УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Милан Иванковић

ИНТЕГРАЦИЈА МОДУЛА ЗА НАДЗОР И КОНФИГУРАЦИЈУ УРЕЂАЈА ЗАСНОВАНОГ НА TR-069 ПРОТОКОЛУ ЗА АНДРОИД ПЛАТФОРМУ

МАСТЕР РАД

Нови Сад, 2018

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**:		Милан Иванковић	
Ментор, **МН**:		проф. Др Илија Башичевић	
Наслов рада, **НР**:		Интеграција модула за надзор и конфигурацију уређаја заснованог на TP-069 протоколу за Андроид платформу	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		2018	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		7/62/15/7/25/0/0	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кqучне речи, **ПО**:		ДТВ, СТБ, АЦС, ЦПЕ	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		Овај рад представља побољшања постојећег решења за интеграцију модула за конфигурацију и надгледање уређаја базираног на TP-069 комуникационом протоколу у форми системског сервиса за *set-top-boks* (СТБ) уређаје базиране на Андроид платформи. Решење је прилагођено за две нове платформе и проширено специфичним функцијама за *Android Nougat*. Надограђен је комуникацијски слој између TP-069 сервиса и Андроид апликација. Ажурирање програмске подршке или инсталирање апликације, прилагођено је за *Android Nougat*. У оквиру TP-069, клијентског сервиса додат је СТУН клијентски модул, модул за отпремање датотека са исписима, и имплементирани су специфични параметри по захтјеву оператора.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	проф. Др Мирослав Поповић	
	Члан:	проф. Др Иван Мезеи	Потпис ментора
	Члан, ментор:	проф. Др Илија Башичевић	

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**:		Milan Ivanković	
Mentor, **MN**:		Ilija Bašičević, PhD	
Title, **TI**:		Integration of a TR-069 based module for monitoring and configuration of devices for Android platform	
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**:		2018	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/62/15/7/25/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		STB, DTV, ACS, CPE	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		In this paper we present an integration of a device configuration and monitoring module based on the TR-069 protocol (also known as CPE Wan Management Protocol) as a system service for Android-based set-top-boxes (STBs). This is an update of an existing solution with specific functions for Android Nougat platforms and some specific customer requirements. The main functions that were added are OTA image uprgreade, STUN client module, communication service, Log Upload, specific parameter implementation.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	Miroslav Popović, PhD	
	Member:	Ivan Mezei, PhD	Menthor's sign
	Member, Mentor:	Ilija Bašičević, PhD	

Zahvalnost

Zahvaljujem se institutu RT-RK na pruženoj mogućnosti za realizaciju ovog rada.
Takođe, zahvaljujem se mentoru dr Iliji Bašičeviću, stručnom saradniku Darku Dejanoviću i mojim kolegama na savetima i pomoći tokom izrade završnog rada. Posebno se zahvaljujem porodici i devojci na pruženoj podršci, motivaciji i brizi tokom dosadašnjeg školovanja.

SADRŽAJ

SPISAK SLIKA

SPISAK TABELA

SKRAĆENICE

DTV - *Digital Television,* Digitalna televizija

STB - *Set-Top Box*, Uređaj za prijem TV signala

DVB - *Digital Video Broadcasting,* Standard za digitalnu televiziju

JNI - *Java native interface,* sprega Java programskog jezika i C koda

LAN - *Local Area Network*, Lokalna mreža

WLAN - *Wireless Local Area Network*, Bežična lokalna mreža

API - *Application program interface,* Aplikativna programska spre

TR-069 - *Tehnical Report 069,* Tehnički izvještaj 069

ACS - *Auto Configuration Server*, Server koji podržava TR-069 protokol

CPE - *Customer-Premises Equipment*, Klijentski uređaj koji podržava TR-069 protokol

AIDL - *Android Interface Definition Language,* Jezik za definisanje programske sprege u Android operativnom sistema

STUN - *Simple Traversal of User Datagram Protocol* – Protokol o komunikaciji klijenta i poslužioca

NAT - *Network Address Translator* – Prevodilac mrežnih adresa

1. Uvod

Broj uređaja potrošačke elektronike ubrzano raste u poslednjih par godina. Pored kvantitativnog rasta broja uređaja, dolazi do tehnološkog napretka u njihovom razvoju, čime je omogućeno još više funkcionalnosti za krajnje korisnike. Veći zahtevi korisničkih uređaja doprinose i sve većim zahtevima za fizičkim resursima. Multimedijalni uređaji, kao što su telefoni i televizori, već su uveliko zasnovani na internetu. Uređaji sa ovakvim potrebama zahtevaju od operatera (eng. Service provider) komunikacionih i IPTV (eng. Internet Protocol Television) usluga da obezbedi besprekoran rad svojih mreža, kako bi se korisnicima omogućila što bolja usluga.

Razvoj mrežnih tehnologija i brzine razmene podataka je dovelo do mogućnosti da se mreže sastavljene od uređaja potrošačke elektronike kontrolišu i nadgledanju sa udaljenih stanica. Poslužioci (eng. Servers) uređaja potrošačke elektronike su sistemi zasnovani na računaru, čiji je osnovni zadatak komunikacija sa uređajima sa jedne strane i manipulacija podacima sa druge strane.

