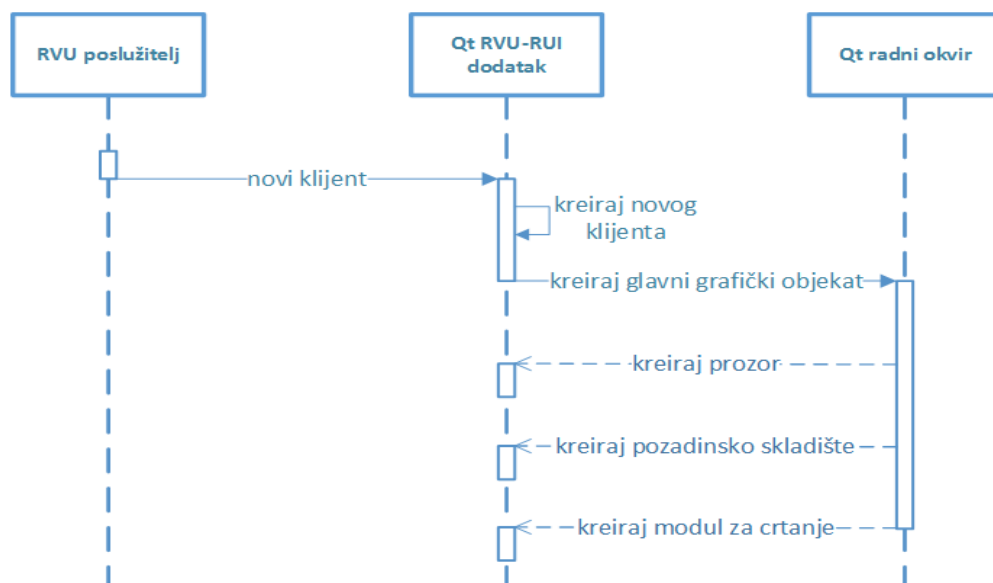
Obavezne komponente bazne klase koje je potrebno realizovati su RVU poslužitelj i klijent koje apstrahuju njima pridružene uređaje. Ostale komponente bazne klase su podklase koje nasleđuju apstraktne klase QPA sloja. Pregled komponenti bazne klase:

- RVU poslužitelj - "Singleton" klasa koja omogućuje kreiranje RVU poslužitelja, kao i pristup njegovoj funkcionalnosti. Objekat ove klase sadrži mapu povezanih RVU klijenata.
- Raspoređivač (engl. dispatcher) - Omogućuje propagaciju događaja generisanih od strane RVU klijenata do odgovarajućih komponenti Qt radnog okvira. Objekat ove klase se kreira prilikom pokretanja aplikacije i isti je za sve RVU klijente.
- RVU klijent - Klasa čiji će objekti predstavljati RVU klijente. Svaki RVU klijent poseduje po jedan objekat od sledeće četiri klase kojim se omogućuje upravljanje iscertavanjem GKS-e. Qt radni okvir kreira ove objekte, pri čemu se u procesu kreiranja oni sprežu sa odgovarajućim RVU klijentom.
- Ekran (engl. screen) - Apstrakcija ekrana RVU klijenta, putem kojeg je moguće odrediti veličinu prozora u kojem će se iscertavati aplikacija. Ova klasa korištenjem APS-e RVU-RUI klijenta u svom konstruktoru dobavlja informacije o ekranu klijenta. Ovako kreirani objekat se koristiti od strane Qt radnog okvira da bi se dobavile informacije o ekranu RVU klijenta.
- Prozor (engl. window) - Apstrakcija prozora aplikacije. Objekti klase prozor se nalaze unutar prethodno opisanog ekrana. Kako se na RVU klijentu aplikacija iscertava uvek preko celog ekrana, moguće je koristiti podrazumevanu (engl. default) realizaciju ove klase. Svaki RVU klijent poseduje po jedan objekat klase prozor uz pomoć kojeg se vrši identifikacija klijenta. Kada Qt radni okvir želi da izvrši neku operaciju nad prozorom, tada se uz pomoć identifikatora prozora određuje na kom RVU klijentu je potrebno sprovesti tu operaciju.

- Pozadinsko skladište (engl. *backingStore*) - Apstrakcija površine (engl. *surface*) prozora koja sadrži kontekst koji je potrebno iscrtati na strani klijenta. Korištenjem metoda ove klase Qt radni okvir ažurira GKS-u na RVU klijentu.
- Modul za crtanje (engl. *paintEngine*) - Modul koji omogućuje iscrtavanje pojedinačnih komponenti GKS-e lokalno na RVU poslužitelju, nakon čega se tako formirana komponenta putem APS-e RVU-RUI poslužitelja prenosi do RVU klijenta i iscrtava na njemu. Nakon što se završi ažuriranje GKS-e u grafičkom baferu (engl. "off-screen" buffer) na RVU klijentu, putem metode pozadinskog skladišta dolazi do zamene tog grafičkog bafera i bafera za prikaz (engl. *double buffering*).



Slika 15. Prikaz strukture Qt RVU-RUI dodatka

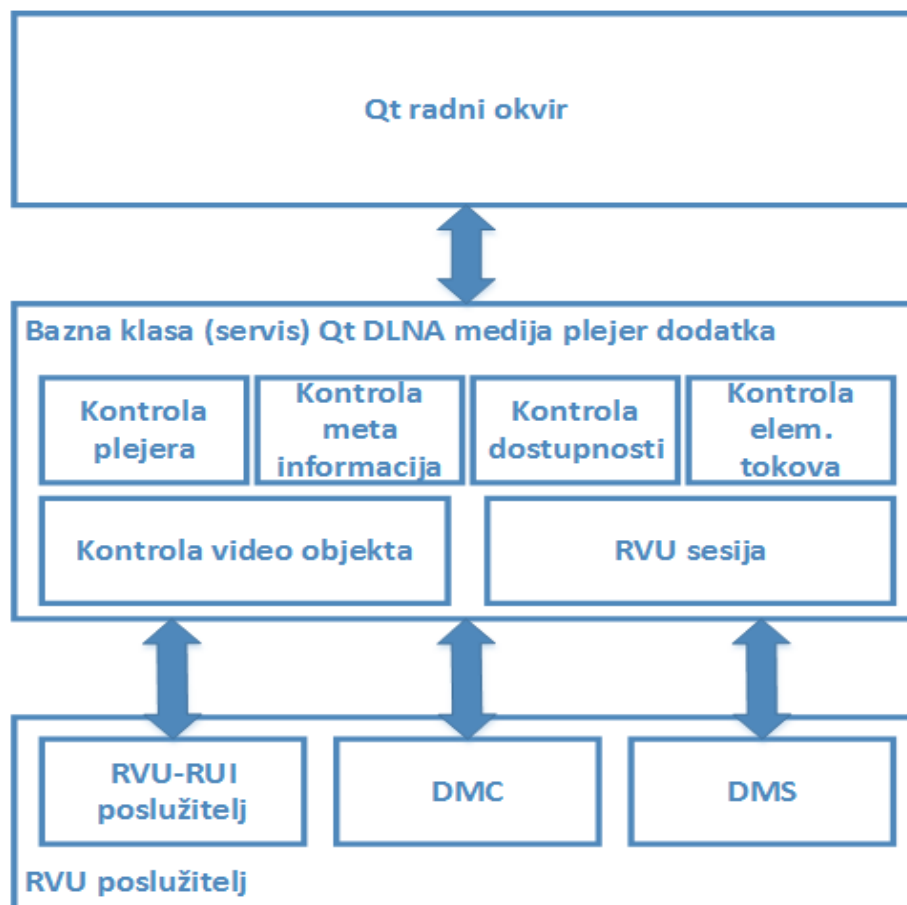


Slika 16. Prikaz dodavanja novog klijenta i sprežnih objekata

RVU poslužitelj dostavi informaciju Qt RVU-RUI dodatku da se pojavio novi klijent, nakon čega dodatak kreira objekat RVU klijenta kojim se vrši njegova apstrakcija. Za tog klijenta se dodaje odgovarajući ekran i kreira novi glavni grafički objekat (engl. main widget). Nakon toga Qt radni okvir dalje kreira prozor, pozadinsko skladište tog prozora i modul za crtanje uz pomoć kojeg se crta po tom pozadinskom skladištu. Ova tri objekta se sprežu sa kreiranim klijentom i na taj način se stvara veza između RVU klijenta i Qt radnog okvira, odnosno na taj način je klijent predstavljen u okvir Qt radnog okvira.

3.2.2 Qt DLNA medija plejer dodatak

Prethodno koncipirani Qt RVU-RUI dodatak zadužen je za kreiranje GKS-e na RVU klijentu. Kako bi se unutar GKS-e omogućila reprodukcija multimedijalnih datoteka potrebno je realizovati dodatni Qt dodatak unutar Qt multimedija modula. Ovaj Qt DLNA medija plejer dodatak predstavlja vezu između Qt radnog okvira i RVU poslužitelja, odnosno DMC-a i DMS-a RVU poslužitelja. DMS omogućuje deljenje željenog sadržaja dok se korištenjem DMC-a vrši pretraga sadržaja i upravljanje njegovom reprodukcijom.



Slika 17. Prikaz struktura Qt DLNA medija plejer dodatka

Analogno Qt RVU-RUI dodatku za početak je potrebno realizovati fabričku klasu. Ova klasa kreira neophodni objekat baznog servisa koji omogućuje pristup kontrolnim klasama Qt DLNA medija plejer dodatka. Ostatak funkcionalnosti fabričke klase su mogućnost brisanja servisa, dobavljanje mogućnosti servisa kao i procena mogućnosti reprodukcije nekog sadržaja u zavisnosti od formata. Bazna klasa (servis) koju je potrebno realizovati treba da omogući da na osnovu ključne reči koju prosledi Qt radni okvir pronade traženu kontrolu i stavi je na raspolaganje radnom okviru. Putem kontrolnih klasa Qt radni okvir realizuje željenu funkcionalnost. Sa druge kontrolne klase ne pristupaju APS-i RVU poslužitelja direktno nego putem posebne klase RVU sesija koja omotava (engl. wrap) funkcionalnost RVU poslužitelja. Klasa RVU sesija pruža veći nivo apstrakcije RVU poslužitelja i olakšava realizaciju neophodnih kontrolnih modula. Qt DLNA medija plejer spada u medija plejer klasu dodataka za koju je potrebno realizovati sledeće kontrole:

- Kontrola plejera - treba da omogući kontrolu reprodukcije multimedijalnih datoteka. Svoju funkcionalnost realizuje promenom stanja mašine stanja (engl. Finite State Machine) RVU sesije. Stanja RVU sesije se preslikavaju na stanja "AVTransport" servisa ugrađenog u DMR RVU klijenta. Dodatne sposobnosti ovog modula koje treba realizovati su dobavljanje informacija o stanju trenutne reprodukcije kao i podešavanje jačine audio tona. Podešavanje jačine tona se vrši putem "RenderingControl" servisa DMR-a.
- Kontrola meta informacija - treba da omogući dobavljanje meta informacija, poput: žanra, albuma, autora, rejtinga, itd. Informacije se prikupljaju korišćenjem DMC-a, koji nakon što pronade datoteku na ugrađenom DMS-a iz DLNA objekta koji reprezentuje datoteku izvlači meta informacije. Izvlačenje meta podataka se vrši kada RVU sesiji kontroler plejera zada datoteka koju je potrebno reprodukovati.
- Kontrola dostupnosti - treba da omogući dobavljanje informacija o stanju DMR-a RVU klijenta, npr. proveru da li DMR RVU klijenta može da izvrši reprodukciju datoteke.
- Kontrola elementarnih tokova - treba da omogući upravljanje elementarnim tokovima koji se nalaze unutar transportnog toka. Poslužitelj koji deli transportni tok mora DMC-u isporučiti informacije o dostupnim audio i video PID-ovima. Nakon što DMC dobije ove informacije koristeći RVU proširenja "RenderingControl" UPnP servisa treba da podesi PID elementarnog toka koji se želi reprodukovati na RVU klijentu.
- Kontrola video objekta (engl. video widget) - Ova kontrola sadrži video objekat koji predstavlja komponentu GKS-e, odnosno video prozor unutar kojeg se vrši

reprodukcija video datoteke. Ovaj modul spregnut je sa Qt RVU-RUI dodatkom, kako bi se omogućilo pozicioniranje video prozora unutar GKS-e RVU klijenta. Proces pozicioniranja se obavlja bez učešća klase RVU sesija i po tome je ovaj modul specifičan u odnosu na ostale kontrole. Dodatne mogućnosti ovog kontrolera su postavljanje osvetljaja, kontrasta, zasićenja ekrana RVU klijenta koje se obavljaju pomoću DMC-a.

3.3 Qt radni okvir

Realizacijom Qt RVU-RUI i Qt DLNA medija plejer dodataka formirana je veza između RVU poslužitelja i Qt radnog okvira usled čega je stvorena potencijalna mogućnost za razvoj RVU aplikacija uz pomoć Qt radnog okvira. Kako bi se ovo zapravo omogućilo realizovana je klasa RVU aplikacija koja nasleđuje ugrađenu klasu Qt aplikacija. Kada se kreira objekat klase RVU aplikacija kao jedan od parametar se prosleđuje tip aplikacije. Ukoliko se kroz ovaj tip navede da se ne radi o RVU aplikaciji klasa RVU aplikacija se ponaša identično kao i nadklasa. Sa druge strane ukoliko se navede RVU tip aplikacije učitaće se prethodno realizovana dva dodatka i na taj način će se ista aplikacija izvršavati u RVU kontekstu. Qt grafički objekat (engl. widget) koji se prosledi RVU aplikaciji predstavlja kontekst aplikacije, odnosno ono što će RVU klijent videti na svom ekranu. Kako bi se omogućilo dinamičko zadavanje željenog Qt objekta, bez potrebe za ponovnim prevođenjem Qt radnog okvira, Qt RVU aplikaciji se kao dodatni parametar prosleđuje objekat fabrike Qt objekata. Programer koji razvija Qt aplikaciju realizuje ovu fabriku i u njoj navodi koji Qt objekat želi da kreira kada se na RVU poslužitelja poveže novi RVU klijent. Na ovaj način kada klasa RVU poslužitelja u Qt RVU-RUI dodatku pozove metodu za kreiranje objekta fabrike dinamički će se kreirati željeni objekat.

4. Programsko rešenje

U ovom poglavlju opisano je programsko rešenje RVU poslužitelja i Qt proširenja. RVU poslužitelj realizovan je u programskom jeziku C, dok su Qt RVU-RUI i Qt DLNA medija plejer dodaci kao i Qt RUI aplikativna klasa realizovane u C++. RVU poslužitelj je preveden kao dinamička biblioteka koja se povezuje sa Qt RVU-RUI i Qt DLNA medija plejer dodacima u cilju realizacije njihove funkcionalnosti. U zavisnosti od ulaznih parametara Qt radni okvir učitava ove dodatke i na taj način omogućuje izvršavanje Qt aplikacija u RVU kontekstu. RVU kontekst podrazumeva izvršavanje aplikacije na RVU poslužitelju, dok se kontekst aplikacije iscrtava na RVU klijentu.

