



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Никола Шпирин

**Једна реализација руководца
системских ресурса дигиталног ТВ
уређаја заснованом на Андроид
платформи**

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 2013



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:	Монографска документација		
Тип записа, ТЗ:	Текстуални штампани материјал		
Врста рада, ВР:	Завршни (Bachelor) рад		
Аутор, АУ:	Никола Шпиритић		
Ментор, МН:	др Јелена Ковачевић		
Наслов рада, НР:	Једна реализација руковаоца системских ресурса дигиталног ТВ уређаја заснованом на Андроид платформи		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2013		
Издавач, ИЗ:	Ауторски репринт		
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6		
Физички опис рада, ФО: (поглавља/страница/цитата/табела/слика/графика/прилога)	7/37/0/0/10/0/0		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника		
Предметна одредница/Кључне речи, ПО:	Android, Binder, middleware, DTV , IPC (engl. Inter-process Communication)		
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	Рад приказује једно решење реализације руковаоца системских ресурса ДТВ уређаја заснованом на Андроид оперативном систему. Циљ реализације је програмска подршка која омогућава конзистентно коришћење дељених хардверских и софтверских ресурса платформе у вишеклијентском окружењу. У раду је такође приказан опис програмске подршке задужене за међупроцесну комуникацију искоришћену за реализацију клијент-серверске архитектуре руковаоца.		
Датум прихватања теме, ДП:			
Датум одбране, ДО:			
Чланови комисије, КО:	Председник:	Проф. Др. Никола Теслић	
	Члан:	Мр. Милан Савић	Потпис ментора
	Члан, ментор:	Др. Јелена Ковачевић	



KEY WORDS DOCUMENTATION

Accession number, ANO:			
Identification number, INO:			
Document type, DT:	Monographic publication		
Type of record, TR:	Textual printed material		
Contents code, CC:	Bachelor Thesis		
Author, AU:	Nikola Špirić		
Mentor, MN:	Jelena Kovačević, PhD		
Title, TI:	One Implementation of Android Based DTV Device System Resource Manager		
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:	2013		
Publisher, PB:	Author's reprint		
Publication place, PP:	Novi Sad, Dositeja Obradovica sq. 6		
Physical description, PD: (chapters/pages/ref./tables/pictures/graphs/appendices)	7/37/0/0/10/0/0		
Scientific field, SF:	Electrical Engineering		
Scientific discipline, SD:	Computer Engineering, Engineering of Computer Based Systems		
Subject/Key words, S/KW:	Android, Binder, middleware, DTV , IPC		
UC			
Holding data, HD:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia		
Note, N:			
Abstract, AB:	This paper presents one implementation of Android based DTV device system resource manager. The goal is the realization of software that enables consistent usage of shared hardware and software resources in a multi-client environment. Description of inter-process communication mechanisms employed in the development of the client-server architecture is given.		
Accepted by the Scientific Board on, ASB:			
Defended on, DE:			
Defended Board, DB:	President:	Nikola Teslić, PhD	
	Member:	Milan Savić , MSc	Menthor's sign
	Member, Mentor:	Jelena Kovačević, PhD	

Zahvalnost

Zahvaljujem se mentoru dr Jeleni Kovačević i stručnim saradnicima Nikoli Kuzmanoviću i Milanu Saviću na savetima i pomoći tokom izrade završnog rada.

Posebno se zahvaljujem kolegama iz AMUSE tima na stručnoj pomoći prilikom izrade ovog rada.

Na kraju se zahvaljujem institutu RT-RK na pruženoj mogućnosti za realizaciju ovog rada.

SADRŽAJ

1.	Uvod	1
2.	Teorijske osnove.....	3
2.1	Android platforma.....	3
2.2	Digitalna televizija	4
2.3	Koncept DTV srednjeg sloja.....	5
2.4	Međuprocesna komunikacija	6
3.	Koncept rešenja	9
3.1	Comedia 3.0	9
3.2	Nexus	11
3.3	Realizacija.....	12
4.	Programsko rešenje.....	15
4.1	Realizacija međuprocesne komunikacije	15
4.1.1	Interfejs IResourceManager	16
4.1.2	Klasa BpResourceManager	18
4.1.3	Klasa BnResourceManager	20
4.2	Klasa ResourceManager	20
4.2.1	Klasa ResourceHolder	21
4.2.2	Apstraktna klasa AManagedResource.....	22
4.3	Asinhrona međuprocesna povratna funkcija.....	23
4.3.1	Klasa ResourceManagerListener	24
4.4	Klasa ResourceManagerClient.....	25
4.5	C omotač rukovaoca resursima	26
5.	Ispitivanje i verifikacija	28
5.1	Verifikacija u kontrolisanim uslovima.....	28
5.2	Verifikacija u realnim uslovima.....	29
6.	Zaključak	30
7.	Literatura	31

SPISAK SLIKA

Slika 2.1 Struktura Android softverskog steka	4
Slika 2.2 Hierarhijska organizacija slojeva.....	6
Slika 2.3 Životni vek jedne transakcije između dva procesa	8
Slika 3.1 Arhitektura Comedia DTV srednjeg sloja	10
Slika 3.2 Arhitektura Nexus softverskog steka.....	11
Slika 4.1 Hierarhijska organizacija klasa rukovaoca resursima	16
Slika 4.2 Proces dobavljanja resursa.....	19
Slika 4.3 Odnos osnovnih komponenti sistema rukovalca resursima.....	23
Slika 4.4 Hierarhijska organizacija klasa međuprocesne povratne funkcije	24
Slika 4.5 Scenario oduzimanja resursa od korisnika sa nižim prioritetom.....	25

SKRACENICE

- IPC** - *Interprocess Communication*, Međuprocesna komunikacija
- RPC** - *Remote Procedure Call*, Poziv udaljene procedure
- RTTI** - *Run-Time Type Information*, Tipska informacija u vremenu rada
- CHAL** - *Comedia Hardware Abstraction Layer*, Sloj apstrakcije fizičke platforme
- DTV** - *Digital Television*, Digitalna televizija
- OS** - *Operating System*, Operativni sistem
- API** - Application Programming Interface, Programska sprega
- PES** - Program Elementary Stream, Elementarni programski tok

1. Uvod

U ovom radu opisano je jedno rešenje realizacije rukovaoca sistemskih resursa DTV (engl. Digital Television) uređaja zasnovanom na Android platformi. Cilj realizacije je programska podrška koja omogućava konzistentno korišćenje deljenih hardverskih i softverskih resursa platforme u višekorisničkom okruženju. U radu je takođe prikazan opis programske podrške zadužene za međuprocesnu komunikaciju iskorišćenu za realizaciju klijent-serverske arhitekture rukovaoca.

Jedan od problema koji je uočen prilikom razvijanja programske podrške na Android [1] platformi je nepostojanje mehanizma za rukovanje sistemskim resursima hardverske platforme između više prisutnih korisnika istih. Realizacija mehanizma rukovanja resursa je neophodna ukoliko pojedinačni moduli platforme (e.g. Andorid multimedijalni sloj) nisu jedini direktni korisnici hardverskih resursa platforme. Potrebu za korišćenjem pomenutih resursa, u razmatranom slučaju, ima i DTV srednji sloj koji je prvobitno razvijen kao programska komponenta nezavisna od samog operativnog sistema. Radi realizacije svojih osnovnih funkcionalnosti, među koje spada reprodukcija audio i video sadržaja, srednji sloj mora imati direktni pristup sistemskim resursima platforme. Android platforma ne poseduje mehanizam kontrole i evidencije iskorišćenih resursa, a samim tim i indirektne kontrole srednjeg sloja, usled čega postoji mogućnost dolaska do nepredvidivosti i nestabilnost rada celokupnog sistema. Da bi se izbegla izmena izvornog koda rukovaoca (engl. driver) u cilju dodavanja podrške višekorisničkog režima rada, uveden je novi sloj programske podrške između korisničkih aplikacija i hardverke programske sprege koji pruža bezbedan i konzistentan simultan rad više korisnika. Rešenje je implementirano na digitalnom TV prijemniku BCM97435VMS kompanije Broadcom. Kao DTV srednji sloj korišćena je Comedia 3.0.

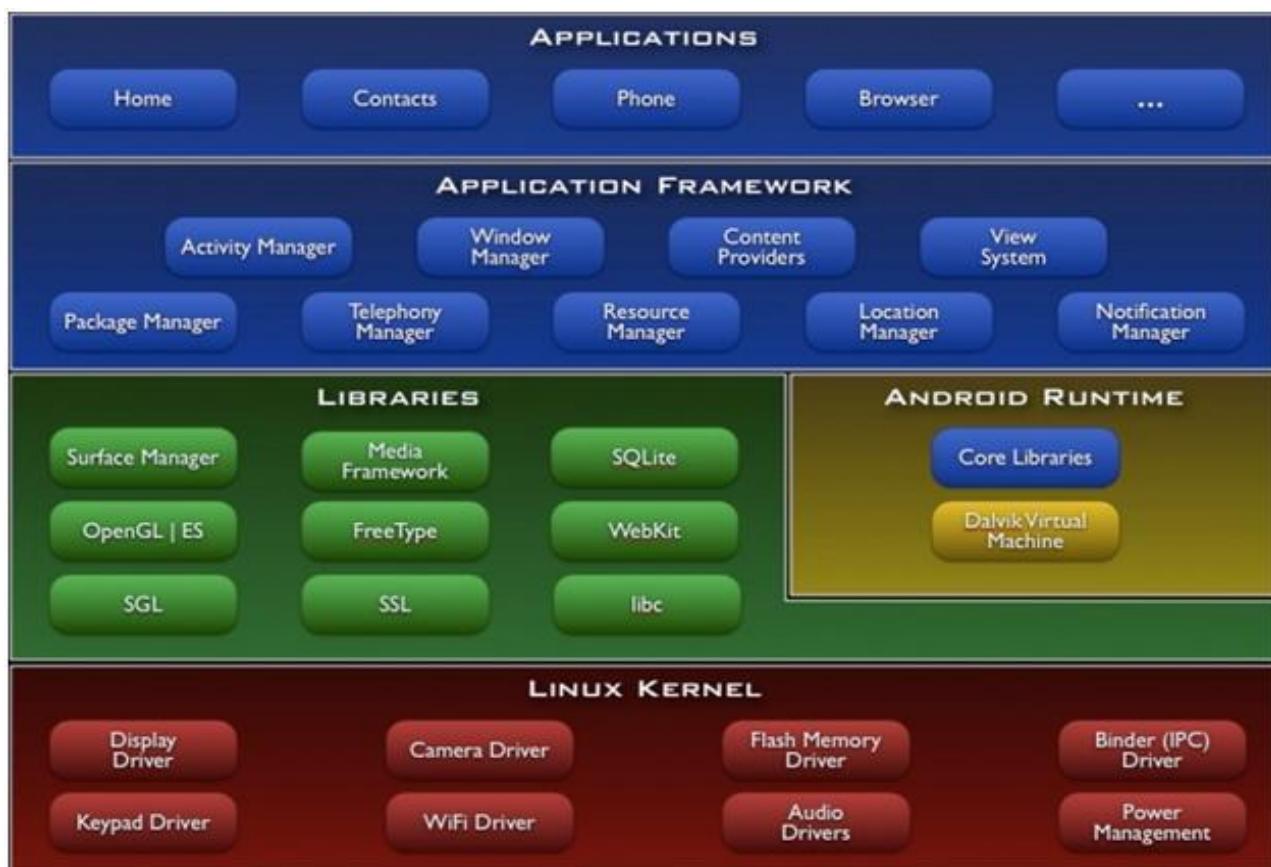
Rad se sastoji od sedam poglavlja. U drugom poglavlju date su teorijske osnove međuprocsne komunikacije neophodne za realizaciju sistema, kao i opis koncepta DTV srednjeg sloja. Treće poglavlje opisuje koncept rešenja gde je počevši od opšte slike rešenja, razrađeno do nivoa detalja kako bi rešenje trebalo da izgleda. U četvrtom poglavlju data je konkretizacija rešenja koja podrazumeva detalje realizacije modula sa opisima sprege između njih. U petom i šestom poglavlju dat je kratak osvrt na ono što je urađen zajedno sa evaluacijom rada rešenja.