RVU poslužitelj je programska biblioteka koja apstrahuje funkcionalnost DMS-a, RUIS-a, DMR-a i RVU-RUI poslužitelja. DMS se koristi kako bi se omogućilo deljenje sadržaja, RUIS omogućuje deljenje GKS-a, dok se procesom prenosa i reprodukcijom datoteke na RVU klijentu upravlja putem DMC-a. Ugrađeni RVU-RUI modul omogućuje kreiranje uniformne GKS-e na različitim RVU klijentima, bez obzira na tip platforme na kojem se taj klijent izvršava. Realizacijom ove biblioteke omogućeno je kreiranje RVU aplikacija, međutim ovaj proces je znatno kompleksniji i dugotrajniji u odnosu na okruženja za razvoj aplikacija poput Qt radnog okvira. Upravo zbog toga je upotrebljen ovaj radni okvir kako bi se proces kreiranja aplikacija olakšao, a samim tim i ubrzao. Sprega između Qt radnog okvira i RVU poslužitelja ostvarena je kroz prethodno dva spomenuta dodatka. Istu Qt aplikaciju moguće je izvršavati lokalno ili udaljeno u RVU kontekstu, bez potrebe za bilo kakvom modifikacijom, što je dodatna prednost ovog rešenja.

4.1 RVU poslužitelj

RVU poslužitelja ima složenu strukturu prikazanu na slici 10. Za realizaciju ugrađenog DMS-a, RUI poslužitelja i DMC-a kao polazna tačka korištena je DLNA biblioteka razvijena na institutu RT-RK. Ova početna realizacija je modifikovana kako bi se prilagodila RVU protokolu, odnosno kako bi se dodala proširenja uređaja i servisa definisanih RVU specifikacijom. U cilju realizacije ugrađenog RVU-RUI poslužitelja u DLNA biblioteku dodat je novi RVU-RUI modul, na koji će prilikom opisa biti stavljen poseban akcenat. Na ovaj način formirana je RVU programska biblioteka koja realizuje funkcionalnost RVU poslužitelja.

4.1.1 DMS i RUI poslužitelj

DMS je DLNA klasa koja realizuje “MediaServer” UPnP uređaj. Kako bi se formirao ovaj uređaj iskorištena je UPnP biblioteka. Realizovani DMS poseduje sve obavezne UPnP servise DMS DLNA klase i RVU poslužitelja, a to su: “ConnectionManager” i “ContentDirectory” servisi. U ovom rešenju su realizovane sve obavezne akcije i promenljive stanja ovih servisa iz verzije 1.0 UPnP specifikacije. Početna realizacija UPnP servisa "ConnectionManager" je proširena "PrepareForConnection" i "ConnectionComplete" UPnP akcijama kako bi se zadovoljili zahtevi RVU specifikacije. Kao ugrađeni HTTP poslužitelj iskorišten je “mongos” (engl. mongoose) otvoreni (engl. “open-source”) HTTP poslužitelj putem kojeg je omogućena dostava deljenih datoteka. Za skladištenje informacija o deljenim datotekama iskorištena je “skuel-lajt” (engl. sqlite) biblioteka, koja realizuje relacionu bazu podataka, dok je zaštita podataka prilikom prenosa ostvarena upotrebom DTCP-IP biblioteke. Realizacija DMS DLNA klase je preuzeta iz RT-RK DLNA biblioteke, pri čemu pomenuta proširenja nisu dovela do promene APS-e DMS-a. U cilju boljeg sagledavanja kompletnog rešenja u daljem tekstu data je APS-a DMS-a, koja se koristi od strane Qt DLNA medija plejer dodatka.

Svaka od DLNA klasa predstavljena je strukturom *dlna_dev_t* koja sadrži sledeća polja:

- tip DLNA klase (DMS, DMR, RUISrc, RUISink)
- ime uređaja
- identifikator uređaja - UDN uređaja
- IP adresa uređaja
- pokazivač na ugrađeni uređaj ukoliko postoji
- pokazivač na roditeljski uređaj ukoliko postoji
- privatni podaci

Datoteke koje se dele putem DMS-a predstavljene su strukturom *dlna_object_t*, pri čemu su najvažnija polja ove strukture:

- identifikator datoteke
- identifikator roditeljskog direktorijuma
- tip objekta (direktorijum, slika, video, audio, objekat frekvencijskog odabirača)
- ime datoteke
- DLNA profil objekta [12]
- putanja do objekta - URI
- MIME tip
- adresa DTCP izvora (ukoliko je sadržaj zaštićen DTCP-IP protokolom)
- prolaz DTCP izvora (ukoliko je sadržaj zaštićen DTCP-IP protokolom)
- struktura sa meta informacijama

DLNA objekti se nalaze u direktorijima (poseban tip DLNA objekta) koji su predstavljeni strukturom *dlna_dir_t*:

- identifikator direktorijuma
- identifikator roditeljskog direktorijuma
- ime
- broj podobjekata u direktorijumu (engl. childer count)
- indeks trenutne pozicije u direktorijumu (indeks podobjekta)
- privatni podaci

Objekat frekvencijskog odabirača je definisan strukturom *dlna_tuner_handler_t*:

- identifikator frekvencijskog odabirača
- ime frekvencijskog odabirača
- lista DTV servisa, pri čemu je svaki servis predstavljen strukturom *dlna_broadcast_channel_t*, koja sadrži:
 - profil DTV servisa
 - MIME tip
 - ime servisa
 - putanja do poslužitelja koji isporučuje DTV servis
 - indeks servisa
 - adresa DTCP izvora (ukoliko je sadržaj zaštićen DTCP-IP protokolom)
 - prolaz DTCP izvora (ukoliko je sadržaj zaštićen DTCP-IP protokolom)
- dužina liste servisa

Svaka funkcija DLNA biblioteke vraća grešku predstavljenu nabrojanjem (engl. enum) *dlna_error_t* čije vrednosti mogu biti:

- greška ne postoji
- biblioteka nije inicijalizovana
- funkcija nije implementirana
- greška u redosledu poziva funkcija nad DLNA klasom
- greška u redosledu poziva funkcija nad DMC-om
- nepoznata greška (ne zna se šta je uzrok greške)
- nema dovoljno memorije
- željeni sistemski resurs nije dostupan
- nema više dostupnih resursa
- nema više dostupnih rukovaoca (engl. handle)
- opšta UPnP greška

Funkcije:

dlna_error_t dlna_dms_create(dlna_dev_hnd_t* handle, const char* db_path, const char* udn);

Kreiranje DMS uređaja na osnovu zadate baze podataka, i izgenerisanog UDN-a.

Parametri:

- handle - pokazivač na rukovaoc putem kojeg će se kasnije rukovati DMS-om
- db_path - putanja do baze podataka
- udn - identifikator uređaja

dlna_error_t dlna_dms_destory(dlna_dev_hnd_t handle);

Uništavanje ranije kreiranog DMS-a.

Parametri:

- handle - rukovaoc DMS-a

dlna_error_t dlna_dev_set_prop(dlna_dev_hnd_t handle, dlna_dev_dev_prop_t type, char* value);

Podešavanja različitih atributa DMS-a, nakon njegovog kreiranja.

Parametri:

- handle - rukovaoc DMS-a
- type - atribut DMS-a (ime, ime proizvođača, model, opis modela, broj modela, itd.)

- value - vrednost atributa

dlna_error_t dlna_dms_dtcp_enable(dlna_dev_hnd_t handle, uint16_t port);

Ospostavljanje DTCP podrške ugrađenog HTTP poslužitelja.

Parametri:

- handle - rukovaoc DMS-a
- port - prolaz DTCP izvora

dlna_error_t dlna_dms_start(dlna_dev_hnd_t handle, uint16_t streaming_port);

Pokretanje DMS-a. Kao drugi parametar se prosleđuje prolaz koji će se koristiti od strane ugrađenog HTTP poslužitelja za distribuciju sadržaja.

Parametri:

- handle - rukovaoc DMS-a
- streaming_port - prolaz ugrađenog HTTP poslužitelja

dlna_error_t dlna_dms_stop(dlna_dev_hnd_t handle);

Stopiranje prethodno pokrenutog DMS-a.

Parametri:

- handle - rukovaoc DMS-a

dlna_error_t dlna_dms_share_file(dlna_dev_hnd_t handle, char* path, int use_dtcp);

Pozivom ove funkcije omogućuje se deljenje multimedijalne datoteke od strane DMS-a.

Parametri:

- handle - rukovaoc DMS-a
- path - putanja do datoteke
- use_dtcp - indikacija zaštite datoteke putem DTCP-IP protokola

dlna_error_t dlna_dms_unshare_file(dlna_dev_hnd_t handle, char* path);

Brisanje datoteke iz baze datoteka dostupnih na DMS-u.

Parametri:

- handle - rukovaoc DMS-a
- path - putanja do datoteke

dlna_error_t dlna_dms_share_tuner(dlna_dev_hnd_t handle, tuner_handler_t* tuner_hnd);

Deljenje objekta frekvencijskog odabirača.

Parametri:

- handle - rukovaoc DMS-a
- tuner_hnd - pokazivač na strukturu koja opisuje frekvencijski odabirač

```
dlna_error_t dlna_dms_unshare_tuner(dlna_dev_hnd_t handle, tuner_handler_t* tuner_hnd);
```

Brisanje frekvencijskog odabirača.

Parametri:

- handle - rukovaoc DMS-a
- tuner_hnd - pokazivač na strukturu koja opisuje frekvencijski odabirač

RVU poslužitelj poseduje ugrađenog RUI poslužitelja, kako bi omogućio deljenje RVU-RUI udaljene GKS-e. Po principu uvedenom u konceptu rešenja RUI poslužitelj je ugrađen u DMS. Realizacija RUI poslužitelja u RT-RK DLNA biblioteci je morala biti prilagođena kako bi se omogućilo deljenje RVU-RUI GKS-e. Adaptacija RUI poslužitelja izvršena je modifikacijom "RemoteUIServer" UPnP servisa, tj modifikacijom njegove "GetComatibleUIs" akcije. Ova akcija je proširena podrškom deljenja RVU-RUI udaljene GKS-e. Putem ove akcije RVU klijent dobija informacije o ugrađenom RVU-RUI poslužitelju (njegovoj IP adresi i prolazu), dostavlja se opis RVU-RUI GKS-e. Takođe je dodata podrška za filtriranje spiska (engl. listing) GKS-a koji se može naći kao ulazni parametar akcije "GetCompatibleUIs". Kada se do RUI poslužitelja dostavi spisak, tada je odgovor (izveštaj) na akciju "GetCompatibleUIs" samo tražena GKS-a ukoliko ona postoji na RUI poslužitelju. Važno je napomenuti da APS-a RUI poslužitelja nije morala biti modifikovana kako bi se obezbedila ova funkcionalnost.

Pregled APS-e RUI poslužitelja:

```
dlna_error_t dlna_ruis_create(dlna_dev_hnd_t* handle, const char* udn);
```

Kreiranje ugrađenog RUI poslužitelja.

Parametri:

- handle - pokazivač na rukovaoc putem kojeg će se kasnije rukovati RUI poslužiteljem. Struktura koja opisuje DMS, sadrži pokazivač na ovog rukovaoca, i obrnuto struktura koja opisuje RUIS sadrži pokazivač na roditeljski uređaj, tj. na DMS.
- udn - identifikator uređaja

dlna_error_t dlna_ruis_destroy(dlna_dev_hnd_t handle);

Uništavanje ranije kreiranog RUI poslužitelja.

Parametri:

- handle - rukovaoc RUI poslužitelja

dlna_error_t dlna_ruis_enable(dlna_dev_hnd_t handle, int enable);

Pokretanje i stopiranje RUI poslužitelja.

Parametri:

- handle - rukovaoc RUI poslužitelja
- enable - ukoliko se kao vrednost ovog parametra prosledi broj različit od 0 RUI poslužitelj će se pokrenuti, a ukoliko se prosledi 0 dolazi do zaustavljanja RUI poslužitelja.

dlna_error_t dlna_ruis_add_ui(dlna_dev_hnd_t handle, dlna_ruis_ui_t* ui, char* name, char* desc);

Dodavanje korisničke sprege u listu deljenih sprege.

Parametri:

- handle - rukovaoc RUI poslužitelja
- ui - pokazivač na dodatu GKS-u
- name - ime GKS-e
- desc - opis GKS-e

dlna_error_t dlna_ruis_remove_ui(dlna_dev_hnd_t handle, dlna_ruis_ui_t ui);

Brisanje korisničke sprege iz liste deljenih sprege.

Parametri:

- handle - rukovaoc RUI poslužitelja
- ui - pokazivač GKS-u koju je potrebno obrisati

dlna_error_t dlna_ruis_update_ui(dlna_dev_hnd_t handle);

Ažuriranje RUI kontrolnih tačaka, nakon što se doda nova GKS-e na RUI poslužitelju.

Parametri:

- handle - rukovaoc RUI poslužitelja

4.1.2 DMC

Realizovani DMC sadrži kontrolne tačke sledeća četiri servisa: “ContentDirectory”, “ConnectionManager”, “AVTransport” i “RenderingControl” koji se nalaze u DMS-u i DMR-u. Kontrolna tačka “ContentDirectory” servisa je dodata kako bi se omogućila kontrola DMS-a. Na ovaj način je obavezni kontroler DMR-a zamenjen DMC-om. Postojeća realizacija DMC-a u RT-RK DLNA biblioteci koja je u skladu sa verzijom 1.0 UPnP specifikacije je nadograđena kako bi se dodala podrška za proširenja definisana RVU specifikacijom. Ova proširenja predstavljena su u prethodno opisanom konceptu rešenja.

DMC omogućuje kontrolu DMS-a i DMR-a. Za svaku od ove dve DLNA klase DMC sadrži po jednu listu u kojima se nalaze dostupni DMS-i, odnosno DMR-i. Servisi koji se nalaze u ovim DLNA klasama predstavljeni su strukturom *ctrl_dev_service_t* koja sadrži sledeća polja:

- identifikator servisa
- tip servisa
- URL događaja koji se koristi prilikom obnavljanja pretplate na servis
- URL kontrole koja se koristi prilikom slanja UPnP akcija servisa
- identifikator pretplate na servis

Svaki uređaj u DMC je predstavljen strukturom *ctrl_dev_desc_t*:

- UDN uređaja
- URL koji ukazuje na opisa uređaja
- ime uređaja
- URL prezentacije
- URL ikonice uređaja
- vreme isteka reklamacije uređaja
- IP adresa uređaja
- MAC adresa uređaja

DMS DLNA klasa je predstavljena strukturom *dms_ctrl_dev_t*:

- polje tipa *ctrl_dev_desc_t* sa opisom uređaja
- polje tipa *ctrl_dev_service_t* sa opisom “ConnectionManager” servisa
- polje tipa *ctrl_dev_service_t* sa opisom “ContentDirectory” servisa

DMR DLNA klasa je predstavljena strukturom *dmr_ctrl_dev_t*:

- polje tipa *ctrl_dev_desc_t* sa opisom uređaja
- polje tipa *ctrl_dev_service_t* sa opisom “ConnectionManager” servisa
- polje tipa *ctrl_dev_service_t* sa opisom “RenderingControl” servisa
- polje tipa *ctrl_dev_service_t* sa opisom “AVTransport” servisa

Nakon prijema opisa uređaja vrši se njegovo raščlanjivanje (engl. parse) u kojem se dobijaju URL-ovi na kojima se nalaze opisi servisa. Raščlanjivanjem opisa servisa dobija se lista dostupnih akcija i promenljivih stanja. DMC eksplicitno vodi računa da se u opisu servisa DMR-a nalaze proširenja definisana RVU specifikacijom. Ukoliko ona nisu dostupna smatra se da taj DMR ne može biti ugrađen u RVU klijenta.