2. Teorijske osnove

2.1 Android platforma

Android je Linux zasnovana platforma, prvobitno namenjena mobilnim uređajima poput pametnih telefona, tablet računara itd. U današnje vreme mnogi proizvođači mobilnih telefona koriste Android kao platformu. Razvijen od strane kompanije Google, Android je realizovan u obliku softverskog steka koji uključuje jezgro operativnog sistema, srednji (engl. middleware) i aplikativni sloj programa. Jedan od razloga velikog interesovanja za Android je dostupnost u izvornom kodu. Samim tim Android je pogodan za dalja poboljšanja i modifikacije u skladu sa potrebama pojedinaca širom sveta. Ceo stek, koji je prvobitno namenjem uređajima zasnovanim na ARM (engl. Advanced RISC machine) arhitekturi, 2009. godine prilagođen [2] i MIPS (Microprocessor without Interlocked Pipeline Stages) arhitekturi procesora što ga je učinilo pogodnijim za korišćenje u čitavom novom spektru uređaja poput televizora i VoIP telefona.

Android se sastoji iz jezgra operativnog sistema zaduženog za rukovanje fizičkom arhitekturom i funkcionalnostima niskog nivoa konkretne platforme i skupa dodatnih biblioteka zaduženih za dodatnu programsku podršku kao što je iscrtavanje grafike, reprodukcija audio i video sadržaja, iscrtavanje WEB sadržaja, rukovanje bazama podataka itd. U sklopu biblioteka se nalazi i odvojeni "Android Runtime" sloj, koji sadrži bazne biblioteke neophodne za realizaciju samog operativnog sistema kao i Dalvik virtuelnu mašinu zaduženu za pokretanje aplikacija višeg nivoa napisanih u Java programskom jeziku. Na višim slojevima Android steka nalaze se biblioteke neophodne za pristup osnovnim funkcionalnostima sistema od strane korisnika, dok se na kraјnjem sloju nalaze same aplikacije kojima korisnik ima pristup. Slika 2.1 detaljnije ilustruje pomenute slojeve.



Slika 2.1 Struktura Android softverskog steka

2.2 Digitalna televizija

Pod pojmom digitalne televizije podrazumeva se transmisija audio i video zapisa zajedno sa dodatnim informacijama u putem digitalno procesiranog i multipleksiranog signala. Pojava i uspostavljanje standarda koji se koriste u digitalnoj televiziji vezuje se za poslednju dekadu prošlog veka. Prva demonstracija digitalnog TV prenosa održana je 1995. godine. U narednim godinama sledi prestanak emitovanja analognog signala i zamena istog sa digitalnim.

Slika i zvuk se prilikom snimanja pretvaraju u digitalnu formu postupcima odmeravanja, kvantovanja i kodovanja i u takvom obliku se prenose kroz mediju prenosna, odnosno kanal. Kako je DTV sadržaj digitalno kodiran, pogodan je za komprimovanje korišćenjem postojećih metoda. Kompresija omogućava emitovanje približno deset puta više kanala od analogne televizije.

Pored pomenutih prednosti DTV uvodi i širok spektar drugih pogodnosti, kao što su:

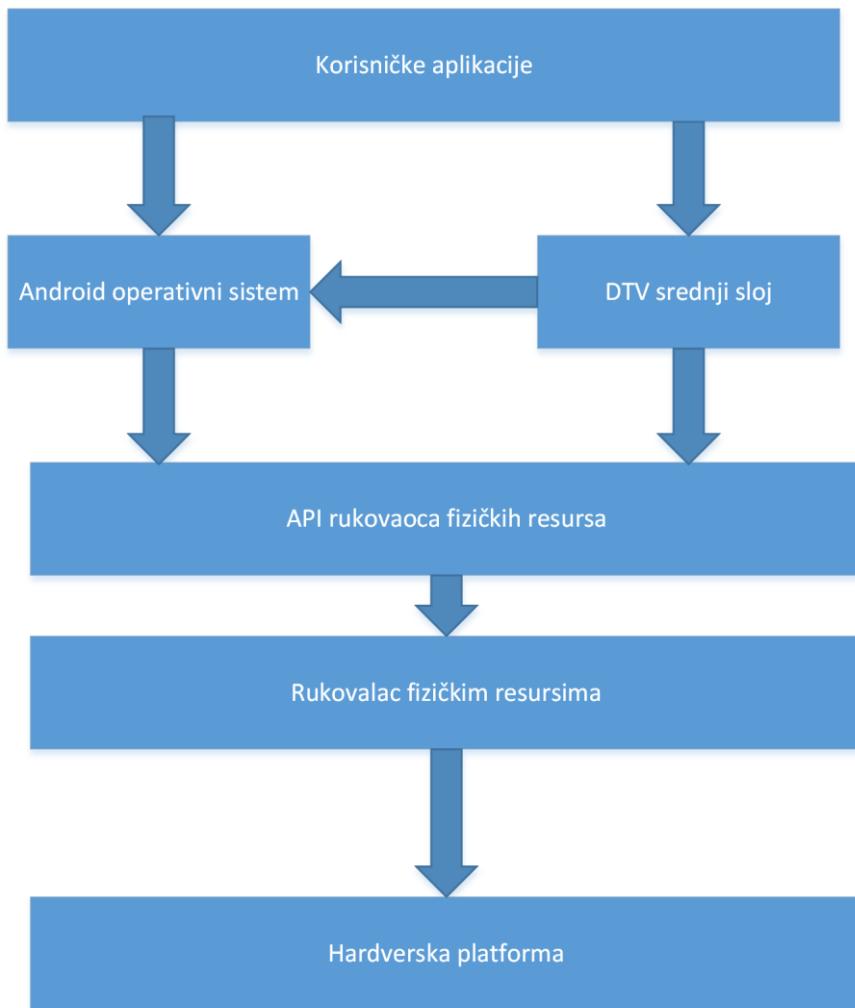
- Bolji kvalitet slike i zvuka koji više ne mogu biti u prenosu ometani interferencijom sa drugim signalima, bez obzira na rastojanje na koje se prenose.

- Digitalna televizija omogućava nove aplikacije kao što je interaktivna televizija koja omogućava izvršavanje računarskih aplikacija pisanih u Java, HTML ili ActionScript jeziku.
- Pored boljeg kvaliteta slike i zvuka, obezbeđuje i mogućnost njihove različite prezentacije (promena formata slike, broja kanala zvuka itd.)
- Omogućava uvođenje dodatnih usluga poput izvora jezika za titlovanje i audio kanala, interaktivni i multimedijalni sadržaji, pristup internetu itd.

2.3 Koncept DTV srednjeg sloja

Pod konceptom srednjeg sloja (engl. middleware), podrazumeva se kompleksna programska podrška zadužena apstrakciju fizičkih okruženja (hardverskih platforma) korisničkih aplikacija. Iako postoje razne definicije ovog sloja, kao i mnoštvo realizacija, u ovom radu biće razmatran sloj koji apstrahuje funkcionalnosti hardverske platforme relevantne za razvoj aplikacija zasnovanih na tehnologijama digitalne televizije, među koje spada reprodukcija audio i video sadržaja, EPG (engl. Electronic Program Guide), teletekst, PVR (engl. Personal Video Recorder), itd. Većina proizvođača DTV programske podrške koja se koristi u DTV prijemnicima ne ograničava svoje usluge samo na osnovne funkcije operativnog sistema, već se operativni sistem prodaje kao deo kompletног paketa koji uključuje i srednji sloj. Srednji sloj obezbeđuje odgovarajuću programsku spregu koja pruža apstrakciju funkcionalnosti TV uređaja, hardverske platforme kao i funkcija operativnog sistema čime je omogućeno da proizvođači aplikativne DTV programske podrške ne moraju da poznaju hardverske specifičnosti DTV prijemnika. Ovim je omogućeno da se aplikacije koje koriste isti srednji sloj mogu izvršavati na različitim hardverskim platformama.

Slika 2.2. ilustruje hierarhijsku organizaciju slojeva prisutnih na Android zasnovanoj DTV platformi, gde se lakše može uočiti odnos srednjeg sloja i operativnog sistema u zavisnosti od iskorišćenja hardverskih resursa. Savremeni DTV srednji slojevi obično podržavaju apstrakciju rukovanja grafičkom korisničkom spregom, međutim u posmatranom slučaju za tu funkcionalnost aplikacija se oslanja na Android platformu. Iz ovog razloga, sa tačke gledišta aplikacije, Android i srednji sloj se nalaze na istom hierarhijskom nivou.



Slika 2.2 Hierarhijska organizacija slojeva

2.4 Međuprocesna komunikacija

Radi uspešne realizacija rukovaoca sistemskih resursa neophodno je postojanje mehanizma komunikacije dva procesa operativnog sistema.