Pregled najbitnijih funkcija APS-e DMC-a:

dlna_error_t dlna_dmc_create(dlna_ctrl_hnd_t* handle, dlna_ctrl_type_t type);

Kreiranje DMC-a.

Parametri:

- handle - pokazivač na rukovaoc putem kojeg će se kasnije rukovati DMC-om
- type - tip kontrolera (DMC ili DMP)

dlna_error_t dlna_dmc_destroy(dlna_ctrl_hnd_t handle);

Uništavanje prethodno kreiranog DMC-a.

Parametri:

- handle - rukovaoc DMC-a

dlna_error_t dlna_dmc_get_current(dlna_ctrl_hnd_t* handle);

Dobavljanje trenutno aktivnog DMC-a (ukoliko je potreban pristup iz drugog modula).

Parametri:

- handle - pokazivač na rukovaoc DMC-a

dlna_error_t dlna_dmc_start(dlna_ctrl_hnd_t handle);

Pokretanje DMC-a kako bi se započela pretraga uređaja.

Parametri:

- handle - rukovaoc DMC-a

dlna_error_t dlna_dmc_stop(dlna_ctrl_hnd_t handle);

Zaustavljanje DMC-a.

Parametri:

- handle - rukovaoc DMC-a

dlna_error_t dlna_dmc_get_server_count(dlna_ctrl_hnd_t handle, uint8_t* cnt);

Dobavljanje broja trenutno raspoloživih DMS-a.

Parametri:

- handle - rukovaoc DMC-a
- cnt - broj trenutno dostupnih DMS-a

dlna_error_t dlna_dmc_get_server(dlna_ctrl_hnd_t handle, uint8_t index, dlna_dev_t* server);

Dobavljanje strukture sa opisom DMS-a.

Parametri:

- handle - rukovaoc DMC-a
- index - indeks DMS-a u listi raspoloživih
- server - pokazivač na strukturu sa opisom DMS-a

dlna_error_t dlna_dmc_get_renderer_count(dlna_ctrl_hnd_t handle, uint8_t* cnt);

Dobavljanje broja trenutno raspoloživih DMR-a.

Parametri:

- handle - rukovaoc DMC-a
- cnt - broj trenutno dostupnih DMR-a

dlna_error_t dlna_dmc_get_renderer(dlna_ctrl_hnd_t handle, uint8_t index, dlna_dev_t* renderer);

Dobavljanje strukture sa opisom DMR-a.

Parametri:

- handle - rukovaoc DMC-a
- index - indeks DMR-a u listi raspoloživih
- renderer - pokazivač na strukturu sa opisom DMR-a

dlna_error_t dlna_dmc_opendir(dlna_ctrl_hnd_t handle, char* udn, char* id, dlna_dir_t** dir);

Otvaranje direktorijuma. Direktorijum se otvara slanjem “Browse” UPnP akcije “ContentDirectory” servisa.

Parametri:

- handle - rukovaoc DMC-a
- udn - UDN DMS na kojem je potrebno otvoriti direktorijum
- id - identifikator direktorijuma

- dir - pokazivač na strukturu otvorenog direktorijuma

dlna_error_t dlna_dmc_closedir(dlna_ctrl_hnd_t handle, dlna_dir_t* dir);

Zatvaranje prethodno otvorenog direktorijuma (oslobađanje zauzetih resursa). Pre otvaranja sledećeg direktorijuma trenutni je potrebno osloboditi, tj. u jednom trenutku je moguće imati samo jedan otvoreni direktorijum.

- handle - rukovaoc DMC-a
- dir - direktorijum kojeg je potrebno zatvoriti

dlna_error_t dlna_dmc_readdir(dlna_ctrl_hnd_t handle, dlna_dir_t* dir, dlna_object_t** object);

Čitanje datoteka iz otvorenog direktorijuma. Direktorijum se čita slanjem “Browse” UPnP akcije “ContentDirectory” servisa.

Parametri:

- handle - rukovaoc DMC-a
- dir - direktorijum kojeg je potrebno pročitati
- object - pokazivač na strukturu koja sadrži predstavu objekta

dlna_error_t dlna_dmc_play(dlna_ctrl_hnd_t handle, char* srv, char* id, char* rnd, float speed);

Reprodukcija datoteke sa zadatog DMS-a na zadatom DMR-u. Komanda apstrahuje dve akcije “AVTransport” servisa. Prva akcija “setAVTransportURI” postavlja putanju do datoteke, a druga “Play” započinje reprodukciju datoteke sa zadatom brzinom.

Parametri:

- handle - rukovaoc DMC-a
- srv - UDN DMS-a
- id - identifikator DLNA objekta koji se želi reprodukovati
- rnd - UDN DMR-a
- speed - brzina reprodukcije

dlna_error_t dlna_dmc_stop(dlna_ctrl_hnd_t handle, char* rndr_udn);

Zaustavljanje trenutne reprodukcije. Izvršava se “Stop” UPnP akcija “AVTransport” servisa.

Parametri:

- handle - rukovaoc DMC-a

- `rndr_udn` - UDN DMR-a na kojem se zaustavlja reprodukcija

`dlna_error_t dlna_dmc_pause(dlna_ctrl_hnd_t handle, char* rndr_udn);`

Pauziranje trenutne reprodukcije. Izvršava se "Pause" UPnP akcija "AVTransport" servisa.

Parametri:

- `handle` - rukovaoc DMC-a
- `rndr_udn` - UDN DMR-a na kojem se pauzira reprodukcija

`dlna_error_t dlna_dmc_seek(dlna_ctrl_hnd_t handle, char* rndr_udn, char* seek_pos);`

Promena trenutne pozicije reprodukcije. Izvršava se "Seek" UPnP akcija "AVTransport" servisa.

Parametri:

- `handle` - rukovaoc DMC-a
- `rndr_udn` - UDN DMR-a na kojem se vrši promena pozicije
- `seek_pos` - željena pozicija reprodukcije (format definisan DLNA direktivama)

4.1.3 RVU-RUI poslužitelj

RVU-RUI poslužitelj predstavlja novi modul koji je dodat u DLNA biblioteku kako bi se realizovala neophodna podrška za RVU-RUI GKS-u. Modul je podeljen prema logičkim celinama koje su organizovane u programski stek, prikazan na slici 13. U narednih nekoliko poglavlja opisana je funkcionalnost svakog od ovih modula kao i njegova sprega sa ostatkom sistema.

4.1.3.1 Modul sa alatima

Analizom RVU specifikacije utvrđena je potreba za sekvencijalnom obradom komandi pristiglih od klijenta. Sa druge strane komande koje se šalju RVU klijentu ne moraju biti obrađene redosledom kojim su poslate, usled mogućnosti odložene obrade za neke tipove komandi. Da bi se odgovorilo na ove zahteve realizovane su sledeće strukture podataka:

- `red` - podržane operacije: inicijalizacija, ubacivanje elementa, dobavljanje elementa, kao i provera trenutne veličine reda
- `lista` - podržane operacije: inicijalizacija, deinicijalizacija, ubacivanje u listu ispred ili iza proizvoljnog elementa, dobavljanje elementa, uklanjanje elementa, pretraga elemenata na osnovu sadržaja.

Realizovane strukture podataka omogućuju konkurentan pristup podacima (engl. "thread-safe") bez narušavanja konzistentnosti.

4.1.3.2 Komunikacioni modul

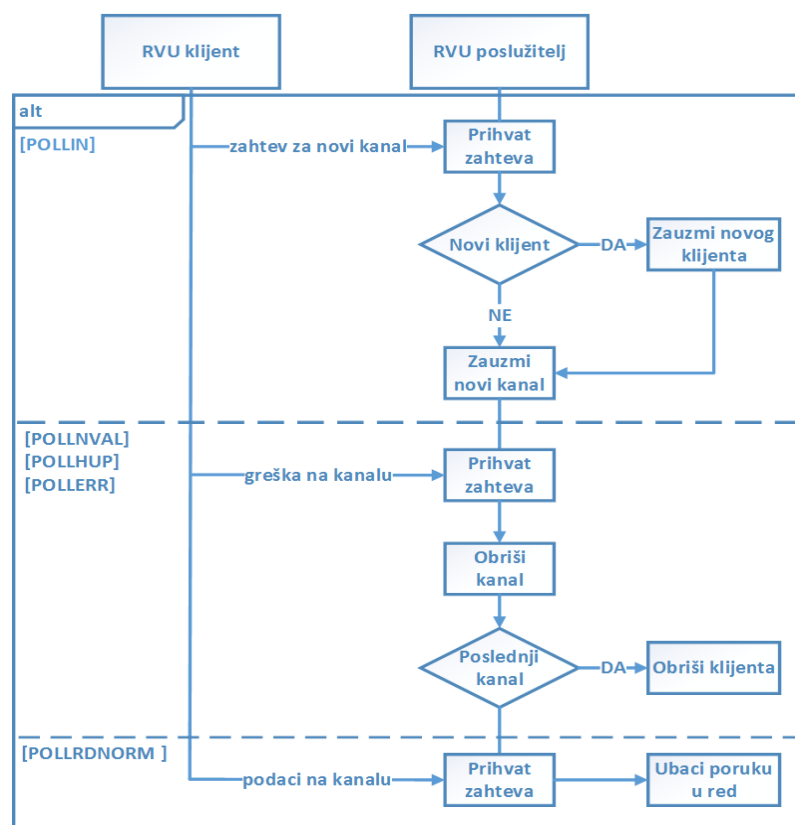
Komunikacioni modul sastoji se iz ugrađenog TCP poslužitelja i TCP klijenta. Tokom inicijalizacije TCP poslužitelja vrši se inicijalizacija celokupnog komunikacionog sistema. Proces inicijalizacije podrazumeva kreiranje utičnice (engl. socket) i njenog povezivanje sa adresom i prolazom RVU-RUI poslužitelja. Prolaz poslužitelja se nalazi u opisu RVU-RUI GKS-e i omogućuje prihvatanje zahteva za otvaranje kanala poslatih od strane RVU klijenta. Kreiranoj utičnici pridružen je jedan ili više deskriptora, pri čemu prvi deskriptor služi za detekciju zahteva za otvaranje novih kanala, a ostali su pridruženi kanalima i ubacuju se u listu nakon njihovog otvaranja. Poslednja etapa procesa inicijalizacije podrazumeva kreiranje niti koja omogućuje asinhrono registrovanje različitih vrsta događaja na utičnici, odnosno na pomenutim deskriptorima. Nakon što se na komunikacionom nivou izvrše neophodne akcije, informacije o događaju se propagiraju do viših programskih slojeva putem povratne funkcije podešene prilikom inicijalizacije modula. TCP poslužitelj detektuje sledeće vrste događaja definisane u zaglavlju *poll.h*:

- **POLLIN** - Detekcija novih zahteva. Ovaj događaj javlja se samo na prvom deskriptoru, nakon čega se proverava da li je zahtev stigao od novog ili postojećeg klijenta. Ukoliko se radi o novom klijentu, šalje se indikacija kako bi se zauzeo deskriptor za njega. Bez obzira da li se radi o postojećem ili novom klijentu, za klijenta se zauzima novi kanal, nakon čega se njegov deskriptor ubacuje u listu deskriptora pridruženih utičnici kako bi se omogućilo praćenje događaja na njemu.
- **POLLNVAL** - Greška na kanalu. Kanal na kojem se dogodila ova greška se uništava.
- **POLLHUP** i **POLLERR** - Greška u komunikaciji. Nakon detekcije ove greške saobraćaj na kanalu se može odvijati samo u jednom smeru usled čega je potrebno zatvoriti kanal.
- **POLLRDNORM** - Podaci su dostupni na kanalu. Na osnovu deskriptora vrši se identifikacija kanala nakon čega se sklapa cela poruka i ubacuje u red za skladištenje primljenih poruka koji je pridružen kanalu.
- **POLLPRI** i **POLLRDBAND** - Hitni podaci su dostupni na kanalu. Ovaj tip događaja se ignoriše jer je vezan za razmenu informacija o utičnicama.

TCP klijent omogućuje slanje zahteva RVU klijentu za otvaranje kanala. Iz prethodno izloženog se jasno da zaključiti da oba RVU elementa (poslužitelj i klijent) moraju u sebi posedovati i TCP poslužitelja i TCP klijenta.

RVU-RUI elementi su na komunikacionom nivou predstavljeni strukturom *rvu_rui_element_desc_t*:

- tip elemenata
- identifikator elementa
- IP adresa
- unija koja sadrži strukturu poslužitelja ili klijenta.
 - struktura sa opisom poslužitelja sadrži:
 - utičnicu
 - listu deskriptora pridruženih utičnici
 - broj deskriptora
 - listu kanala otvorenih od strane poslužitelja
 - struktura sa opisom klijenta sadrži:
 - listu kanala otvorenih od strane klijenta



Slika 18. Prikaz scenarija za različite vrste događaja u komunikacionom modulu

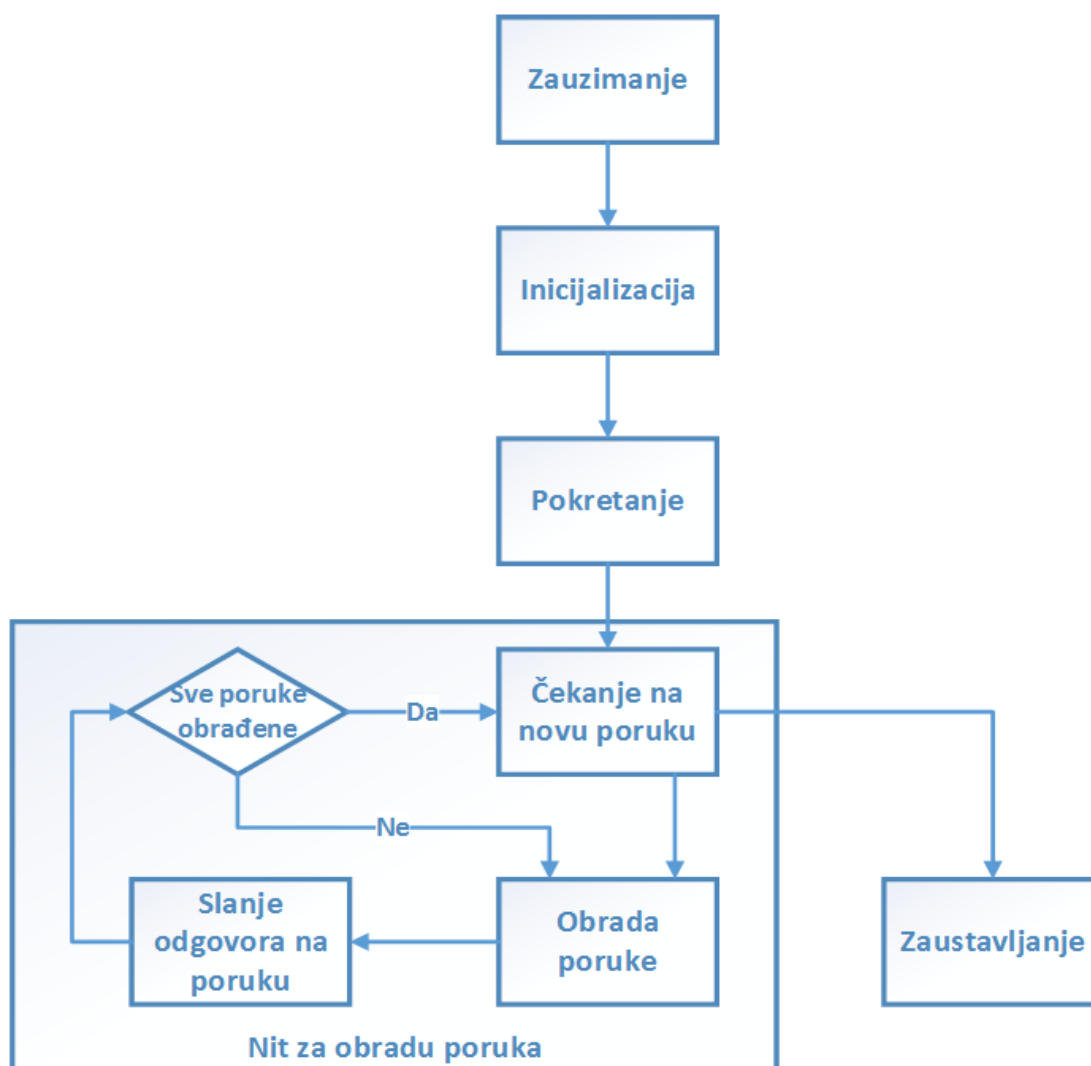
Putem komunikacionog modula viši programski slojevi mogu da šalju generisane komande i podatke RVU-RUI klijentima, otvaraju nove kanale i izvršavaju pojedine akcije koje je potrebno obaviti na komunikacionom nivou nakon obrade komande.