Kao što je to slučaj sa svim savremenim operativnim sistemima i Android platforma je zasnovana na multiprocesnom jezgru operativnog sistema [3]. Ovo znači da svaka izvršiva datoteka dobija određeni memorijski okvir i poseduje sopstveni stek, rezervisanu procesnu memoriju, itd. Iz bezbednosnih razloga, jedan proces ne sme biti u mogućnosti da vrši manipulaciju podataka drugog procesa. U cilju realizacije ovog zahteva, operativni sistem mora integrisati neku vrstu procesne izolacije. U slučaju Linux operativnog sistema, koncept virtuelne memorije je uveden [4] gde se svakom procesu dodeljuje sopstveni adresni prostor sa virtuelnim adresama koje se dalje mapiraju na fizičke adrese od strane operativnog sistema. Ovim pristupom jednom procesu je uspešno onemogućen pristup memorijskom prostoru

drugog procesa. Procesna izolacija daje svakom procesu bezbednost memorije, ali u većini slučajeva postojanje mehanizma komunikacije dva procesa je neophodno. Međuprocesna komunikacija (u daljem tekstu IPC engl. Interprocess Communication) podrazumeva razmenu podataka između dva procesa. U cilju odabira adekvatnog IPC mehanizma za konkretan problem, razmotreni su neki od postojećih metoda. Neki od postojećih IPC metoda realizovanih u Linux operativnom sistemu [4] su:

- Signal – Jedna od najstarijih IPC metoda. Jedan proces može poslati signal drugom procesu iste grupe.
- Pipe – Jednosmerni tok podataka koji povezuje standardni izlaz jednog procesa sa standardnim ulazom drugog.
- Socket – Omogućuje dvosmernu komunikaciju dva procesa putem slanja toka podataka na istom socket-u.
- Deljena memorija – Lokacija u sistemskoj memoriji koja se mapira u virtualni adresni prostor dva procesa tako da oba procesa imaju potpun pristup istom segmentu fizičke memorije.

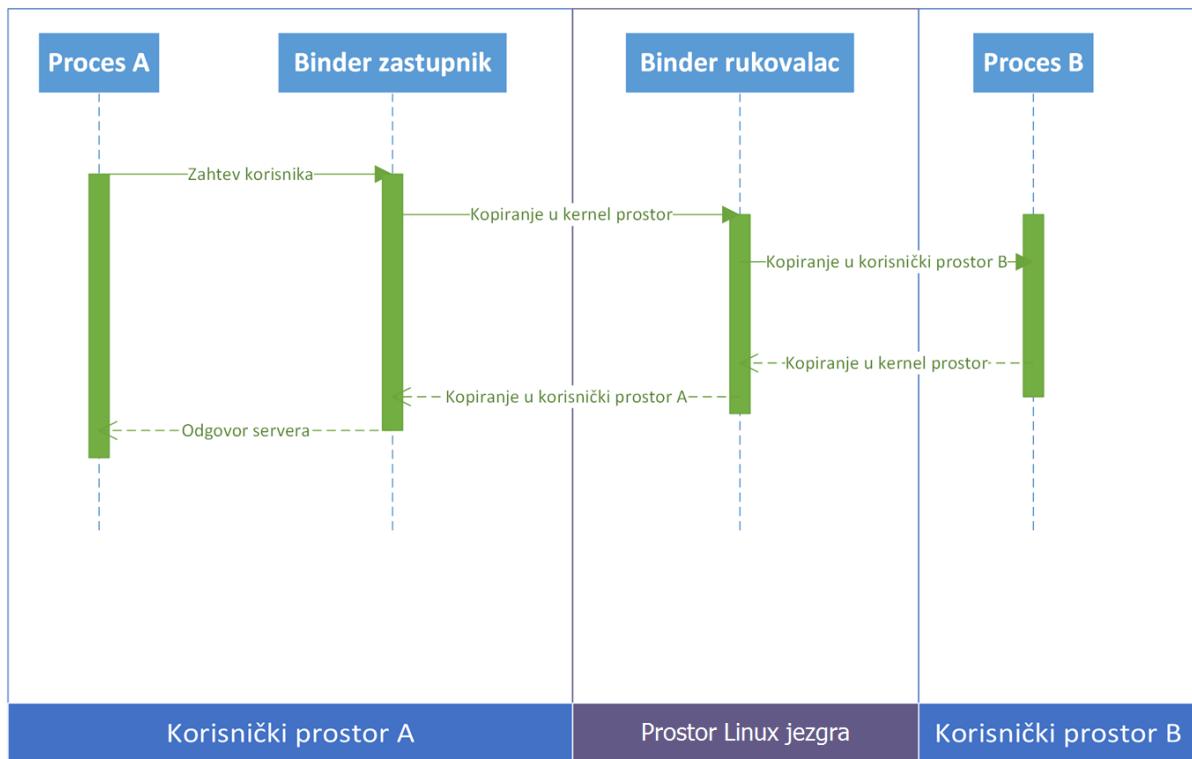
Iako su svi od navedenih metoda u potpunosti validni načini realizacije međuprocesne komunikacije, u Android platformi iskorišćen je metod koji znatno olakšava razmenu apstraktnih tipova podataka između dva procesa. Ovaj metod je realizovan u vidu Binder radnog okvira.

Binder radni okvir je prvobitno razvijen pod nazivom OpenBinder [5] od strane Be Inc kompanije. Iz originalne dokumentacije OpenBinder je definisan kao "... komponenta arhitekture sistemskog nivoa, dizajnirana da pruži uslugu bogatije apstrakcije visokog nivoa kao nadogradnja tradicionalnih servisa operativnih sistema.". Konkretno Binder nudi mehanizam pristupa podacima i funkcijama iz jednog izvršnog okruženja ka drugom.

Komunikacija Binder radnog okvira se zasniva na klijent-server arhitekturi. Klijent započinje komunikaciju slanjem zahteva serveru, nakon čega čeka na odgovor. Ovakva komunikacija je iz perspektive korisnika Binder-a u potpunosti sinhrona, što znatno olakšava realizaciju komunikacije ali zahteva dodatnu intervenciju programera ukoliko je asinhrona komunikacija neophodna.

Slikom 2. ilustrovan je primer jedne transakcije i njenog životnog veka. Pod pojmom transakcije podrazumeva se razmena podataka između dva procesa u vidu jednog klijentskog zahteva i jednog serverskog odgovora na zahtev [6]. Sa klijentske strane komunikacija sa Android Binder rukovaocem vrši se putem klijentskog Binder zastupnika, koji indirektno komunicira sa jednom od više serverskih niti koje obrađuju klijentske zahteve. Međuprocesna

transakcija uvodi prekoračenje (engl. overhead) u vidu 4 operacija kopiranja, između dva korisnička memoriska prostora i Android prostora jezgra, kao što je prikazano na slici.



Slika 2.3 Životni vek jedne transakcije između dva procesa

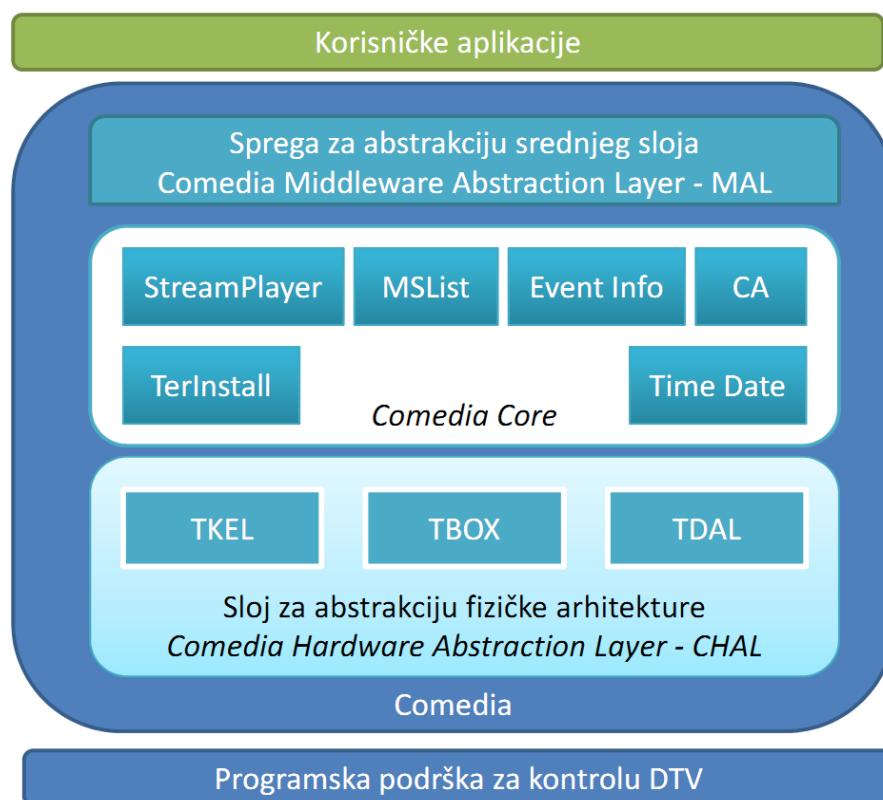
Prenos podataka u transakciji vrši se putem Parcel objekta. Parcel objekti omogućavaju dekompoziciju apstraktnih tipova podataka na elementarne delove i njihovo skladištenje u kontinualni blok memorije koji je dalje moguće kopirati iz jednog memoriskog prostora u drugi.

3. Koncept rešenja

Programska podrška za rukovanje sistemskim resursima DTV uređaja na Android platformi realizovana je kao rukovalac-poslužilac arhitektura. Rukovalac resursima je programska komponenta realizovana kao zastupnik između svih korisnika hardverskih resursa i rukovaoca fizičke platforme (engl. driver). Glavna DTV aplikacija sistema, koja se oslanja na pomenuti srednji sloj je Android4TV [8].

3.1 Comedia 3.0

Srednji sloj Comedia realizovan je iz tri osnovna dela [7]: Comedia Core, CHAL i MAL kao što je ilustrovano na slici 3.1. Comedia Core, tj. jezgro srednjeg sloja, zadužen je za realizaciju osnovnih DTV funkcionalnosti i nezavisan je od fizičke platforme.



Slika 3.1 Arhitektura Comedia DTV srednjeg sloja

MAL je akronim od “Middleware Abstraction Layer”, odnosno sloj za apstrakciju srednjeg sloja koji vrši enkapsulaciju svih funkcionalnosti srednjeg sloja kroz programsku spregu koja omogućava laku integraciju sa slojevima višeg nivoa, realizovanim kao Java ili Javascript/HTML aplikacije.

CHAL je akronim od “Comedia Hardware Abstraction Layer”, odnosno sloj za apstrakciju fizičke arhitekture, na koji se jezgro komedije oslanja za svoju osnovnu funkcionalnost. CHAL sloj je zavisan od fizičke arhitekture prijemnika i da bi se omogućila funkcionalnost Comedia s rednjeg sloja, CHAL je sloj koji treba preneti na ciljnu platformu. Sastoji se od tri nezavisne celine:

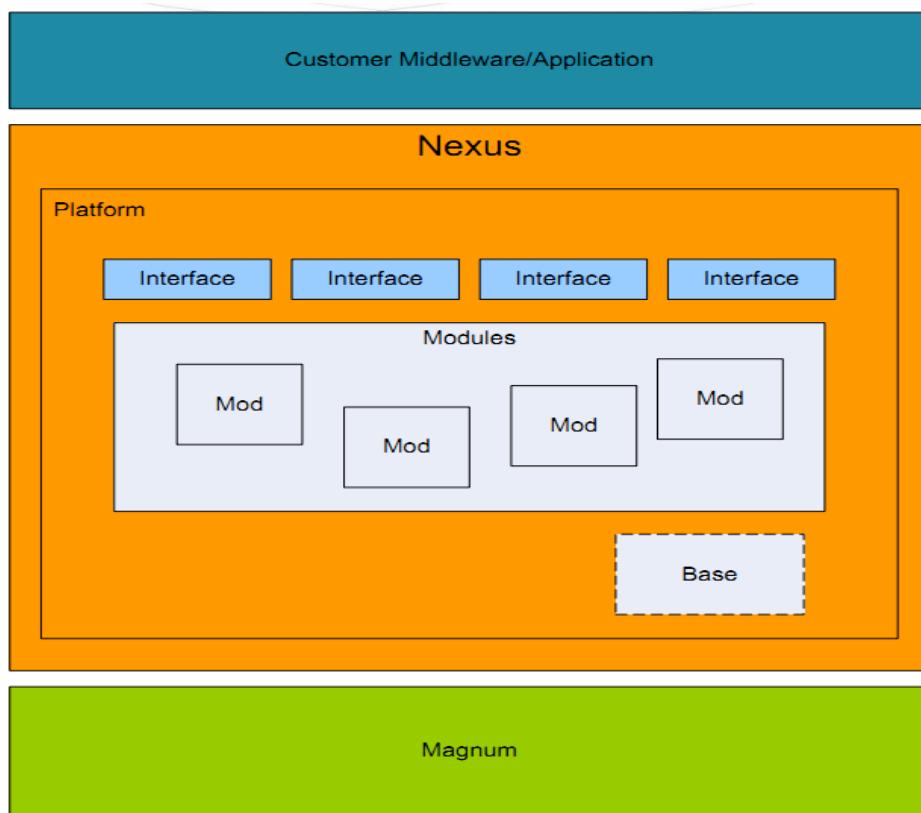
- TKEL – Sloj za apstrakciju funkcionalnosti operativnog sistema.
- TBOX – Modul koji deklariše pomoćne funkcije i makroe i služi u svrhu debagovanja programske podrške.
- TDAL – Sloj za apstrakciju rukovalaca koji se sastoji od mnoštva modula. Svaki blok fizičke arhitekture DTV prijemnika poseduje odgovarajući TDAL rukovalac.

Od intresa za programsko rešenje opisano u ovom radu je TDAL modul AV. Uloga ovog modula je da u potpunosti kontroliše postupak dekodovanja video i audio PES u okviru DTV prijemnika. Kako su audio i video dekoderi fizički resursi koji se dele sa

multimedijalnim slojem Android platforme, dobavljanje i korišćenje istih se vrši putem realizovanog rukovaoca resursa.

3.2 Nexus

Nexus je modularan API visokog nivoa dizajniran za Broadcom DTV uređaje. Cilj Nexus-a je olakšavanje razvoja aplikacija, pružanjem jednostavnog interfejsa koji apstrahuje funkcionalnosti nižih slojeva programske sprege rukovaoca fizičkim resursima. Osnovne funkcionalnosti fizičke platforme izložene su Nexus programskoj podršci putem Magnum modula.



Slika 3.2 Arhitektura Nexus softverskog steka

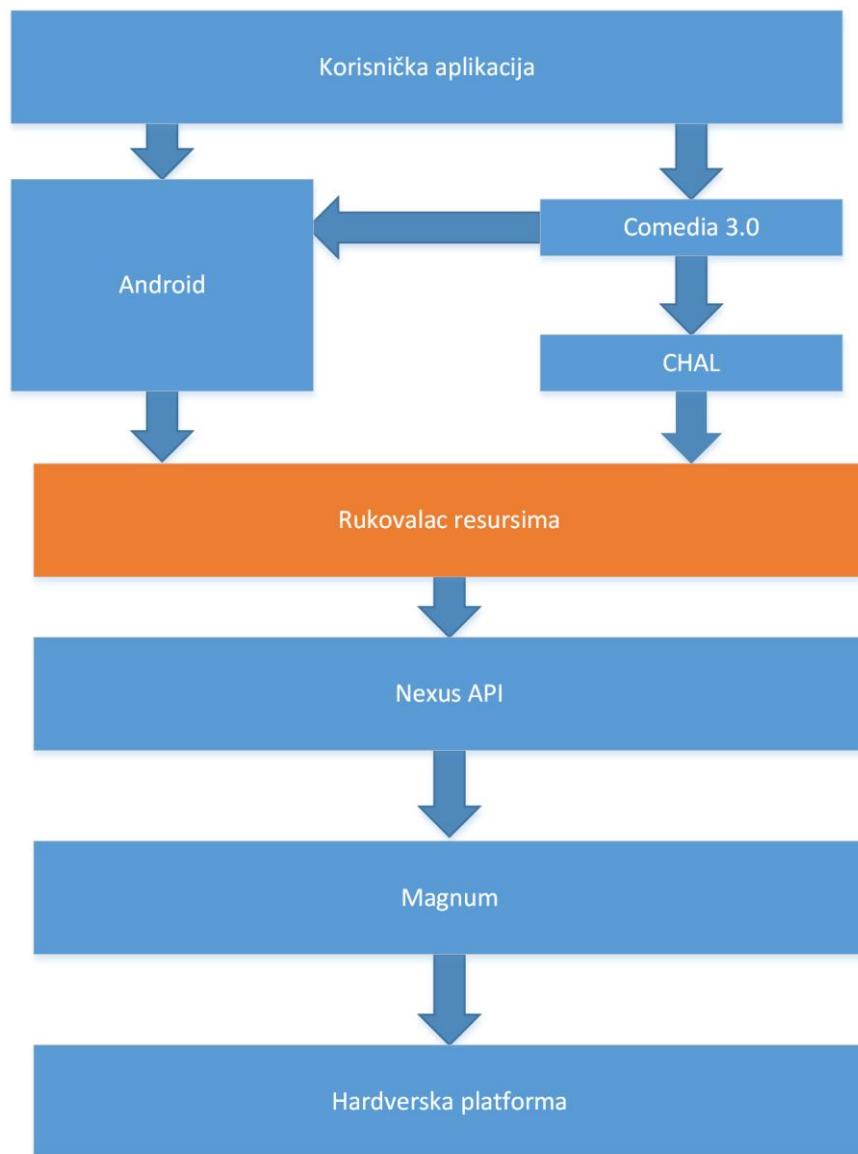
Realizovan rukovaoc resursa se je uveden kao posrednički sloj između Nexus API-ja i korisnika fizičkih resursa. Resursi od interesa čije je rukovanje realizovano opisanom programskom podrškom, enkapsulirani su putem sledeći Nexus interfejsa:

- NEXUS_VideoDecoder – Apstrakcija jedinice dekodovanja video sadržaja.
- NEXUS_AudioDecoder – Apstrakcija jedinice dekodovanja audio sadržaja.
- NEXUS_Playpump – Mehanizam dostavljanja kodovanog audio/video sadržaja do respektivnog dekodera.

- NEXUS_Recpump –Mehanizam čitanja dostavljenog audio/video sadržaja. Koristi se za realizaciju PVR funkcionalnosti.

3.3 Realizacija

Rukovalac resursima je realizovan kao Android servis čime je omogućena dostupnost funkcionalnosti istog širom sistema. Pored realizacije samog servisa, modifikacija multimedijalnog sloja Android platforme kao i sloja apstrakcije fizičke arhitekture (u daljem tekstu CHAL) modula Comedia srednjeg sloja je bila neophodna.



Slika 3.1 Hierarhijska organizacija slojeva nakon realizacije rukovaoca resursima

Slika 3.1 ilustruje novu hierarhijsku raspodelu slojeva posle dodavanja rukovaoca hardverskim resursima. Jedan od glavnih resursa u posmatranom sistemu čije je rukovođenje

neophodno su audio i video dekoderi. Potrebu za korišćenjem oba ima i Androidov multimedijalni sloj kao i CHAL čija je jedna od osnovnih funkcionalnosti podrška audio i video reprodukcije. Uzimajući u obzir konkretne zahteve korisnika na dатој platformi, bilo je potrebno realizovati sledeće funkcionalnosti rukovaoca:

- Registracija korisnika resursa.
- Evidencija privilegija pojedinih korisnika.
- Mehanizam dobavljanja traženih resursa
- Evidencija dostupnosti i sposobnosti hardverskih resursa.
- Mehanizam asinhronne, dvosmerne komunikacije između korisnika i rukovaoca.
- Realizacija međuprocesne povratne funkcije.

Rukovalac resursima realizovan je kao Android servis za čije pokretanje je zadužena sama platforma, što je realizovano izmenom “init.rc” datoteke. Pri inicijalizaciji server vrši registraciju svoje jedinstvene instance kod Androidovog rukovaoca servisima (*ServiceManager*). Posle registracije, svaka klijentska aplikacija može da dobavi instancu servera, ukoliko joj je poznat njegov identifikator. Identifikatori servisa u Android platformi su njihova jedinstvena imena u vidu niza karaktera, i kao takva moraju biti unapred poznata klijentskoj aplikaciji koja zahteva njene usluge.

Pri inicijalizaciji sistema server vrši evidenciju raspoloživih resursa (e.g. ukupan broj dostupnih audio/video dekodera) u zavisnosti od mogućnosti konkretnе platforme i otpočinje čekanje i prihvata korisničkih veza. Da bi pojedinačni korisnici bili u mogućnosti dobavljanja i upotrebe resursa, njihova registracija kod servera je neophodna. Putem registracije korisnika, server ima mogućnost vođenja evidencije vlasništva pojedinačnih resursa kao i prioriteta pojedinačnih korisnika.

Sve funkcionalnosti servera sa korisničke strane su enkapsulirane C++ klasom putem koje korisnici dobijaju pristup javnom programskoj sprezi servera. Registracija korisnika se na ovaj način svodi na poziv jedne funkcije pomenutog objekta. Komunikacija sa serverskim servisom je ostvarena korišćenjem Android Binder programske podrške.