4.1.3.3 Modul za rukovanje kanalima

Modul za rukovanje kanalima omogućuje rukovanje TCP tokovima formiranim između RVU-RVU poslužitelja i RVU-RUI klijenta. Svaki kanal predstavljen je strukturom *rvu_rui_channel_desc_t* koja sadrži:

- tip kanala (komandni kanal, kanal za podatke, nepoznat tip - privremeno stanje do pristizanja prvog bajta koji će odrediti tip kanala)
- deskriptor kanala - veza kanala sa utičnicom
- identifikator kanala
- tip elementa kojem je kanal pridružen (RVU-RUI poslužitelj ili RVU-RUI klijent)
- identifikator elementa
- identifikatore kanala sa kojima je kanal povezan (povezanost komandnih kanala i kanala za podatke)
- ime poslednje komande za koju je stigao odgovor
- identifikator (engl. token) zadnje poslate poruke
- identifikator zadnjeg odgovora na komandu
- povratni port TCP poslužitelja na kojem RVU-RUI klijent čeka zahteve za otvaranje kanala
- rukovaoc niti - nit u kojoj se obrađuju poruke pristigle na kanalu
- red za skladištenje primljenih poruka
- lista za skladištenje poslatih poruka

Funkcije ovog modula omogućuju zauzimanje i inicijalizaciju kanala kao i dobavljanje strukture sa opisom kanala na osnovu njegovog identifikatora ili deskriptora. Kako bi se pokrenula obrada poruka pristiglih na kanalu potrebno je startovati kanal. Zaustavljanjem kanala brišu se sve neobrađene poruke i oslobađaju resursi pridruženi kanalu, tj. dolazi do njegovog brisanja. Životni vek kanala je prikazan na sledećem dijagramu toka.



Slika 19. Životni vek kanala

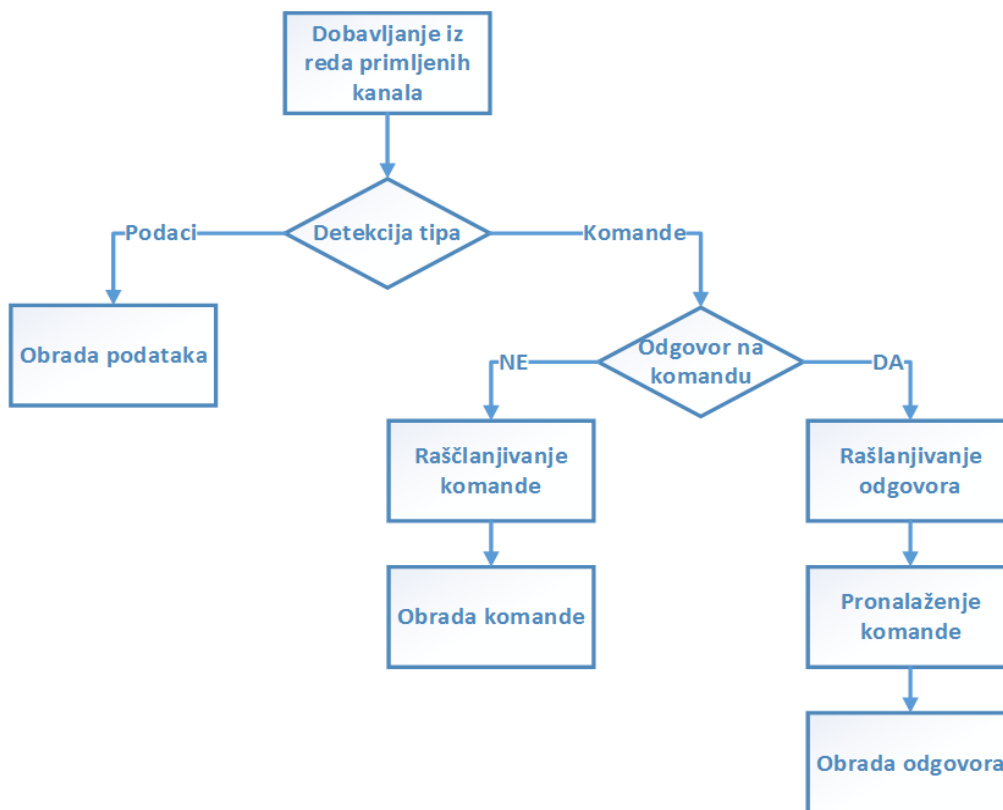
Red za obradu poruka sinhronizovan je uslovnom varijablom (engl. condition variable) koja sprečava neprestanu proveru dužine reda. Kada se u red ubaci nova poruka nit se odblokira i pokreće se obrada poruke i slanje odgovora. Poruka se obrađuje na dva nivoa komunikacionom i deskriptivnom. Na komunikacionom nivou obavljaju se akcije koje se tiču TCP kanala, dok se na deskriptivnom nivou obrađuju informacije viših nivoa apstrakcije (podržane grafičke operacije, rezolucije dekodovanja, različite vrste korisničkih događaja, itd.). Nit vrši obradu poruka sve dok u redu postoje neobrađene poruke. Zaustavljanjem kanala zaustavlja se i nit za obradu.

4.1.3.4 Modul za rukovanje RVU-RUI porukama

RVU-RUI poruke mogu se podeliti na komande i podatke. Vrsta poruka koja se šalje preko kanala određena je vrednošću njenog prvog bajta. U zavisnosti od tipa poruke vrši se propagacija do odgovarajućeg modula za obradu. RVU-RUI klijent može poslati podatke RVU-RUI poslužitelju ukoliko mu on izda adekvatnu komandu. Ovaj proces može se sprovesti u cilju

provere uspešno primljenih podataka na klijentu (poslužitelj šalje podatke do klijenta, nakon čega vrši njihovo čitanje), mada se u opštem slučaju ne koristi.

Obrada poruka primljenih na nekom od otvorenih kanala vrši se putem modula za rukovanje RVU-RUI porukama. Proces obrade poruke je prikazan na sledećem dijagramu toka.



Slika 20. Proces obrade podataka

Po prijemu poruke proverava se njen tip, nakon čega se utvrđuje da li je ona stigla po odgovarajućem tipu kanala (komande se mogu slati samo komandnim kanalom, i obrnuto). Ukoliko je ovaj uslov zadovoljen podaci odnosno poruke se prosleđuju odgovarajućem modulu na obradu. Komandne poruke se raščlanjuju kako bi se utvrdilo da li se radi o odgovoru na poruku ili o komandi izdatoj od strane RVU-RUI klijenta. Komanda za koju je stigao odgovor se utvrđuje na osnovu identifikatora poruke, nakon čega se vrši dalje raščlanjivanje podataka sadržanih u odgovoru. Sa druge strane komande poslate od strane klijenta se takođe raščlanjuju na parametre. Nakon što se izvrši analiza komande, odnosno odgovora na komandu parametri se šalju do komunikacionog modula kako bi se izvršile neophodne akcije na komunikacionom nivou. Komunikacioni modul pre obrade komande, odnosno odgovora propagira dobijene podatke do centralnog modula, putem povratne funkcije u cilju obrade na višem nivou apstrakcije, nakon čega sam sprovodi neophodne akcije. Kada centralni i komunikacioni modul obrade poruku dolazi do njenog brisanja.

Korištenjem modula za rukovanje RVU-RUI porukama omogućeno je generisanje komandi, odgovora na komande kao i poruka sa podacima. Kao ulazni argumenti prosleđuju se tip komande i neophodni parametri, nakon čega se generiše komanda ili odgovor u formatu XML niza znakova (engl. string) definisanog u konceptu rešenja. Na podatke koje je potrebno poslati dodaje se zaglavlje, koje je takođe definisano u konceptu rešenja. Svaka komanda predstavljena je strukturom *rvu_rui_command_hnd_t* putem koje se omogućuje rukovanje sa njom. Ova struktura sadrži:

- ime komande
- pokazivače na sledeće funkcije:
 - funkcija za raščlanjivanje parametara komande
 - funkcija za raščlanjivanje parametara odgovora komande
 - funkcija za generisanje komande
 - funkcija za generisanje odgovora na komandu

4.1.3.5 Modul za rukovanje RVU-RUI klijentima

Modul za rukovanje RVU-RUI klijentima omogućuje inicijalizaciju i deinicijalizaciju klijenta, zauzimanje i oslobađanje deskriptora klijenta. Nakon povezivanja novog klijenta vrši se zauzimanje novog deskriptora i njegova inicijalizacija. U slučaju raskidanja veze obavlja se obrnuti proces.

Putem ovog modula moguće je dodeljivanje i uklanjanje kanala klijentu, kao i pretraga dostupnih komandnih kanala i kanala za podatke. Identifikaciju klijenta, tj. dobavljanje klijenta moguće je izvršiti na osnovu identifikatora klijenta i njegove IP adrese.

4.1.3.6 Modul za rukovanje baferima RVU-RUI klijenta

Podaci se na klijentu čuvaju u baferima, pri čemu u zavisnosti od tipa podataka koji se čuvaju u njima zavisi i tip samog bafera. Podržani tipovi bafera definisani su u konceptu rešenja RVU-RUI poslužitelja. Bafer za prikaz i video bafer se zauzimaju na RVU-RUI klijentu pre uspostavljanja veze sa RVU-RUI poslužiteljem. Ova dva bafera se zauzimaju statički na samom početku zbog toga što su potrebni tokom celokupne RVU-RUI sesije. U toku sesije može postojati samo jedan bafer za prikaz, čiji sadržaj se prikazuje korisniku. U zavisnosti od potreba moguće je zauzeti dodatne video, grafičke i audio bafere. Komandom za dobavljanje informacija o memorijskoj slici RVU-RUI klijenta u svakom trenutku moguće je dobiti informacije o broju i vrsti zauzetih bafera. Modul za rukovanje bafera omogućuje zauzimanje i oslobađanje bafera,

kao i pretragu na osnovu identifikatora bafera. Svaki bafer predstavljen je strukturom *rvu_rui_buffer_t*:

- identifikator bafera
- tip bafera
- identifikator “AVTransport” sesije. Koristi se samo kod video bafera kako bi se omogućilo pozicioniranje video bafera unutar GKS-e.
- lista bafer brava (engl. lock) putem kojih se osigurava pravilno izvršavanje komandi

4.1.3.7 Modul za rukovanje komandama

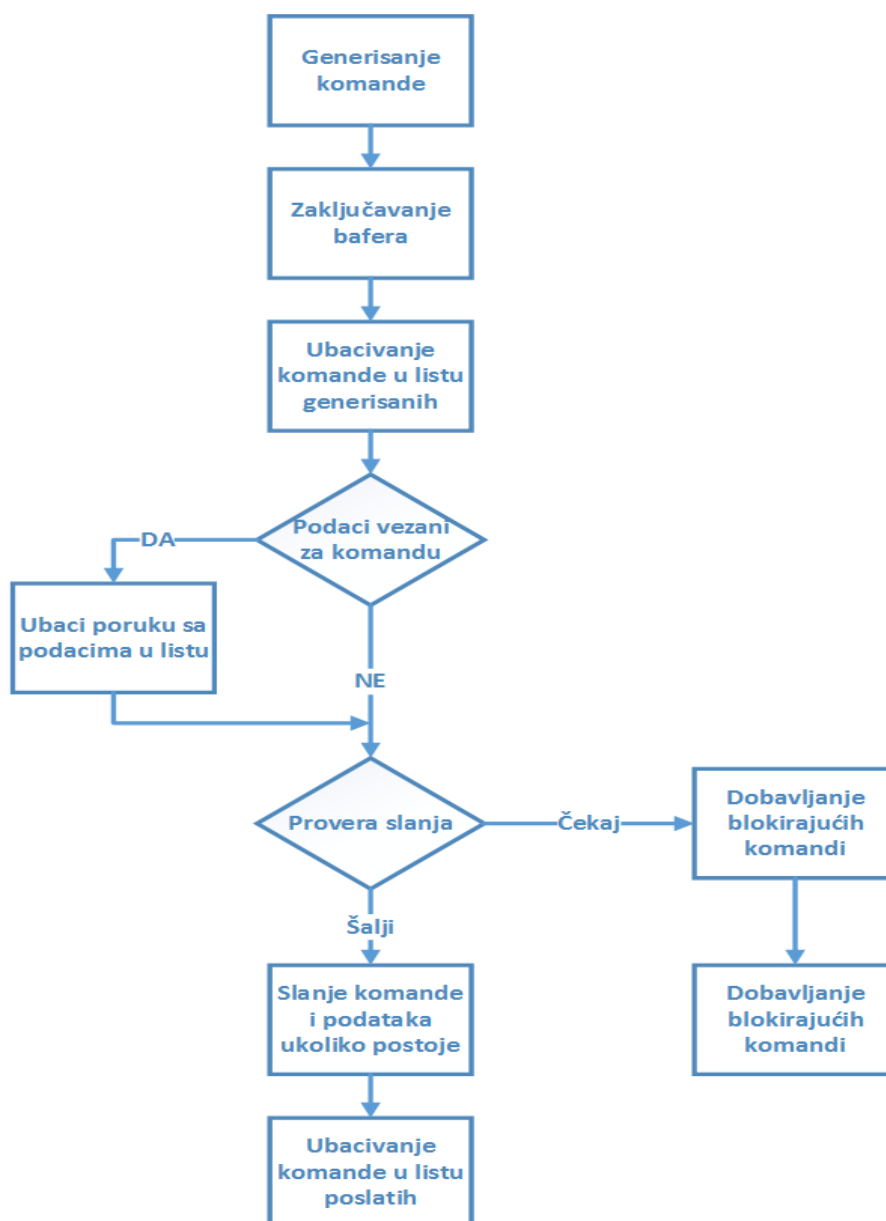
Modul za rukovanje komandama omogućuje dobavljanje identifikatora i proizvoljnih parametara komande. Putem identifikatora moguće je izvršiti pretragu postojećih komandi na kanalu. Najvažnija funkcija ovog modula je osiguravanje pravilnog izvršavanja komandi. Primer ovakvog scenarija je slanje podataka u neki bafer nakon kojeg sledi komanda koja kopira poslati sadržaj u neki drugi bafer. Kako bi se obezbedilo pravilno funkcionisanje potrebno je da druga komanda sačeka da se podaci smeste u bafer i tek onda izvrši, jer će u suprotnom slučaju doći do kopiranja starih odnosno neželjenih podataka. Svaka RVU-RUI komanda predstavljena je strukturom *rvu_rui_command_t*:

- identifikator komande (engl. token)
- tip komande (jedna od komandi iz priloženih tabela)
- bafer koji sadrži niz znakova koji reprezentuje komandu
- dužina bafera
- lista bafer brava
- lista blokiranih komandi

Bafer brava predstavlja strukturu putem koje se spreže bafer i komanda koja rukuje datim baferom. Korištenjem ovih brava omogućeno je zaključavanje i otključavanje bafera od strane komandi čime se postiže njihovo pravilno izvršavanje. Struktura *rvu_rui_buffer_lock_t*:

- pokazivač na bafer (*rvu_rui_buffer_t**)
- pokazivač na komandu (*rvu_rui_command_t**)
- prava čitanja i pisanja u bafer

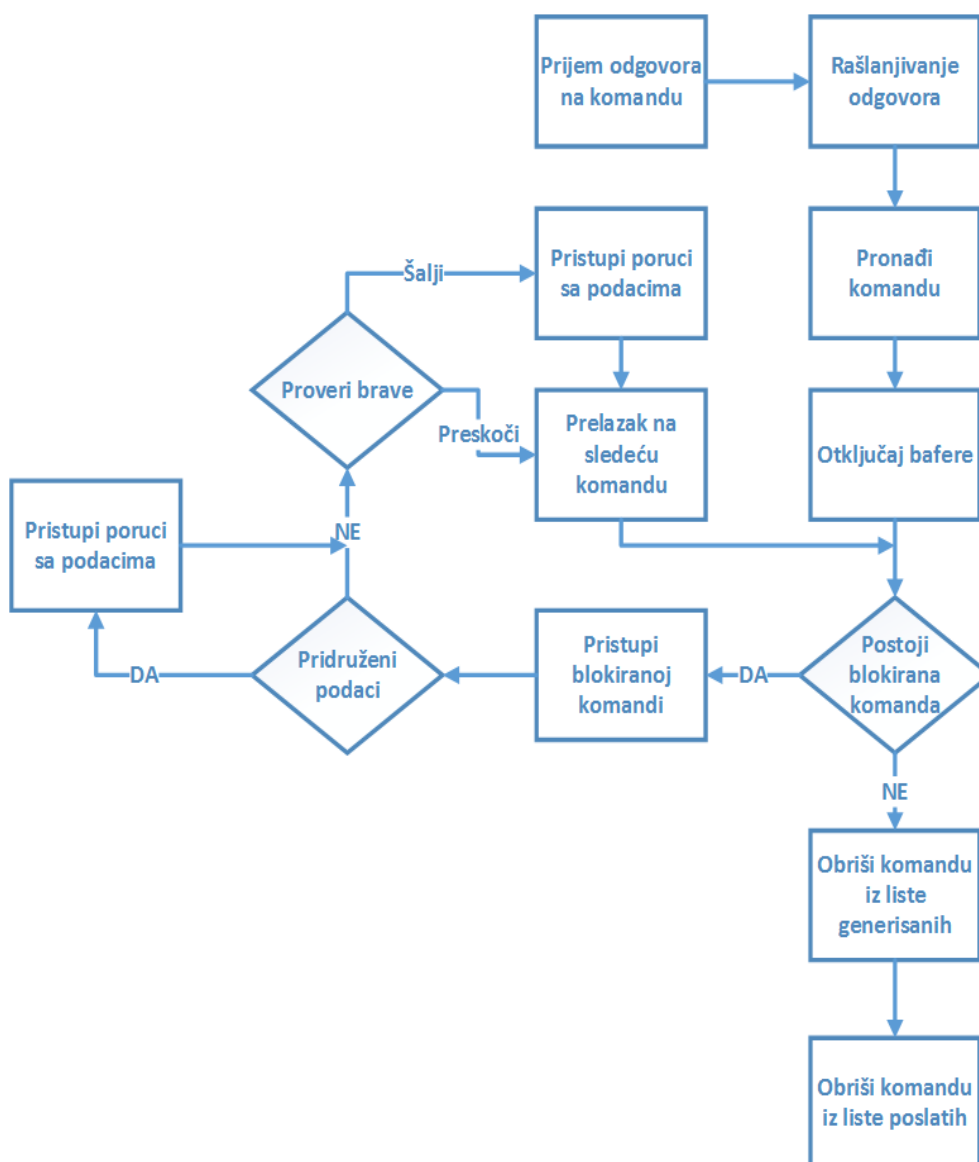
Procedura zaključavanja bafera prikazana je na sledećoj slici:



Slika 21. Algoritam zaključavanja bafera

Prvi korak prilikom slanja komande je njeno generisanje, nakon čega se obavlja identifikacija bafera koji se nalaze u parametrima komande i njihovo zaključavanje. Komanda se zatim ubacuje u listu generisanih komandi na kanalu. Ukoliko je za komandu vezana poruka sa podacima ona se ubacuje u listu generisanih poruka na kanalu za podatke. Sledeći korak podrazumeva proveru slanja, odnosno proverava se da li su baferi zaključani od strane neke druge komande, ukoliko jesu dobavljaju se blokirajuće komande. Za svaku od blokirajućih komandi pristupa se njenoj listu blokiranih komandi i u nju se ubacuje generisana komanda.

Sa druge strane ukoliko baferi nisu blokirani komanda se šalje RVU klijentu, nakon čega se ubacuje u listu poslatih komandi na kanalu. Ukoliko su komandi bili pridruženi podaci obavlja se njihovo slanje nakon što je izvršeno slanje komande.



Slika 22. Algoritam otključavanja bafera

Proces otključavanja bafera započinje prijemom odgovora na komandu nakon čega se vrši njegovo raščlanjivanje i pronalaženje komande za koju je poslat odgovor. Baferi koji su bili pridruženi komandi se otključavaju i proverava se da li je bilo blokiranih komandi. Ukoliko je neka komanda bila blokirana ona se dobavlja preko liste blokiranih komandi. Nakon toga proverava se da li su njoj bili pridruženi podaci. Proveravaju se baferi koji su pridruženi blokiranoj komandi, ova provera vrši se putem brava koje zaključavaju bafere. Ukoliko ona više nije blokirana komanda i njoj pridruženi podaci, ako postoje, se šalju putem adekvatnog kanala. Kada se obradi ova komanda prelazi se na sledeću sve dok se ne dođe do kraja liste blokiranih komandi. Nakon čega se komanda za koju je poslat odgovor briše iz liste generisanih i poslatih komandi.

4.1.3.8 Centralni modul

Centralni modul (jezgro RVU-RUI poslužitelja) koristi sve prethodno opisane module kako bi realizovao funkcionalnost RVU-RUI poslužitelja definisanu specifikacijom. Realizacija je izvršena prema definisanom konceptu rešenja u kojem ovaj modul čine četiri podmodula.

Generalni podmodul omogućuje dodavanje i uklanjanje RVU-RUI klijenta. Kako bi se omogućilo korektno upravljanje klijentima ovaj podmodul zadužen je za prikupljanje informacija o klijentu i praćenju njegovog stanja. Praćenje stanja poslužitelja, njegova inicijalizacija i deinicijalizacija su takođe u nadležnosti ovog podmodula. RVU-RUI poslužitelj opisan je strukturom *rvu_rui_server_desc_t*:

- stanje poslužitelja
- lista povezanih klijenata
- broj povezanih klijenata

Deskriptivne informacije o RVU-RUI klijentu predstavljene su strukturom *rvu_rui_client_desc_t*:

- identifikator klijenta
- lista zauzetih bafera
- količina dostupne i zauzete grafičke memorije
- količina dostupne i zauzete memorije za audio podatke
- količina dostupne i zauzete deljene memorije
- lista događaja koju klijent može generisati
- podržane grafičke operacije
- podržane rezolucije dekodovanja i prikaza videa
- podrška mozaik prikaza videa (istovremenog dekodovanja više video tokova)
- podrška za 3D video
- Z lista klijenta
- trenutna rezolucija prikaza
- broj audio dekodera
- konfiguracije video i audio bafera

Generalni podmodul postavlja povratnu funkciju komunikacionom modulu putem koje dobavlja sve podatke koji se razmene između poslužitelja i klijenta. Ova povratna funkcija omogućuje:

- dodavanje i uklanjanje RVU-RUI klijenta
- obradu događaja dobijenih od RVU-RUI klijenta
- detekciju promene rezolucije dekodovanja videa

- detekciju promene rezolucije ekrana RVU-RUI klijenta
- obradu odgovora na komande

Grafički modul omogućuje slanje podataka koji predstavljaju komponente GKS-e, kao i komandi koje rukuju na njima. Omogućeno je prevođenje kanala između stanja u kojem se grafičke komande trenutno izvršavaju i stanja u kojem se one prikupljaju kako bi se onda sve odjednom izvršile. Dva stanja kanala definisana su RVU-RUI specifikacijom i omogućuje veću slobodu u kreiranju GKS-e na RVU-RUI klijentu.

Audio modul podržava slanje i reprodukciju audio podataka na klijentu, kao i upravljanje audio dekomerima. Video modul omogućuje pozicioniranje i konfigurisanje prikaza videa čijim prenosom i reprodukcijom se upravlja putem “AVTransport” sesije.

Centralni modul pruža APS-u višim nivoima i predstavlja podskup funkcija APS-e RVU poslužitelja. Važno je napomenuti da je omogućeno podešavanje povratne funkcije centralnom modulu putem koje on obaveštava korisnike biblioteke o generisanim događajima (dodavanje i uklanjanje klijenta, kao i propagacija korisničkih događaja).

4.1.3.9 Aplikativna programska sprega RVU-RUI poslužitelja

Prikaz struktura APS-e RVU-RUI poslužitelja:

Argumenti komandi stapanja *dlna_rvu_rui_blit_command_args_t*:

- identifikator prvog i drugog izvorišnog bafera
- x i y pozicija unutar izvorišnih bafera
- dužina i širina bloka unutar izvorišnih bafera
- identifikator odredišnog bafera
- x i y pozicija unutar odredišnog bafera
- dužina i širina bloka unutar odredišnog bafera
- boja
- pravilno zatamnjanja (engl. shade rule)

Konfiguracija video bafera *dlna_rvu_rui_video_config_args_t*:

- x i y pozicija video prozora unutar GKS-e
- širina i visina video prozora

Informacije o grafičkim baferima *dlna_rvu_rui_graphics_info_t*:

- broj dostupnih i zauzetih grafičkih bafera

Informacije o audio baferima *dlna_rvu_rui_audio_info_t*:

- broj dostupnih i zauzetih audio bafera

Funkcije:

***dlna_error_t dlna_rvu_rui_get_client_graphics_info(uint8_t id,
dlna_rvu_rui_graphics_info_t* info);***

Dobavljanje informacija o grafičkim baferima na RVU-RUI klijentu.

Parametri:

- id - identifikator RVU-RUI klijenta
- info - informacije o grafičkim baferima

***dlna_error_t dlna_rvu_rui_get_client_audio_info(uint8_t id,
dlna_rvu_rui_graphics_info_t* info);***

Dobavljanje informacija o audio baferima na RVU-RUI klijentu.

Parametri:

- id - identifikator RVU-RUI klijenta
- info - informacije o audio baferima

***dlna_error_t dlna_rvu_rui_get_client_buffer(uint8_t id, uint8_t buff_index, uint32_t*
buff_handle);***

Dobavljanje identifikatora bafera na osnovu njegovog indeksa dobavljenog putem prethodne dve operacije.

Parametri:

- id - identifikator RVU-RUI klijenta
- buff_index - indeks bafera
- buff_handle - identifikator bafera

dlna_error_t dlna_rvu_rui_queue_commands(uint8_t id, uint8_t flag);

Prevođenje kanala između stanja u kojem se komande sakupljaju, odnosno stanja u kojem se trenutno izvršavaju.

Parametri:

- id - identifikator RVU-RUI klijenta
- flag - broj različit od 0 za odloženo izvršavanje odnosno 0 za trenutno

***dlna_error_t dlna_rvu_rui_send_blit_command(uint8_t id,
dlna_rvu_rui_blit_command_type_t blit_type,
dlna_rvu_rui_blit_command_args_t blit_args);***

Slanje različitih komandi stapanja (engl. “blit”) koje omogućuje formiranje GKS-e RVU klijenta.

Parametri:

- id - identifikator RVU-RUI klijenta
- blit_type - tip komande stapanja (podržane komande navedene su u tabeli 3.)
- blit_args - argumenti komande

dlua_error_t dlua_rvu_rui_send_picture(uint8_t id, uint32_t buff_handle, uint8_t* picture_buff, uint32_t picture_size);

Slanje komande za upis i slanje grafičkih podataka koji predstavljaju komponentu GKS-e.

Parametri:

- id - identifikator RVU-RUI klijenta
- buff_handle - identifikator bafera
- picture_buff - bafer sa grafičkim podacima
- picture_size - veličina bafera

dlua_error_t dlua_rvu_rui_send_audio(uint8_t id, uint8_t buff_handle, uint8_t* audio_buff, uint32_t audio_size);

Slanje komande za upis i slanje audio podataka koji će se reprodukovati unutar GKS-e.

Parametri:

- id - identifikator RVU-RUI klijenta
- buff_handle - identifikator bafera
- audio_buff - bafer sa audio podacima
- audio_size - veličina bafera

dlua_error_t dlua_rvu_rui_play_audio(uint8_t id, uint8_t buff_handle);

Startovanje reprodukcije audio podataka.

Parametri:

- id - identifikator RVU-RUI klijenta
- buff_handle - identifikator audio bafera u kojem se nalaze podaci

dlua_error_t dlua_rvu_rui_stop_audio(uint8_t id, uint8_t buff_handle);

Zaustavljanje reprodukcije audio podataka.

Parametri:

- id - identifikator RVU-RUI klijenta
- buff_handle - identifikator audio bafera u kojem se nalaze podaci

dlna_error_t dlna_rvu_rui_configure_video(uint8_t id, int32_t avt_id, uint8_t fullscreen, dlna_rvu_rui_video_config_args_t video_config_args);

Konfiguracija video prozora unutar GKS-e RVU-RUI klijenta.

Parametri:

- id - identifikator RVU-RUI klijenta
- avt_id - identifikator "AVTransport" sesije
- fullscreen - prikaz video preko celog ekrana
- video_config_args - u slučaju da se video ne prikazuje preko celog ekrana, pozicija video prozora

4.1.3.10 Interakcija izmedju gradivnih komponenti RVU-RUI poslužitelja

Na sledećih nekoliko dijagrama sekvenci prikazani su karakteristični slučajevi korištenja (engl.use-case).

Proces inicijalizacija podrazumeva iniciranje uspostave sesije od strane RVU klijenta. Nakon što se klijent poveže na poslužitelja zauzima se deskriptor za tog klijenta, kao i prvi kanal po kome klijent šalje "Hello" komandu poruku, usled čega se taj kanal identifikuje kao komandi. Poslužitelj zauzima dodatni kanal i šalje "Hello data" poruku klijentu, odnosno zauzima kanal za podatke.

Sledeća faza proces inicijalizacije podrazumeva dobavljanje informacija o klijentu. Dobavljaju se sledeći podaci:

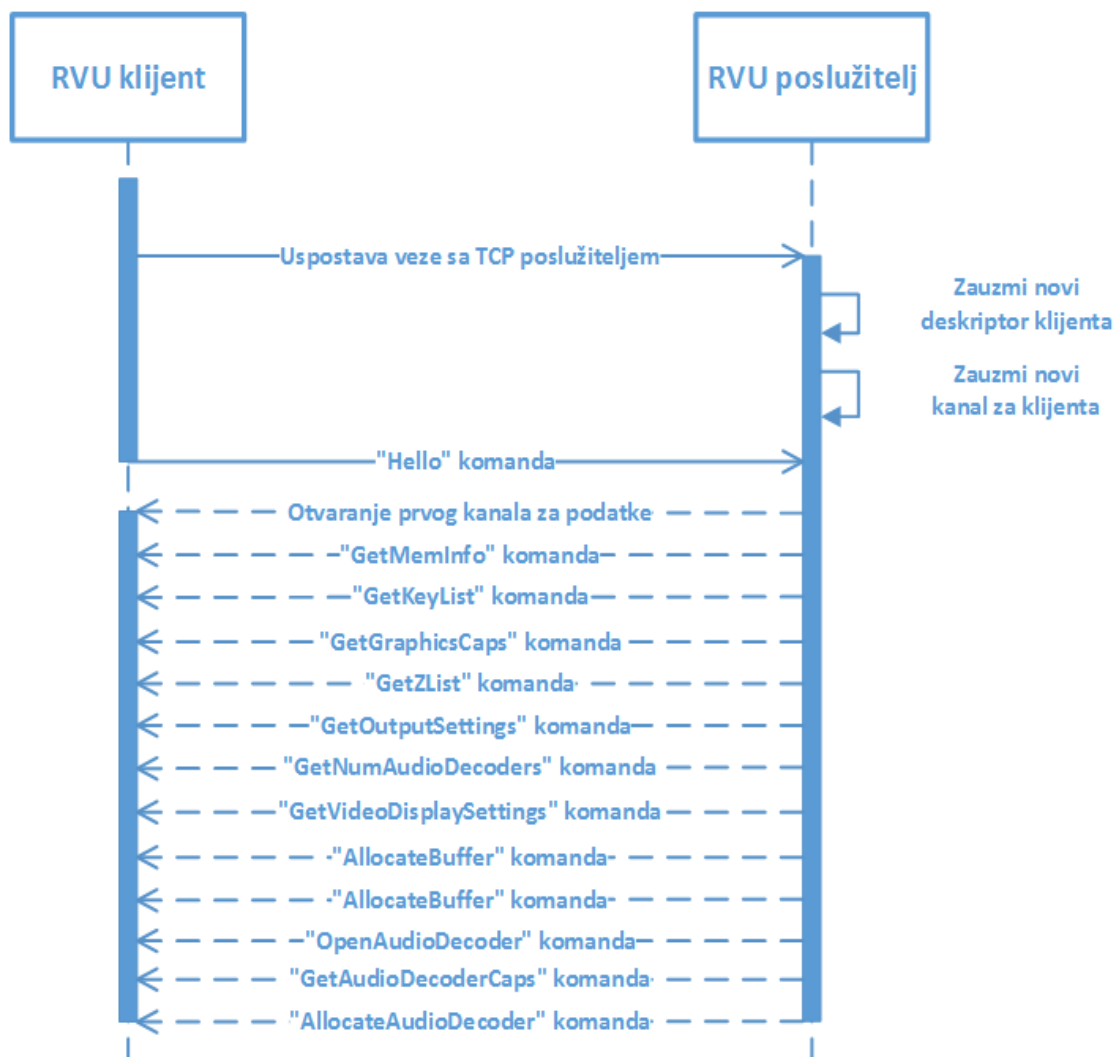
- memorijska slika klijenta
- lista događaja koje klijent može generisati
- podržane grafičke operacije
- Z lista
- konfiguracija ekrana
- broj audio dekodera

Kada se prime odgovori na ove komande dobavljaju se sledeće informacije:

- konfiguracija video bafera
- mogućnosti audio dekodera

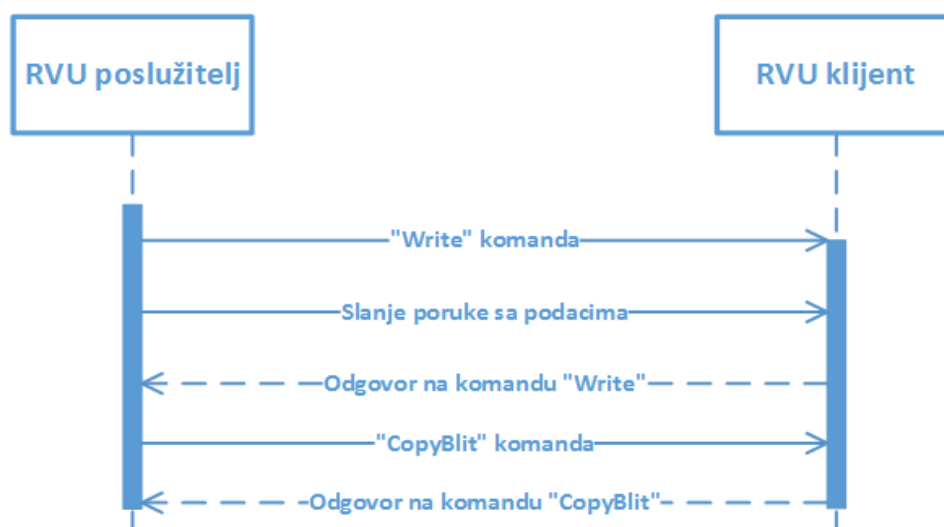
Analizira se memorijska slika klijenta i ukoliko je moguće na inicijalizaciji se zauzimaju dva grafička bafera, koja će se koristiti za formiranje GKS-e. Proverava se broj audio dekodera i

otvara se jedan od njih, ako postoje. Na kraju se zauzima jedan audio bafer u koji će se smeštati audio podaci. Važno je napomenuti da je na sledećem dijagramu sekvenci izostavljen prijem odgovora, kako bi se ostvarila bolja preglednost.



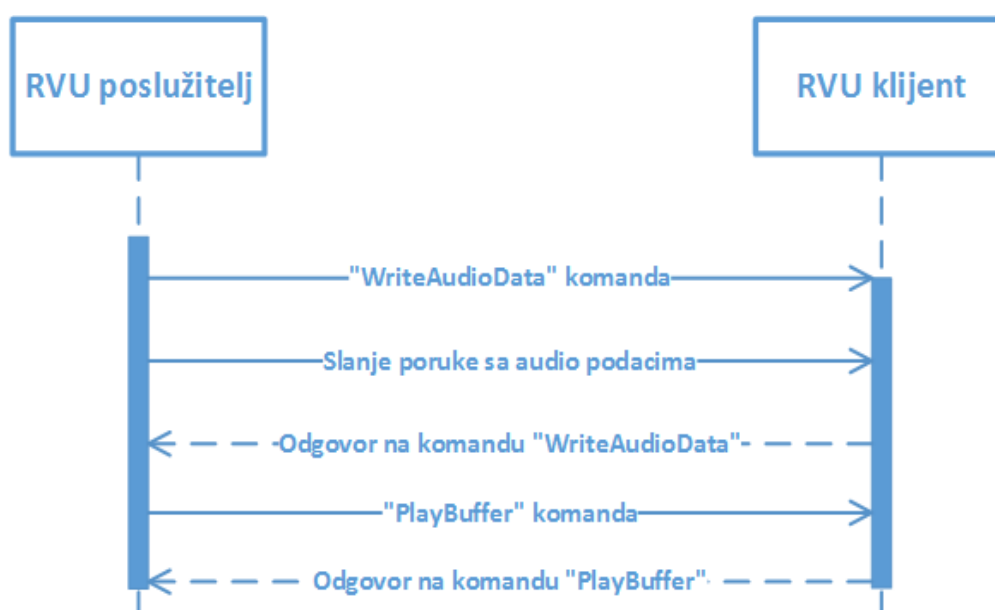
Slika 23. Inicijalizacija sesije i konfiguracija klijenta

Na sledećem dijagramu sekvence opisan je karakterističan proces slanja grafičkih podataka i komande koja rukuje tim podacima. Prvo se šalje "Write" komanda koja određuje po kom kanalu za podatke će podaci stići i u koji grafički bafer je potrebno smestiti te podatke. Nakon toga se podaci šalju po specificiranom kanalu za podatke. Kada podaci stignu na odredište RVU klijent generiše odgovor na komandu. Nakon toga poslužitelj može da pošalje neku grafičku komandu koja će rukovati tim podacima pri čemu je na ovom dijagramu poslata komanda koja će te podatke kopirati u drugi grafički bafer. RVU klijent po prijemu komande izvršava ovu operaciju i šalje odgovor, koji sadrži indicaciju uspešnosti operacije.



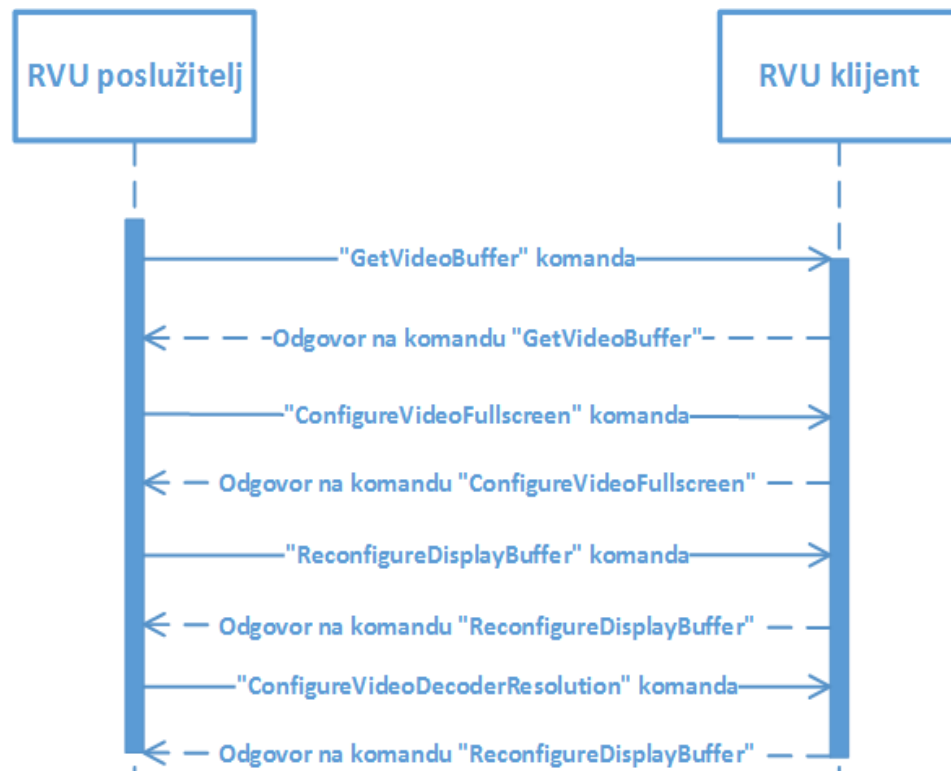
Slika 24. Prikaz karakteristične sekvence grafičkih komandi

Analogno slanju grafičkih komandi, obavlja se slanje audio komandi. Prvo se šalje komanda koja naznačava na kom kanalu za podatke će se preneti audio podaci i određuje audio bafer u koji treba smestiti podatke. Nakon prijema odgovora šalje se komanda "PlayBuffer" koja određuje bafer u kojem se nalaze primljeni audio podaci i započinje njihovu reprodukciju.



Slika 25. Prikaz karakteristične sekvence audio komandi

Poslednji dijagram sekvenci opisuje konfiguraciju video prozora unutar kojeg će se prikazati video čijom reprodukcijom se upravlja putem "AVTransport" sesije. Prva komanda "GetVideoBuffer" omogućuje povezivanje identifikatora "AVTransport" sesije i odgovarajućeg video bafera, putem kojeg se vrši podešavanje video prozora. Nakon što je ta veza uspostavljena vrši se konfiguracija video prozora, pri čemu se ovde video prikazuje preko celog ekrana. Zatim se rekonfiguriše bafer za prikaz kako bi se sakrio sa ekrana, i na kraju se podešava rezolucija dekodovanja videa.



Slika 26. Prikaz konfiguracije video prozora

4.2 Qt dodaci

Realizacija Qt dodataka prikazana je putem dva UML dijagrama klasa, koji opisuju strukturu rešenja. Nakon opisa strukture opisana je interakcija između komponenti ovih dodataka, kao i njihova sprega sa ostatkom sistema.

4.2.1 Qt RVU-RUI dodatak

Struktura Qt RVU-RUI dodatka prikazana je na slici 27. QRemoteGuiIntegration klasa predstavlja baznu klasu dodatka putem koje se pristupa RVU-RUI poslužitelju. Objekat bazne klase se pravi prilikom pokretanja aplikacije pomoću fabričke klase QRemoteGuiIntegrationPlugin.

Konstruktor QRemoteGuiIntegration klase kreira sledeće objekte:

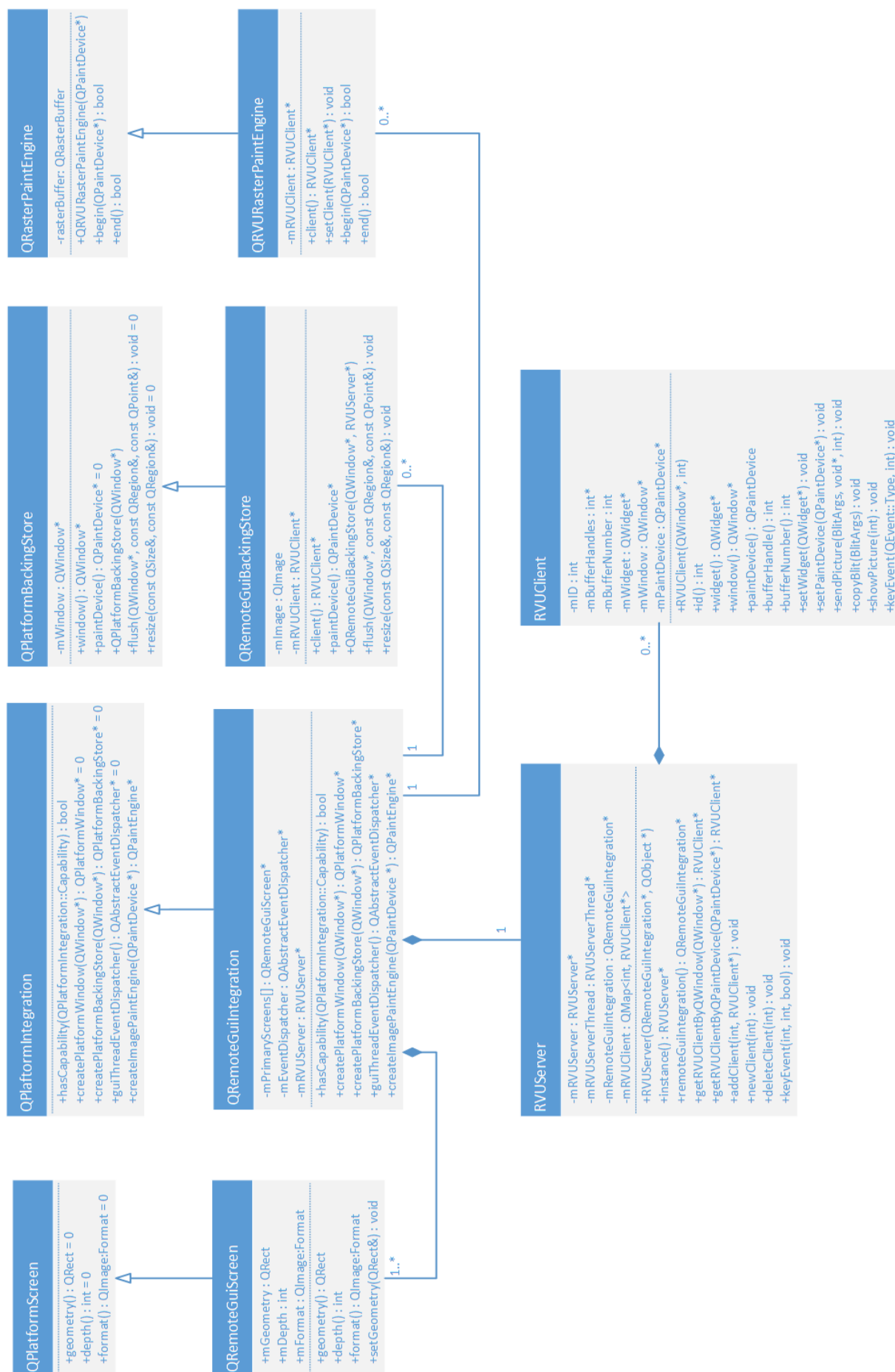
- RVU Server kojim se vrši apstrakcija RVU poslužitelja.
- QAbstractEventDispatcher predstavlja raspoređivač putem kojeg se vrši propagacija događaja generisanih od strane RVU klijenata do Qt radnog okvira.