Prilikom poziva funkcije registracije korisnika, korisnik dobavlja instancu serverskog servisa znajući predefinisano ime istog. Posle sticanja zahteva za registraciju, server dodeljuje jedinstveni identifikator (u daljem tekstu žeton) u vidu 32-bitnog celog broja generisanog na osnovu jedinstvenog imena klijenta, svakom korisniku. Dobavljanje i korišćenje hardverskih resursa (kao i ostalih usluga rukovaoca) nije moguće bez posedovanja ovog žetona. Prilikom završetka rada klijenta, vrši se deinicijalizacija njegove instance, a samim tim i oslobođanje svih resursa asociranih sa pomenutim klijentom.

Mehanizam asinhronog obaveštavanja korisnika o promenama vlasnika resursa neophodan je kod sistema sa više od jednog korisnika. Mogućnosti hardverske platforme su u praksi ograničene, pa je situacija u kojoj dva klijenta moraju da dele isti resurs očekivana. Kako je prvobitna namena opisanog sistema DTV funkcionalnost, veći prioritet od pomenuta dva klijenta ima DTV srednji sloj(u daljem tekstu Comedia) na koji se vodeća Android aplikacija oslanja. U tu svrhu uveden je sistem prioriteta korisnika. Prilikom registracije, svaki korisnik prosleđuje željeni prioritet sopstvene aplikacije serveru. Realizovana su tri različita nivoa prioriteta:

1. Najniži - Prioritet namenjen aplikacijama čiji rad može biti prekinut u svakom trenutku (e.g. ispitne aplikacije koje koriste server).
2. Srednji - Prioritet namenjen za aplikacije čija funkcionalnost nije kritična za sistem, ali čije je neometano funkcionisanje poželjno. U posmatranom sistemu ovaj prioritet uzima Android multimedijalni sloj.
3. Najviši – Prioritet namenjen za aplikacije čiji rad ne sme biti prekinut. Kako je osnovna namena posmatranog sistema DTV funkcionalnost, ovaj prioritet uzima Comedia srednji sloj.

Ukoliko korisnik sa višim prioritetom pošalje zahtev za resursom koji je trenutno zauzet od strane korisnika sa nižim, od servera se očekuje da obavesti trenutnog korisnika o gubitku pomenutog resursa. Korisnik koji je izgubio resurs u tom trenutku ima priliku da odreaguje na način koji neće ugroziti stabilnost sistema i zaustavi rad aplikacije koja je zavisila od konkretnog resursa (e.g. Android VideoView [9] grafička komponenta koja je vršila audio/video reprodukciju u datom trenutku).

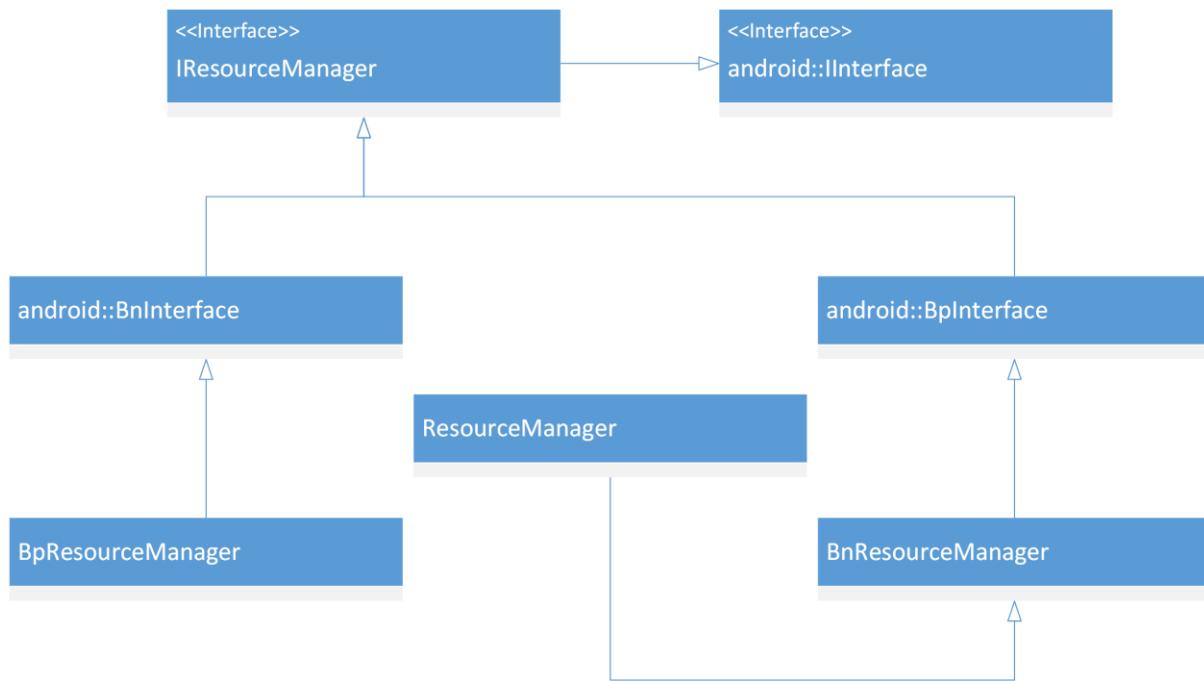
Ovakva dvosmerna asinhrona komunikacija između korisnika i servera realizovana je daljim korišćenjem Android Binder radnog okvira. Pri registraciji, svaki korisnik prosleđuje povratnu funkciju serveru putem koje dobija obaveštenja o bitnim događajima. Prenos ove funkcije ostvaren je putem posebne instance Android Binder objekta po svakom korisniku koja se predaje serveru prilikom registracije. Serijalizacija Binder objekata omogućena je direktno pisanjem pokazivača istih u Parcel objekte [6]. Prilikom primanja transakcije koja obaveštava klijenta o gubitku resursa, vrši se pozivanje registrovane povratne funkcije čime se korisnik obaveštava o događaju.

4. Programsko rešenje

Programsko rešenje se sastoji od 3 glavna modula, realizovana u programskom jeziku C++ uz korišćenje pomoćnih biblioteka Android programskog steka. Dodatan modul realizovan u programskom jeziku C omogućava dostupnost uslugama rukovaoca resursima i srendjem sloju (CHAL) koji je pisan u istom.

4.1 Realizacija međuprocesne komunikacije

Međuprocesna komunikacija između klijenata i servera realizovana je korišćenjem Androidovog Binder randog okvira. Klase neophodne za realizaciju iste su ilustrovane UML dijagramom klase na slici 4.1.



Slika 4.1 Hierarhijska organizacija klasa rukovaoca resursima

4.1.1 Interfejs IResourceManager

Interfejs IResourceManager predstavlja vezu između klijentskog koda i servera. Izvedena iz Androidove klase IIInterface, u sebi sadrži sve javne funkcije kojima se korisnički moduli mogu služiti, odsnosno sve funkcije koje je rukovalac realizuje. Ovu klasu nasleđuje kako korisnička tako i serverska strana sistema. Klasa IResourceManager sadrži sledeće funkcije:

- `virtual ClientToken createClient(const ResourceManagerClientConfig* config)` – Vrši kreaciju i registraciju klijeta, na osnovu pružene konfiguracije u strukturi `ResourceManagerClientConfig` kreira jedinstveni identifikacioni žeton i dostavlja ga klijentu kao povratnu vrednost funkcije.
- `virtual void destroyClient(ClientToken client)` – Vrši deinicijalizaciju klijenta. Svi resursi asocirani sa dotičnim klijentom se oslobođaju i evidentiraju kao dostupni za druge klijente.
- `virtual ResourceManagerError getVideoWindowSettings(ClientToken client, int windowId, VideoWindowSettings* settings)` – Dobavlja trenutna podešavanja video prozora sa datim indeksom.

- `virtual ResourceManagerError setVideoWindowSettings(ClientToken client, int window_id, VideoWindowSettings *settings)` - Postavlja nova podešavanja za dati video prozor.

- `virtual NEXUS_AudioDecoderHandle acquireAudioDecoder(ClientToken client)`
 - Dobavlja instancu video dekodera. Ukoliko su svi dekoderi zauzeti funkcija vraća `NULL` kao rezultat. Ukoliko postoji slobodan dekoder, korisnik ga dobija kao povratnu vrednost, a isti se evidentira kao zauzet onemogućujući ostalim korisnicima da ga dobave.

- `virtual ResourceManagerError releaseAudioDecoder(ClientToken client, NEXUS_AudioDecoderHandle handle)` - Oslobađa vlasništvo korisnika nad datoj instanci dekodera. Dekoder se evidentira kao dostupan omogućujući ostalim korisnicima da ga dobave.

- `virtual NEXUS_VideoDecoderHandle acquireVideoDecoder(ClientToken client)`
 - Analogno funkciji dobavljanja audio dekodera.

- `virtual ResourceManagerError releaseVideoDecoder(ClientToken client, NEXUS_VideoDecoderHandle handle)` - Analogno funkciji oslobađanja audio dekodera.

- `virtual NEXUS_PlaypumpHandle acquirePlaypump(ClientToken client)` - Analogno funkciji dobavljanja audio dekodera.

- `virtual ResourceManagerError releasePlaypump(ClientToken client, NEXUS_PlaypumpHandle handle)` - Analogno funkciji oslobađanja audio dekodera.

- `virtual NEXUS_RecpumpHandle acquireRecpump(ClientToken client)` - Analogno funkciji dobavljanja audio dekodera.

- `virtual ResourceManagerError releaseRecpump(ClientToken client, NEXUS_RecpumpHandle handle)` - Analogno funkciji oslobađanja audio dekodera.