Metode klase QRemoteGuiIntegration omogućuju Qt radnom okviru kreiranje objekata putem kojih će pristupati RVU poslužitelju, odnosno putem kojih će upravljati RVU klijentima:

- createPlatformWindow metoda omogućuje kreiranje glavnog prozora za RVU klijenta. Na osnovu ovog prozora kasnije je moguće identifikovati RVU klijenta.
- createPlatformBackingStore metoda omogućuje kreiranje pozadinskog skladišta uz pomoć kojeg će se iscrtavati GKS-a na RVU klijentu.
- createImagePaintEngine metoda omogućuje kreiranje modula za crtanje uz pomoć kojeg će se vršiti ažuriranje pojedinačnih komponenti GKS-e na RVU klijentu.

Klasa QRemoteGuiScreen reprezentuje ekran RVU klijenta, prilikom dodavanja novog klijenta, objekat ove klase se ubacuje u niz mPrimaryScreen QRemoteGuiIntegration klase. Korišćenjem metode setGeometry ove klase moguće je podesiti veličinu ekrana RVU klijenta unutar Qt radnog okvira.

Klasa QRemoteGuiBackingStore apstrahuje sadržaj prozora koji je potrebno iscrtati na RVU klijentu. Kako svaki klijent poseduje po jedan glavni prozor iz toga sledi da svaki klijent poseduje i po jedan objekat klase QRemoteGuiBackingStore. Objektu ove klase pridružen je objekat QRVURasterPaintEngine uz pomoć kojeg se vrši ažuriranje sadržaja prozora. U metodi begin klase QRVURasterPaintEngine se zabeleži koji deo prozora je potrebno ažurirati. Kada se pozove metoda end ažuriranje tog dela prozora je završeno, pri čemu se sadržaj nalazi u rasterBuffer-u. Sadržaj ovog bafera se enkoduje u PNG sliku prenosi do RVU klijenta i stapa (engl. blit), na odgovarajuću poziciju u grafičkom baferu.



Slika 27. Dijagram klasa Qt RVU-RUI dodatka

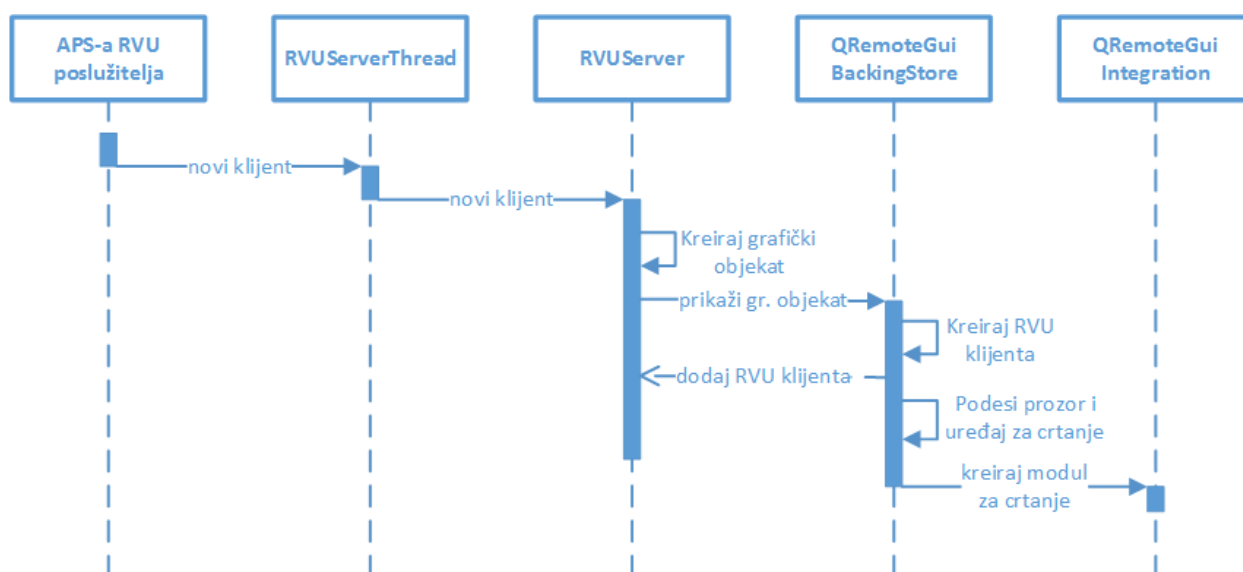
Nakon što su ažurirane sve komponente GKS-e poziva se metoda flush QRemoteGuiBackingStore klase, koja izvrši zamenu bafera za prikaz i grafičkog bafera u kojem je ažurirana GKS-a. Na ovaj način postiže se efekat dvostrukog baferovanja i izbegava treperenje na strani klijenta.

Konstruktor RVU Server klase kreira RVU poslužitelja. Metodama klase RVU Server vrši se sprega sa APS-om RVU poslužitelja. Ova klasa pored kreiranja i uništavanja poslužitelja omogućuje i detekciju dodavanja i uklanjanja klijenta, kao i događaja generisanih na strani RVU klijenta. Sinhronizacija događaja obavlja se putem klase RVU Server Thread koja osigurava sekvencijalnu isporuku događaja. Proces dodavanja klijenta unutar Qt RVU-RUI dodatka prikazan je na slici 28. Događaji se propagiraju do Qt radnog okvira putem raspoređivača QRemoteGuiIntegration klase. Putem klase RVU Server omogućena je i identifikacija klijenta na osnovu prozora i modula za crtanje koji su mu dodeljeni.

Klasa RVUClient apstrahuje RVU klijenta. Objekti ove klase se kreiraju nakon detekcije događaja novog klijenta. Prilikom kreiranja klijenta dodeljuju mu se objekti sledećih klasa:

- glavni grafički objekat (engl. main widget) koji predstavlja aplikaciju koja se iscertava na RVU klijentu
- QPlatformWindow predstavlja prozor unutar kojeg se iscertava sadržaj aplikacije
- QPaintDevice uz pomoć kojeg se vrši sprega klijenta sa QRVURasterPaintEngine-om

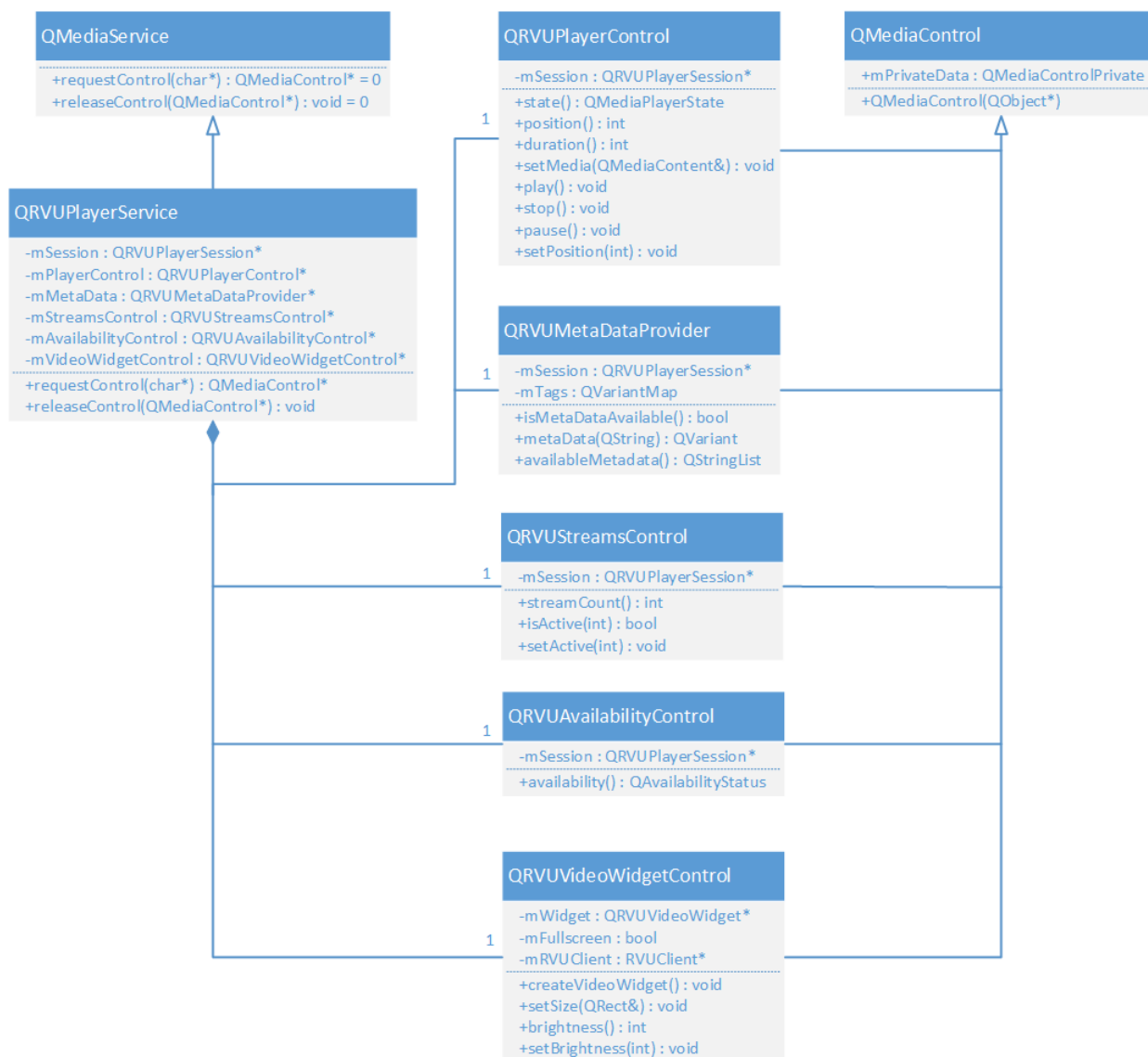
Metode klase RVUClient takođe se apstrahuje APS-a RVU poslužitelja. QRemoteGuiBackingStore i QRVURasterPaintEngine koriste metode sendPicture, copyBlit i showPicture klase RVUClient, kako bi ažurirale GKS-u RVU klijenta.



Slika 28. Proces dodavanja klijenta u Qt RVU-RUI dodatku

4.2.2 Qt DLNA medija plejer dodatak

Dijagram klasa Qt DLNA medija plejer dodatka prikazan je na slici koja sledi:



Slika 29. Dijagram klasa Qt DLNA medija plejer dodatka

Objekat klase QRVUPlayerService-a se kreira putem fabričke klase QRVUPlayerServicePlugin. Klasa QRVUPlayerService omogućuje prihvatanje i otpuštanje kontrole nad kontrolnim klasama koje realizuju funkcionalnost Qt DLNA medija plejer dodatka. Qt radni okvir koristi klasu QRVUPlayerService kako bi preuzeo željenu kontrolu i nakon toga sproveo određenu akciju nad kontrolom. Svaki tip kontrole unutar Qt radnog okvira poseduje ključ putem kojeg se vrši identifikacija kontrole unutar QRVUPlayerService klase.

QRVUPlayerSession klasa apstrahuje APS-u RVU poslužitelja, u cilju lakše realizacije kontrolnih modula. Putem ove klase se pristupa DMC-u i DMS-u koji su ugrađeni RVU poslužitelja.

QRVUPlayerControl klasa omogućuje kontrolu reprodukcije multimedijalnih datoteka na RVU klijentu. Ova kontrola se obavlja indirektno kroz QRVUPlayerSession klasu koja apstrahuje DMC. Metode QRVUPlayerControl klase se preslikavaju na adekvatne funkcije APS-e DMC-a. Prilikom postavljanja datoteke koja se želi reprodukovati QRVUPlayerSession klasa putem APS-e DMS-a pronalazi deljenu datoteku i pribavlja meta informacije dobijene iz DLNA objekta pridruženog datoteci.

QRVUMetaDataProvider klasa omogućuje dobavljanje meta informacija o datoteci koja se trenutno reprodukuje. Putem metode availableMetaData se dobija lista meta informacija koje su dostupne da bi se zatim korištenjem metode metaData dobavila vrednost određene meta informacije. QRVUMetaDataProvider klasa koristi QRVUPlayerSession klasu kako bi dobavila meta informacije.

QRVUAvailabilityControl klasa prati stanje DMR-a na RVU poslužitelju. Putem ovako dobavljenog stanja Qt radni okvir ima informaciju o mogućnosti reprodukcije multimedijalne datoteke na RVU klijentu.

QRVUStreamsControl klasa omogućuje kontrolu elementarnih tokova unutar transportnog toka koji se prenosi putem "AVTransport" sesije. Omogućeno je dobavljanje liste dostupnih tokova, kao i postavljanje nekog od njih kao aktivnog. Metode ove klase se preslikavaju na akcije DMC-a.

QRVUVideoWidget klasa omogućuje pozicioniranje video prozora unutar GKS-e klijenta. Ova klasa u sebi sadrži objekat RVUClient putem kojeg se zna na kom klijentu je potrebno obaviti podešavanje video prozora. Metoda setSize se preslikava na adekvatnu funkciju APS-e RVU-RUI poslužitelja. Pored podešavanja video prozora ovaj kontrolni modul omogućuje podešavanje osvetljaja, kontrasta, zasićenja ekrana RVU klijenta. Ove metode se preslikavaju na funkcije DMC-a koje su spregnute sa UPnP akcijama "RenderingControl" servisa.