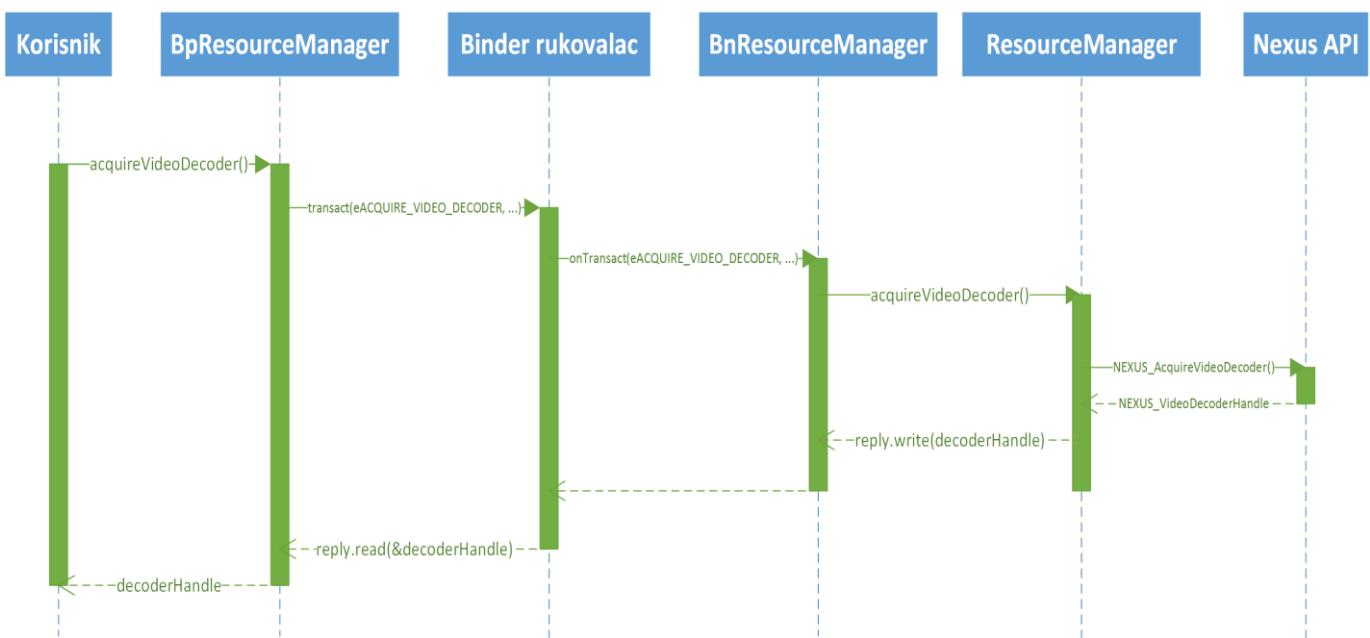
- `virtual ResourceManagerError acquireMosaicCell(ClientToken client, int index, NEXUS_VideoWindowHandle* window, NEXUS_VideoDecoderHandle* decoder)` – Dobavlja par video prozor-dekoder što predstavlja mozaik čeliju, sposobnu za video reprodukciju mozaik modu.
- `virtual ResourceManagerError releaseMosaicCell(ClientToken client, int index)` – Oslobađa prethodno zazutetu mozaik čeliju.

4.1.2 Klasa `BpResourceManager`

Klasa `BpResourceManager` predstavlja posrednički sloj između korisnika i Binder rukovaoca, a samim tim i serverskog dela rukovaoca resurisma. Kako ova klasa nasleđuje zajednički interfejs `IResourceManager`, u njoj se nalaze realizacije svih predefinisanih virtuelnih funkcija. Putem ove klase vrši se apstrakcija komunikacije putem Binder radnog okvira. Korisnik rukovaoca resursima ne brine o načinu prenosa podataka i dekompozicije podataka koji se šalju.

Dobavljanje `IBinder` instance, putem koje ova klasa može da komunicira sa `BnResourceManager` klasom (što je serverski pandan `BpResourceManager` klasi) vrši se putem funkcije `remote()` koja se nalazi u Android `BpInterface` klasi čija je povratna vrednost pokazivač na `IBinder` instancu.

Osnovna jedinica komunikacije između klijenta i servera je transakcija. Transakcija se sastoji od prenosa dva `Parcel` objekta, jednog od klijenta ka serveru (što predstavlja zahtev poziva funkcionalnosti) i jednog od servera ka klijentu (što predstavlja odgovor na prethodno pomenut zahtev). Životni vek transakcije je detaljnije ilustrovan dijagramom sekvene na slici 4.2 gde je prikazan proces dobavljanja video dekodera.



Slika 4.2 Proces dobavljanja resursa

Realizacija funkcija definisanih `IResourceManager` interfejsom se svodi na dekompoziciju svih apstraktnih tipova podataka koji su argumenti funkcije, i njihov upis u `Parcel` objekat koji je dalje pogodan za slanje serverskom procesu. Posle kreacije `Parcel` objekta, u kom se sada nalaze svi parametri funkcije, podaci se dalje šalju serveru pozivom `IBinder` funkcije:

```
IBinder::transact(uint32_t code, const Parcel& data, Parcel* reply)
```

Čiji su parametri:

- `uint32_t code` – Jedinstveni identifikator poruke koja se šalje. Na osnovu njega serverska strana određuje na koji način će interpretirati sadržaj poruke. Kodovi relevantni za realizovan rukovalac resursima nalaze se u `IResourceManager::ApiName` enumeraciji. U konkretnom slučaju svaki član enumeracija odgovara jednoj funkciji interfejsa.
- `const Parcel& data` – `Parcel` objekat koji se šalje serveru i koji sadrži prethodno upisane parametre funkcije.
- `Parcel* reply` – `Parcel` objekat u koji će biti upisan odgovor serverske strane. Putem ovog parametra ostvarena je dvosmerna komunikacija između klijenta i servera.

4.1.3 Klasa BnResourceManager

Apstraktna klasa `BnResourceManager`, kao i `BpResourceManager` nasleđuje interfejs `IResourceManager`. Međutim, jedina funkcija koja je realizovana u ovoj klasi je:

```
virtual status_t onTransact(uint32_t code, const Parcel& data, Parcel* reply,
int32_t flags
```

Posle slanja svake poruke od strane klijenta, server dobija poziv ove funkcije. Parametri funkcije su identični kao parametri prethodno opisane funkcije `IBinder::transact`. `BnResourceManager` klasa na osnovu koda transakcije interpretira sadržaj `Parcel` objekta. Rezultat njegove dekompozicije su apstraktni tipovi podataka poslani od strane korisnika koji predstavljaju parametre funkcije koja se poziva. Posle prikupljanja parametara vrši se poziv odgovarajuće funkcije realizovane u `ResourceManager` klasi i pisanje rezultata iste u drugi `Parcel` objekat zadužen za prenos odgovora do klijenta.

4.2 Klasa ResourceManager

Modul zadužen za osnovnu fukcionalnost rukovaoca resursima. Kako je komunikacija putem Binder radnog okvira apstrakovana u njenoj baznoj klasi `BnResourceManager`, klasa `ResourceManager` je zadužena samo za realizaciju glavnih funkcija rukovaoca. Pored realizacije funkcija definisanih interfejsom `IResourceManager`, klasa `ResourceManager` poseduje i dodatne privatne funkcije čiji opis sledi.

- `static ClientToken generateClientToken(const char* clientId) – "Hash"` funkcija koja generiše jedinstveni klijentski žeton na osnovu imena klijenta. Ova vrednost služi kao jedinstveni identifikator klijenta.
- `bool clientExists(ClientToken token) – Proverava da li korisnik sa datim tokenom već postoji.`
- `ResourceHolder* createClient(const char* name) – Kreira strukturu unutrašnje reprezentacije korisnika. Ime koristi za generaciju žetona, pomoću kog generisanu strukturu uvezuje u android::KeyedVector strukturu.`
- `static void instantiate() – Vrši inicijalizaciju platforme, evidenciju slobodnih resursa kao i registraciju Android servisa. Registracija servisa se vrši korišćenjem Android ServiceManager klase čija se jedinstvena instanca dobavlja putem sistemskog poziva android::defaultServiceManager() funkcije. Posle dobavljanja ServiceManager interfejsa, registracija servisa rukovaoca resursima se svodi na poziv funkcije addService gde se kao parametar prosleđuje`

jedinstvena instanca rukovaoca kao i jedinstvno ime koje služi kao identifikator servisa.

4.2.1 Klasa ResourceHolder

Klasa koja predstavlja serversku reprezentaciju korisnika resursa. Svaki registrovan korisnik dobija novu instancu ove klase čijij je jedinstveni identifikator žeton generisan na osnovu imena korisnika. U cilju evidencije prioriteta klijenata, svaki klijent je dužan da pri pri registraciji pruži nivo važnosti svoje aplikacije. Nivoi prioriteta su definisani enumeracijom:

```
enum ClientPriority{
    ePRIORITY_LOW,
    ePRIORITY_NORMAL,
    ePRIORITY_CRITICAL
};
```

Ovaj mehanizam pruža rukovaocu resursa mogućnost selektivne dodelje resursa. Pod selektivnom dodelom se podrazumeva da korisnik sa nižim prioritetom neće biti u mogućnosti oduzimanja resursa korisniku sa višim prioritetom. U konkretnom slučaju CHAL sloj Comedia srednjeg sloja uzima prioritet `ePRIORITY_CRITICAL`, čime obezbeđuje neprekidan rad srednjeg sloja a samim tim i DTV aplikacije koja je kritična za funkcionalnost platforme.

Klasa sadrži sledeće funkcije:

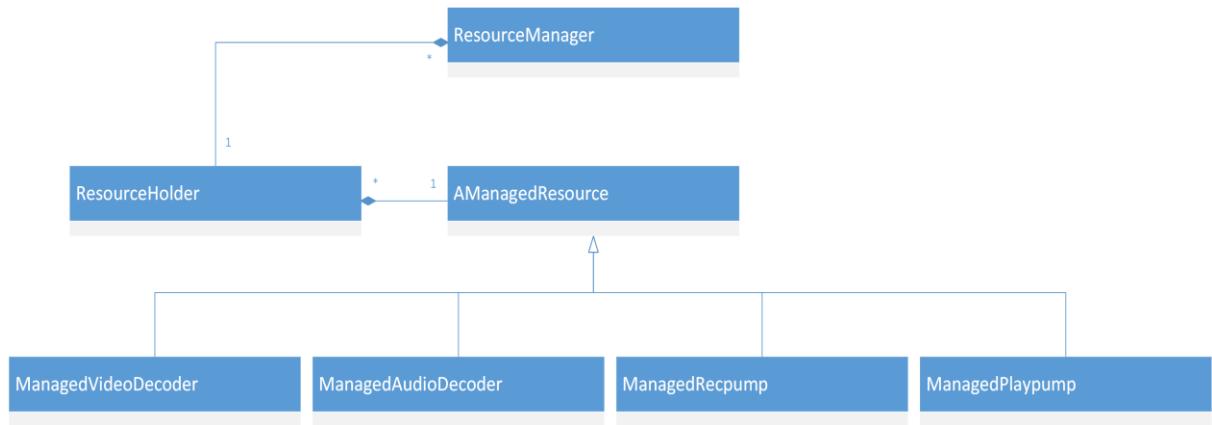
- `ClientToken getToken() const` – Vraća jedinstveni identifikator korisnika u vidu 32-bitnog broja.
- `const char* getName() const` – Vraća ime korisnika pod kojim je izvršena registracija.
- `void assignResource(AManagedResource* resource)` – Dodeljuje resurs klijentu. Referenca na dati resurs se smešta u listu posedovanih resursa u realizovana kao `android::Vector` klasa.
- `bool ownsResource(NEXUS_VideoDecoderHandle)` – Proverava da li klijent poseduje dati video dekoder.
- `bool ownsResource(NEXUS_AudioDecoderHandle)` – Proverava da li klijent poseduje dati audio dekoder.