4.3 Qt radni okvir

Qt radni okvir proširen je klasom QRemoteGUIApplication. Ova klasa nasleđuje klasu QApplication koja reprezentuje aplikaciju unutar Qt radnog okvira. QRemoteGUIApplication klasa je realizovana kako bi se kreiranjem objekta ove klase aplikacija automatski pokrenula u RVU kontekstu, odnosno prilikom pokretanja aplikacije automatski se učitavaju Qt RVU-RUI i Qt DLNA medija plejer dodaci. Kako bi se sprečilo izlaz iz aplikacije kada na RVU poslužitelja nisu povezani klijenti prilikom kreiranja objekta QRemoteGUIApplication klase kreira se pomoćni grafički objekat koji drži kontekst aplikacije u tom slučaju. Ovaj grafički objekat se ne iscertava ni na jednom od RVU klijenata. Dinamičko kreiranje proizvoljnog grafičkog objekta za

RVU klijenta omogućeno je putem fabričke klase koja se podesi objektu RVU Server klase. Na ovaj način kada objekat RVU Servera detektuje novog klijenta, putem ove fabrike kreira grafički objekat za tog klijenta. Na samom kraju važno je napomenuti da svi realizovani programski blokovi ne unose promene unutar Qt radnog okvira, već samo proširuju njegovu funkcionalnost.

5. Ispitivanje i verifikacija rešenja

Opisano programsko rešenje realizovano je na Linux i Android platformi. Verifikacija realizovanog rešenja izvršena je iz dva dela. U prvom delu ispitana je funkcionalnost RVU programske biblioteke. Drugi deo verifikacije rešenja obuhvata evaluaciju integracije RVU biblioteke u Qt radni okvir, pri čemu je ovaj proces izvršen proverom pravilnog funkcionisanja realizovanih Qt dodataka.

U cilju verifikacije funkcionalnosti RVU programske biblioteke realizovana je ispitna aplikacija koja omogućuje reprodukciju tipičnih slučajeva korištenja (engl. use-case). Svaki podmodul RVU poslužitelja testiran je nezavisno od ostatka sistema. Nakon što je potvrđeno njihovo pravilno funkcionisanje izvršeno je testiranje celokupnog sistema. Testovi su sprovedeni nad sledećim modulim RVU poslužitelja:

- DMS - Ispitna sekvenca: kreiranje i startovanje DMS, dodavanje i uklanjanje deljene datoteke, stopiranje i brisanje DMS-a.
- RUIS - Ispitna sekvenca: kreiranje i startovanje RUIS-a, dodavanje i uklanjanje deljenih GKS-a, stopiranje i brisanje RUIS-a.
- DMC - Ispitna sekvenca: detekcija dodavanja i uklanjanja DMS-a i DMR-a. Nakon toga pozivane su funkcije DMC-a koje se preslikavaju na UPnP akcije pri čemu je provera njihovog pravilnog funkcionisanja izvršena praćenjem stanja ciljanog DMS-a, odnosno DMR-a.
- RVU-RUI poslužitelja - Ispitna sekvenca: detekcija dodavanja i uklanjanja RVU-RUI klijenata. Ostatak testova grupisan je prema vrstama podržanih tipova komandi. Formirana je sekvenca grafičkih komandi koju je potrebno obaviti na RVU-RUI klijentu, pri čemu je praćenjem njegovog stanja utvrđeno pravilno

izvršavanje ovih komandi. Audio komande i komande za konfiguraciju video prozora ispitane su na isti način.

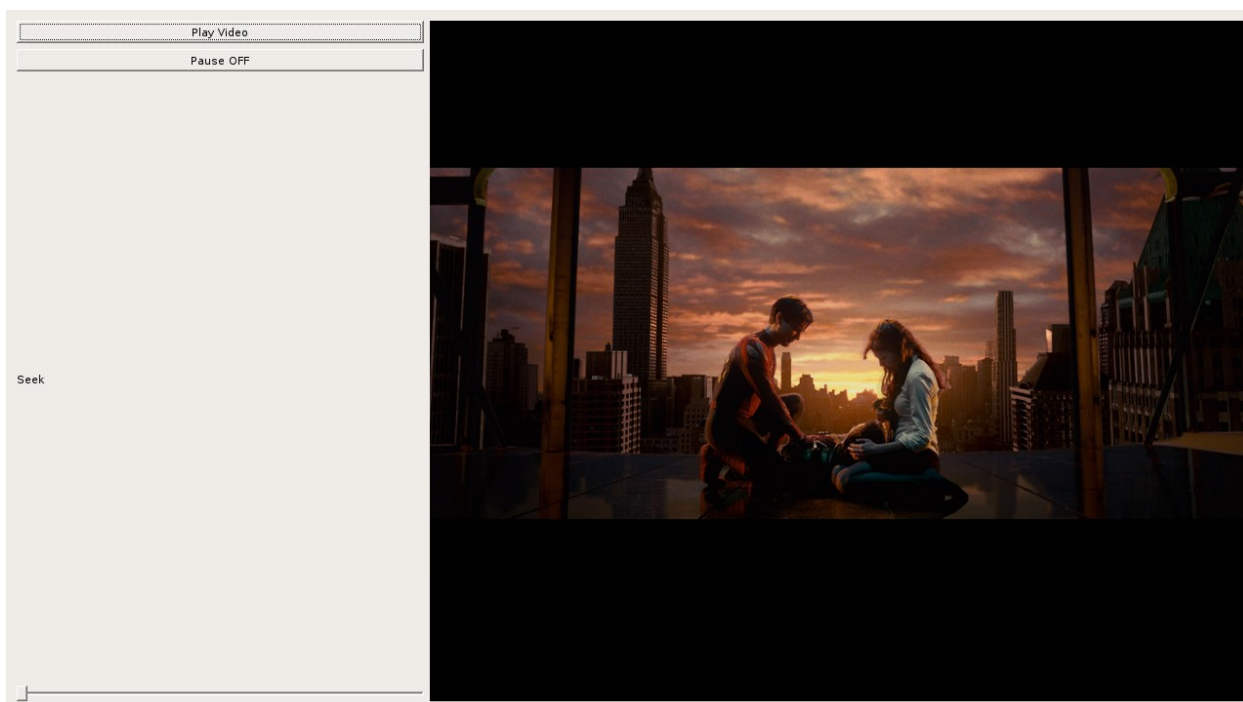
Svi ispitni slučajevi nad pojedinačnim podmodulima RVU poslužitelja uspešno su sprovedeni. Kombinovanjem prethodno opisanih testnih sekvenci izvršena je analiza RVU poslužitelja, čime je potvrđeno njegovo pravilno funkcionisanje.

Izvršena su neophodna merenja kako bi se ustanovila brzina uspostavljanja veze između RVU poslužitelja i klijenta, kao i brzina konfiguracije video prozora na RVU klijentu, odnosno brzina kojom se ostvaruje sprega između RVU-RUI i "AVTransport" sesije. Prosečna brzina uspostave veze i početne konfiguracije RVU klijenta iznosi 250ms, dok se video prozor u proseku konfiguriše za 40 ms. Praćenje ostalih parametara direktno je zavisno od izgleda i brzine ažuriranja grafičke korisničke sprege i zbog toga nije prikazano u ovoj analizi.

Verifikacija integracije RVU biblioteke u Qt radni okvir putem Qt RVU-RUI i Qt DLNA medija plejer dodataka izvršena je kreiranjem ispitne Qt aplikacije. Cilj ove ispitne aplikacije je provera:

- pravilnog iscrtavanja grafičkih komponenti Qt aplikacije na RVU klijentu
- pravilne reprodukcije multimedijalnih datoteka i pozicioniranja video prozora unutar GKS-e
- pravilne propagacije događaja generisanih na RVU klijentu unutar Qt radnog okvira

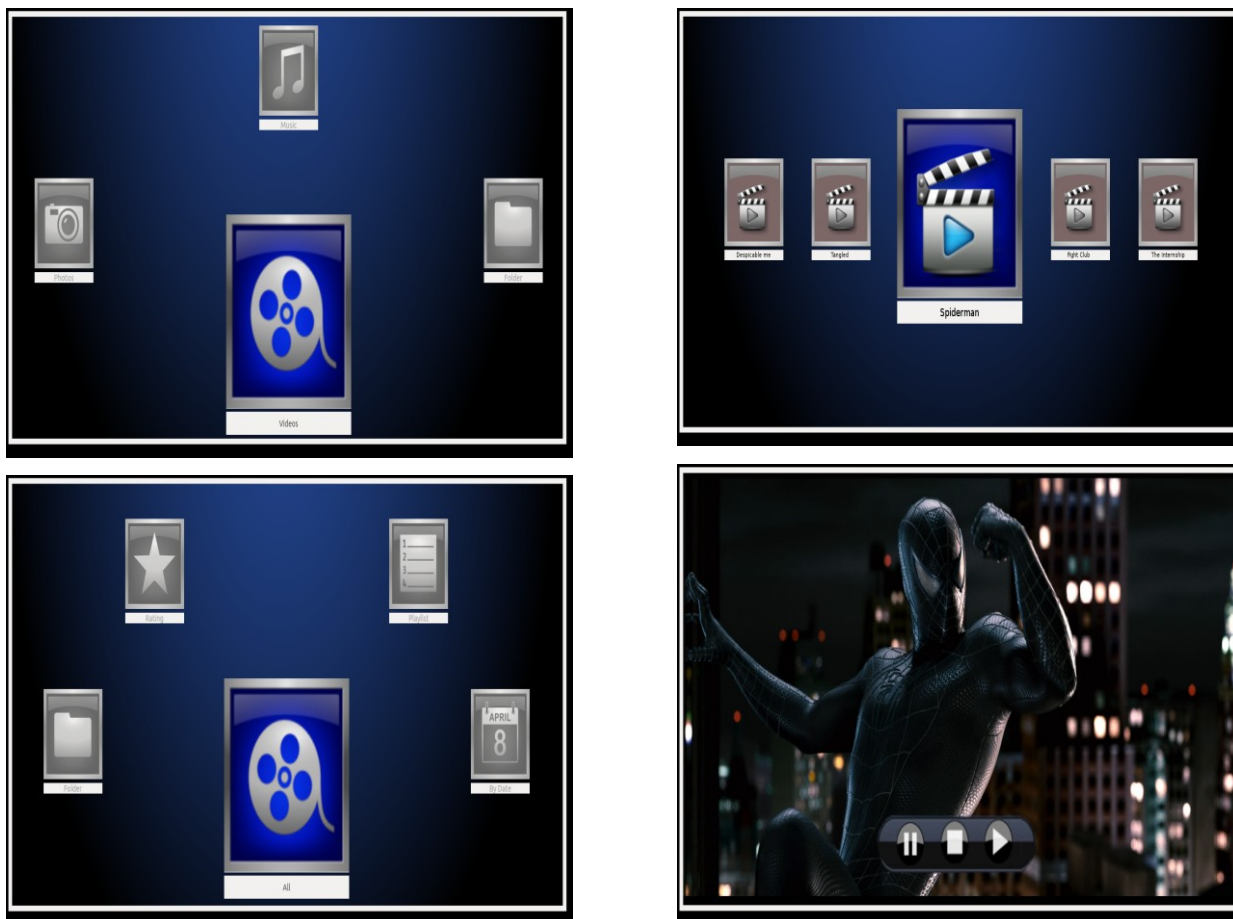
Izgled ispitne Qt aplikacije koja se iscrtava na klijentu prikazan je na sledećoj slici:



Slika 30. Prikaz ispitne Qt aplikacije

Tokom izvršavanja Qt aplikacije praćeno je stanje RVU poslužitelja i RVU klijenta,. Analizom prikupljenih podataka, tj. razmenjenih komandi i PNG slika potvrđeno je pravilno funkcionisanje sistema.

U cilju demonstracije punog potencijala realizovanog rešenja realizovana je Qt DLNA pretraživač (engl. browser) aplikacija. Putem ove aplikacije moguće je reprodukovati deljeni sadržaj sa dostupnih DMS-a na RVU klijentu. Izgled ove aplikacije prikazan je na sledećim slikama:



Slika 31. Izgled DLNA pretraživač aplikacije na RVU klijentu

6. Zaključak

U ovom radu realizovano je radno okruženje za kreiranje Qt RVU aplikacija. Aplikacija se izvršava na RVU poslužitelju, dok se njen kontekst iscertava na RVU klijentima prisutnim u okviru kućne mreže. RVU aplikacije moguće je kreirati i putem APS-e RVU poslužitelja, međutim kako bi se ovaj proces pojednostavio iskorišten je Qt radni okvir, koji je putem Qt RVU-RUI i Qt DLNA medija plejer dodataka spregnut sa RVU poslužiteljem.

Rešenje je razvijeno na Linux i Android platformi. Proces verifikacije uspešno je izvršen prema predviđenom planu. Ukoliko je potrebno rešenje je moguće brzo i jednostavno prilagoditi bilo kojoj platformi podržanoj od strane Qt radnog okvira.

U cilju prezentacije realizovanog rešenja kreirana je Qt DLNA pretraživač aplikacija koja omogućuje prikaz svih funkcionalnosti realizovanog rešenja, tj. prikazano je udaljeno iscertavanje GKS-e i kontrola reprodukcije multimedijalnih datoteka na RVU klijentu.

Dalji napredak ovog rešenja imao bi za cilj dodatnu optimizaciju iscertavanja GKS-e na RVU klijentu, čime bi se postigla veća brzina ažuriranja GKS-e. Dodatna optimizacija ostvarila bi se razvojem algoritma koji bi vodio evidenciju o prethodno realizovanim grafičkim operacijama i na taj način omogućio korišćenje šireg spektra RVU-RUI grafičkih komandi unutar Qt RVU-RUI dodatka. Ovim postupkom eliminisao bi se ponovni prenos grafičkih komponenti koje se već nalaze na RVU klijentu.

7. Literatura

- [1] RVU specifikacija , *RVU_Protocol_Specification_V1_0_Rev1_42*
- [2] Van Dang Khoa Mai; Younghun Kim, "Using DLNA cloud for sharing multimedia contents beyond home networks," *Advanced Communication Technology (ICACT), 2014 16th International Conference on* , vol., no., pp.54,57, 16-19 Feb. 2014.
- [3] Gois, J.P.; Batagelo, H.C., "Interactive Graphics Applications with OpenGL Shading Language and Qt," *Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 2012 25th SIBGRAPI Conference on* , vol., no., pp.1,20, 22-25 Aug. 2012.
- [4] Qt Platform Abstraction, www.qt-project.org/wiki/Qt-Platform-Abstraction
- [5] Seungchul Kim; Sangtaick Park; Jinwoo Hong, "GUI screen-sharing smart remote control for smart TV user interface," *ICT Convergence (ICTC), 2013 International Conference on* , vol., no., pp.711,713, 14-16 Oct. 2013.
- [6] RUIS specifikacija, *UPnP-rui-RemoteUIServerDevice-v1-Device*
- [7] Lu Yiqin; Fang Fang; Liu Wei, "Home Networking and Control Based on UPnP: An Implementation," *Computer Science and Engineering, 2009. WCSE '09. Second International Workshop on* , vol.2, no., pp.385,389, 28-30 Oct. 2009.
- [8] DLNA arhitektura i protokoli, *DLNA Guidelines Architectures and Protocols*
- [9] UPnP arhitektura uređaja, *UPnP Device Architecture version 1.0*
- [10] Juha Leppilahti, Helsinki University of Technology, "Remote UI protocols for home environment"
- [11] RVU alijansa, *RVU-White-Paper*
- [12] DLNA medija profili, DLNA medija profili, *DLNA Guidelines Media Format Profiles*