- `bool ownsResource(NEXUS_RecpumpHandle)` – Proverava da li klijent poseduje datu Recpump instancu.
- `bool ownsResource(NEXUS_PlaypumpHandle)` – Proverava da li klijent poseduje datu Playpump instancu.
- `void unassignResource(NEXUS_VideoDecoderHandle)` – Oduzima dati resurs klijentu, realizovano izbacivanjem iz liste zazuetih resursa klijenta.
- `void unassignResource(NEXUS_AudioDecoderHandle)`
- `void unassignResource(NEXUS_RecpumpHandle)`
- `void unassignResource(NEXUS_PlaypumpHandle)`

4.2.2 Apstraktna klasa `AManagedResource`

Apstraktna klasa iz koje je svaki tip resursa izведен. Tipovi izvedenih resursa, kao i odnos između ostalih komponenti sistema ilustrovani su UML dijagramom klase na slici 4.3. Prilikom inicijalizacije platforme instanciraju se svi dostupni resursi i zajedno se smeštaju u listu resursa realizovanu kao `Android Vector`. Klasa `AManagedResource` sadrži sledeće funkcije:

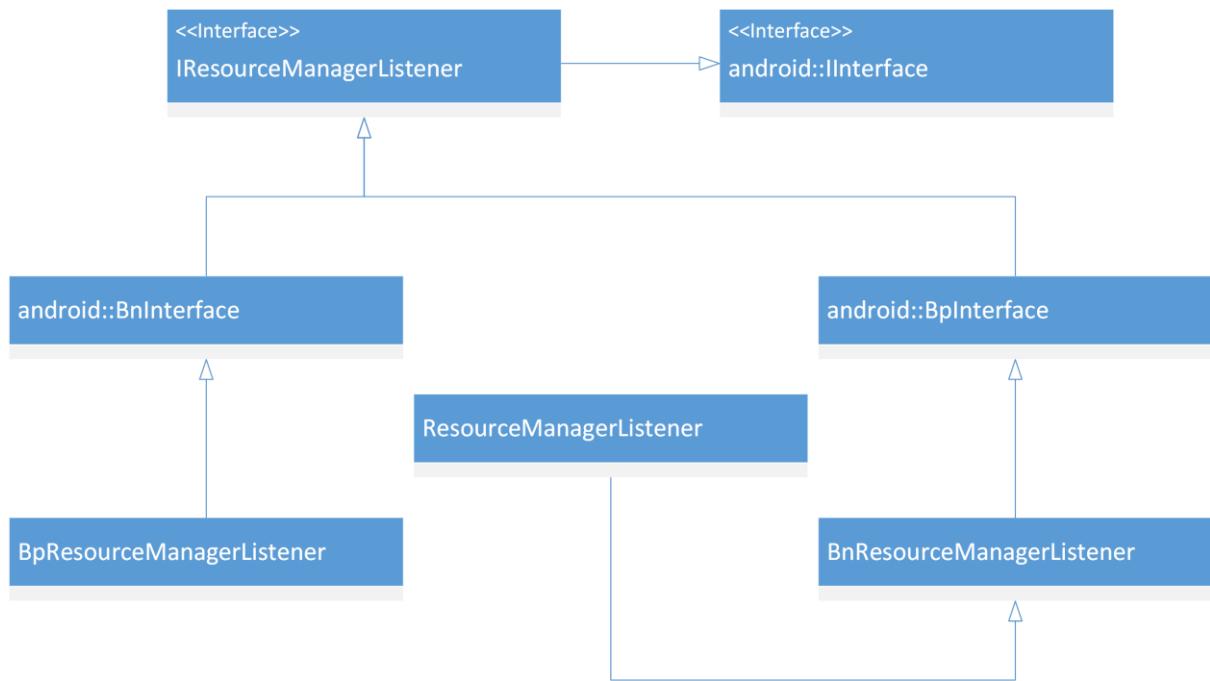
- `ClientToken getOwner() const` – Dobavlja trenutnog vlasnika resursa. Povratna vrednost ove funkcije je validna samo ukoliko vlasnik postoji.
- `bool isOwned()` – Proverava da li je resurs zauzet od strane nekog korisnika.
- `ResourceType getType() const` – Tip resursa u vidu celog broja. Koristi se radi realizacije osnovnog RTTI (engl. Run-Time Type Information) sistema.



Slika 4.3 Odnos osnovnih komponenti sistema rukovalca resursima

4.3 Asinhrona međuprocesna povratna funkcija

Asinhrona međuprocesna povratna funkcija je realizovana analogno komunikaciji u smeru klijent-server. Potreba za ovim mehanizmom javlja se na sistemima sa ograničenim resursima. Ukoliko ne postoji dovoljan broj dekodera za sve korisnike, korisnici sa višim prioritetom dobijaju prednost. U posmatranom slučaju postoje dva korisnika: Android multimedijalni sloj i Comedia DTV srendji sloj. Kako je osnovna namena platforme pružanje DTV usluga, viši prioritet dobija Comedia (odnosno CHAL).



Slika 4.4 Hierarhijska organizacija klasa međuprocesne povratne funkcije

Klase neophodne za realizaciju komunikacije u suprotnom smeru (i.e. server-klijent) ilustrovane su UML dijagramom na slici 4.4. Način rada pomoćnih klasa je identičan kao u prethodno opisanom sistemu, pa detaljniji opis u ovom slučaju nije neophodan.

4.3.1 Klasa ResourceManagerListener

Klase `ResourceManagerListener` služi kao posrednik između servera i povratnih funkcija klijenta. Kako svaki proces dobija različit adresni prostor, prosleđivanje adrese povratne funkcije drugom procesu je nemoguće, pa je uvođenje mehanizam poput ovog neophodno. Klase `ResourceManagerListener` sadrži sledeće funkcije:

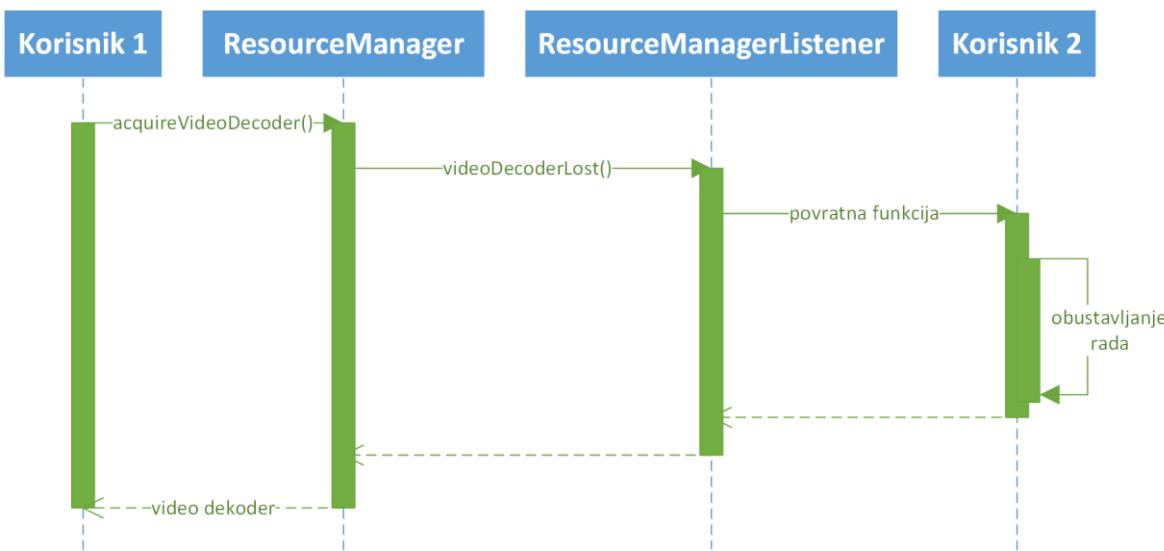
Povratne funkcije koje se pozivaju od strane servera:

- `void onVideoDecoderLost(NEXUS_VideoDecoderHandle handle)`
- `void onAudioDecoderLost(NEXUS_AudioDecoderHandle handle)`
- `void onPlaypumpLost(NEXUS_PlaypumpHandle handle)`
- `void onRecpumpLost(NEXUS_RecpumpHandle handle)`

Registracija povratnih funkcija korisnika za svaki tip resursa posebno:

- `void registerCallbackFunction(VideoCallbackFnc func)`
- `void registerCallbackFunction(AudioCallbackFnc func)`
- `void registerCallbackFunction(PlaypumpCallbackFnc func)`
- `void registerCallbackFunction(RecpumpCallbackFnc func)`

Navedene povratne funkcije predstavljaju mehanizam putem koga rukovalac resursima obaveštava datog korisnika o izgubljenom resursu.



Slika 4.5 Scenario oduzimanja resursa od korisnika sa nižim prioritetom

Slika 4.5 ilustruje scenario u kom prvi korisnik većeg prioriteta oduzima video dekoder koji je u upotrebi od strane korisnika 2. Putem međuprocesne povratne funkcije, korisnik 2 je obavešten o gubitku prava na resurs i dobija priliku da obustavi svoj rad na način koji neće narušiti konzistentnost sistema.

Registracija povratne funkcije kod rukovaoca resursima se vrši pozivom metode definisane u interfejsu `IResourceManager`:

```
ResourceManagerError registerCallback(android::sp<adroid::IBinder> callback);
```

Binder instanca se dobavlja putem funkcija `ResourceManagerListener::asBinder()` realizovane u Android klasi `IInterface` iz koje je klasa izvedena. Serijalizacija Binder objekata omogućena je direktno pisanjem pokazivača istih u `Parcel` objekte [6].

4.4 Klasa ResourceManagerClient

Jednostavna klasa koja služi kao enkapsulacija prethodno opisanih funkcionalnosti. Osnovna namena joj je instanciranje i inicijalizacija klijentskog klasas `BpResourceManager` i `ResourceManagerListener` putem kojih korisnik ima pristup funkcionalnostima rukovaoca resursima. Klasa sadrži sledeće funkcije:

- `IResourceManager* getResourceManager()` – Dobavlja interfejs Binder posredničke instance rukovaoca resursima. Prilikom instanciranja `ResourceManagerClient` klase, instanca Binder-a se dobavlja koristeći Android

ServiceManager klasu i njenu funkciju `getService` koja kao parametar uzima prethodno poznato ime servisa rukovaoca resursima.

- `ResourceManagerListener* ResourceManagerListener() – Dobavlja instancu klasu ResourceManagerListener putem koje korisnik može da registruje povratne funkcije od interesa.`

4.5 C omotač rukovaoca resursima

Radi omogućavanja dostupnosti uslugama rukovaoca resurisma širom sistema, dodat je modul C omotača oko C++ klasa `ResourceManagerClient` i `IResourceManager`. Ovaj modul je realizovan kao C++ dinamička biblioteka, sa javnim interfejsom definisanim u programskom jeziku C, čime je omogućen pristup uslugama rukovaoca modulima sistema poput CHAL, koji je takođe pisan u C-u. Modul se sastoji od sledećih funkcija:

- `ResourceManagerClientHandle ResourceManagerClient_new()`
- `void ResourceManagerClient_delete(ResourceManagerClientHandle handle)`
- `ClientToken`
`ResourceManagerClient_createClient(ResourceManagerClientHandle, const ResourceManagerClientConfig* config)`
- `void ResourceManagerClient_destroyClient(ResourceManagerClientHandle handle, ClientToken client)`
- `ResourceManagerError`
`ResourceManagerClient_getVideoWindowSettings(ResourceManagerClientHandle handle, ClientToken client, int windowId, VideoWindowSettings* settings)`
- `ResourceManagerError`
`ResourceManagerClient_setVideoWindowSettings(ResourceManagerClientHandle handle, ClientToken client, int window_id, VideoWindowSettings *settings)`
- `NEXUS_AudioDecoderHandle`
`ResourceManagerClient_acquireAudioDecoder(ResourceManagerClientHandle handle, ClientToken client)`
- `NexusServiceError`
`ResourceManagerClient_releaseAudioDecoder(ResourceManagerClientHandle handle, ClientToken client, NEXUS_AudioDecoderHandle handle)`
- `NEXUS_VideoDecoderHandle`
`ResourceManagerClient_acquireVideoDecoder(ResourceManagerClientHandle handle, ClientToken client)`
- `ResourceManagerError`
`ResourceManagerClient_releaseVideoDecoder(ResourceManagerClientHandle handle, ClientToken client, NEXUS_VideoDecoderHandle handle)`

- NEXUS_PlaypumpHandle
 ResourceManagerClient_acquirePlaypump(ResourceManagerClientHandle handle, ClientToken client)
- ResourceManagerError
 ResourceManagerClient_releasePlaypump(ResourceManagerClientHandle handle, ClientToken client, NEXUS_PlaypumpHandle handle)
- NEXUS_RecpumpHandle
 ResourceManagerClient_acquireRecpump(ResourceManagerClientHandle handle, ClientToken client)
- ResourceManagerError
 ResourceManagerClient_releaseRecpump(ResourceManagerClientHandle handle, ClientToken client, NEXUS_RecpumpHandle handle)
- ResourceManagerError
 ResourceManagerClient_acquireMosaicCell(ResourceManagerClientHandle handle, ClientToken client, int index, NEXUS_VideoWindowHandle* window, NEXUS_VideoDecoderHandle* decoder)
- ResourceManagerError
 ResourceManagerClient_releaseMosaicCell(ResourceManagerClientHandle handle, ClientToken client, int index)
- void
 ResourceManagerClient_registerCallbackFunction(ResourceManagerClientHandle handle, VideoCallbackFnc func)
- void
 ResourceManagerClient_registerCallbackFunction(ResourceManagerClientHandle handle, AudioCallbackFnc func)
- void
 ResourceManagerClient_registerCallbackFunction(ResourceManagerClientHandle handle, PlaypumpCallbackFnc func)
- void
 ResourceManagerClient_registerCallbackFunction(ResourceManagerClientHandle handle, RecpumpCallbackFnc func)

5. Ispitivanje i verifikacija

Verifikacija ispravnosti realizovanog programskog rešenja obavljena je kroz dve faze. Prva faza podrazumeva verifikaciju ispravnosti osnovnih funkcionalnosti rukovaoca u kontrolisanim uslovima korišćenjem namenske aplikacije dizajnirane specifično za ispitivanje rukovaoca. U drugoj fazi su prethodno verifikovane funkcionalnosti ispitane u realnim uslovima, gde su korisnici rukovaoca multimedijalni sloj Android platforme i aplikacija koja koristi Comedia DTV srednji sloj.

5.1 Verifikacija u kontrolisanim uslovima

U cilju ispitivanja osnovne funkcionalnosti rukovaoca, kreirana je aplikacija za ispitivanje u programskom jeziku C. Najpre je verifikovana kreacija klijenta rukovaoca u aplikaciji koja se se registruje pod imenom i prioritetom dobijenim putem argumenata komandne linije. U zavisnosti od sledećeg parametra komandne linije aplikacija izvršava jedan od dva ispitna scenarija.

Prvi scenario podrazumeva slanje zahteva za određenim video dekoderom, posle čega sledi simulacija zauzetosti aplikacije u vidu pauze od određenog broja sekundi i oslobođanja prethodno zauzetog dekodera. Pokretanjem dve instance iste aplikacije sa različitim prioritetima verifikovana su dva scenarija:

1. Prva aplikacija sa maksimalnim prioritetom zauzima video dekoder, dok druga aplikacija pokrenuta sa određenim kašnjenjem, pokušava da zauzme isti dekoder. Rukovalac resursima odbija zahtev druge aplikacije za dekoderom zbog nižeg prioriteta i aplikacija prestaje sa radom kontrolisan način.

2. Prva aplikacija sa minimalnim prioritetom zauzima video dekoder, dok druga aplikacija pokrenuta maksimalnim prioritetom pokušava da zauzme isti dekoder. Rukovalac resursima oduzima zauzet dekoder od prvog korisnika i obaveštava ga o gubitku istog putem realizovane međuprocesne povratne funkcije. Prva aplikacija prestaje sa radom na kontrolisan način bez narušavanja stabilnosti sistema, dok druga aplikacija započinje rad na predviđen način sa prvobitno zatraženim dekoderom.

Drugim scenarijom verifikovana je stabilnost sistema pokretanjem aplikacije u režimu intezivnog (engl. stress) ispitivanja. Posle pokretanja i registracije, aplikacija verificuje stabilnost sistema u dve faze:

1. U prvoj fazi, aplikacija dobavlja i oslobađa sve moguće resurse u određenom broju iteracija i ispisuje rezultate pokušaja na standardni izlaz.
2. U drugoj fazi, aplikacija pokušava da u zavisnosti od nasumične promenljive ili zauzme ili oslobodi nasumični resurs koji nije prethodno dobavljen.

5.2 Verifikacija u realnim uslovima

U svrhu ispitivanja rukovalca resursima u realnim uslovima kreirana je Android aplikacija koja koristi više VideoView [9] grafičkih komponenti za reprodukciju audio/video sadržaja. Kako je osnovna namena platforme reprodukcija DTV sadržaja, multimedijalnom Android sloju je dodeljen je niži prioritet od Comedia srednjeg sloja. Aplikacija za ispitivanje je pokrenuta i reprodukcija multimedijalnog sadržaja je započeta, posle određenog perioda, aplikacija se sakriva pritiskom na “back” dugme čime je prebačena u neaktivno stanje. Posle sakrivanja ispitne aplikacije, pokrenuta je glavna DTV aplikacija koja se oslanja na Comedia srednji sloj za svoju funkcionalnost. Resursi zauzeti od prethodne ispitne aplikacije su oduzeti zbog nižeg prioriteta i normalno funkcionisanje DTV aplikacije je ostvareno. Pri završetku rada DTV aplikacije, prethodno pokrenuta aplikacija je vraćena u fokus, gde je verifikovano da se reprodukcija multimedijalnog sadržaja završila na konzistentan način bez narušene stabilnosti aplikacije i čitavog sistema.

6. Zaključak

U ovom radu prikazano je jedno rešenje realizacije programske podrške rukovanja sistemskim resursima DTV uređaja na Android platformi.

Prilikom projektovanja programske podrške rukovaoca glavni zadatak je bio da se uspostavi konzistentan pristup deljenim resursima u višeklijentskom okruženju čime je osigurana stabilnost sistema. Uvođenjem novog sloja programske podrške izbegnuta je potrebna modifikacija rukovaoca fizičkim resursima (engl. Driver) prvobitno dizajniranih za rad u okruženju sa jednim klijentom. Sloj apstrakcije hardverskog okruženja (CHAL) srednjeg sloja Comedia, kao i deo multimedijalnog sloja Androida uspešno je prilagođen radu sa rukovaocem resursa.

Realizovana programska podrška ispitana je na digitalnom TV prijemniku u vidu STB platforme BCM97435VMS zasnovane na Android softverskom steku gde je izvršena verifikacija njene funkcionalnosti.

Pristup programskoj podršci omogućen je modulima pisanim u programskom jeziku C korišćenjem omotačke biblioteke postojećih C++ klasa, čime je omogućen pristup funkcionalnostima rukovaoca širom sistema.

U radu je realizovan samo osnovni koncept sistema za rukovanje resursima i ostavljen je prostor za dalja unapređenja. Jedna od osnovnih mana sistema je uska vezanost rukovaoca sa Nexus hardverskom programskom spregom. Dalji razvoj programske podrške bi podrazumevao uvođenje sloja apstrakcije fizičke arhitekture, i integracija realizovane programske podrške u jezgro Android platforme, čime bi se omogućila ponovna upotreba programske podrške na različitim fizičkim platformama zasnovanim na Android softverskom steku.

7. Literatura

- [1] Google Android, <http://www.android.com>
- [2] Kuzmanovic, N.; Maruna, T.; Savic, M.; Miljkovic, G.; Isailovic, D., "Google's android as an application environment for DTV decoder system," Consumer Electronics (ISCE), 2010 IEEE 14th International Symposium on , vol., no., pp.1,5, -10 June 2010
- [3] M. L. Murphy, *The Busy Coder's Guide to Android*, United States of America: CommonsWare, 2008.
- [4] D. A. Rusling, *The Linux Kernel*. 1999.
- [5] D. Hackborn "Introduction to OpenBinder", 2006.
- [6] T. Schreiber, "Android binder", Ph.D. dissertation, Ruhr University Bochum, 2011
- [7] Predavanja iz predmeta *Projektovanje namenskih računarskih struktura 2*, <http://www.rt-rk.uns.ac.rs/studijski-program-2009/ix-2009/pnrs2>
- [8] Vidakovic, M.; Teslic, N.; Maruna, T.; Mihic, V., "Android4TV: A proposition for integration of DTV in Android devices," Consumer Electronics (ICCE), 2012 IEEE International Conference on , vol., no., pp.437,438, 13-16 Jan. 2012
- [9] VideoView class overview
<http://developer.android.com/reference/android/widget/VideoView.html>