U ovom radu je opisan problem, koncept i realizacija integacije modula za nadzor i konfigiraciju uređaja zasnovanog na TR-069 protokolu. Funkcionalnost je ostvarena na Android Nougat baziranom prijemniku za digitalnu televiziju. Dat je prikaz realizacije sprege Java programskog jezika i C koda potrebne za realizaciju funkcionalnosti.

Bitna karakteristika digitalne televizije je kvalitet signala (eng. QoS – Quality of Service). Digitalna televizija je omogućila prenošenje stabilnog signala visoke rezolucije. Iako bolji, i digitalni signal je podložan raznim tipovima šuma. Kod IPTV-ja, zahtevi za kvalitetom signala su još izraženiji zbog mehanizma prenošenja signala. Kvalitet signala kod IP televizije zavisi od propusnosti mreže koju korisnik ima, mrežne opreme i smetnji u prenosu. Usled jako gustog saobraćaja na mreži, dolazi do gubitka na kvalitetu signala, gube se paketi, što kvari korisnički

doživljaj [11]. Menjanje kanala u uslovima gustog saobraćaja, biva produženo i do nekoliko sekundi. Takođe, mogući su kvarovi na digitalnim prijemnicima, kako na programskoj podršci tako i na fizičkim delovima uređaja. Stoga, operaterima je potrebna povratna informacija o dešavanjima na uređajima kako bi mogli pravovremeno da reaguju na promene. Informacija o kvalitetu koju pruža korisnik prilikom anketiranja od strane operatera često nije pouzdana s obzirom na subjektivnost, različiti korisnici dostavljaju različite povratne informacije o kvalitetu. Potreban je objektivniji pristup, gde operater dobijaju informacije o kvalitetu signala sa krajnjeg uređaja. Povratnu informaciju ka operateru omogućavaju klijenti programske podrške na digitalnim prijemnicima koji su u mogućnosti da preuzmu podatke: o signalu koji digitalni televizijski prijemnik dobija, o statusu svake od komponenti digitalnog prijemnika i da te podatke prosledi poslužiocu (eng. Server), kao i da omogući oporavak od greške u toku rada ili ažuriranje programske podrške digitalnog prijemnika. Ukoliko na uređaju postoji aktivna veza sa internetom, povratne informacije se šalju periodično; u suprotnom, podaci se čuvaju u memoriji uređaja sve dok se ne ostvari veza.

Jedan od klijenata koji omogućava navedene odlike je definisan prema standardizovanom TR-069 protokolu (eng. Technical Report 069) [1] koji garantuje siguran prenos. Protokol definiše postojanje TR-069 poslužioca koji prikuplja podatke i TR-069 klijenta koji se nalazi na krajnjem korisničkom uređaju. Za svaki od uređaja postoji posebno definisan model podataka (eng. Data Model) koji jednoznačno, putem parametara, definiše uređaj na kom se TR-069 klijent nalazi. Za digitalne prijemnike, taj model podataka nosi naziv TR-135 (eng. Technical Report 135) [2]. Takođe, standard dozvoljava pravljenje sopstvenog modela podataka uz praćenje standardom precizno definisanih pravila.

Problem se javlja prilikom ugrađivanja TR-069 klijenta u korisnički uređaj. Svaki uređaj je specifičan i ima svoj model podataka, što znači da je za svaki uređaj potrebno uložiti napor da bi se klijentu dostavile fizičke vrednosti parametara na uređaju. Digitalni televizijski prijemnici se razlikuju prema tipu signala koji prihvataju, a samim tim, skup parametara koji ih opisuje su različiti. Dodatni problem je što u praksi svaki proizvođač definiše sopstvene parametre koji nisu deo standarda. Na primer, TR-135 modelom podataka je definisano preko 300 parametara. Za svaki od ovih parametara potrebno je podržati dobavljanje i prosleđivanje korišćenjem Aplikativne programske sprege (eng. API- Application Programming Interface). Vreme potrebno za pisanje prilagodnih funkcija za preuzimanje ili postavljanje vrednosti parametara direktno je srazmerno veličini modela podataka. Veliki broj ovih funkcija sadrže sličan programski kod, međutim, aktivnost dodavanja ovih funkcija je manuelan i dugotrajan posao podložan greškama. Takođe, ukoliko dođe do izmene modela podataka u toku integracije, a to je vrlo čest slučaj,

potrebno je uložiti dodatan napor da bi se kod ažurirao. Održavanje jednom napisanog koda je veoma teško.

Rad je sačinjen od sedam poglavlja. Prvo poglavlje sadrži osnovne podatke o radu i opis rada, odnosno kraći uvod.

Drugo poglavlje sadrži opis i namene Android platforme, koncept digitalne televizije, opis standarda i programske podrške za konfiguraciju i nadzor uređaja.

Treće poglavlje daje kratak opis početnog rešenja, ograničenja koja postavlja i pregled slojeva sistema koji opisuju predlog rešenja, realizacije i prikaza funkcionalnosti modula za konfiguraciju i nadzor uređaja.

Četvrto poglavlje sadrži detaljan opis programskog rešenja, odnosno realizacije funkcija modula za konfiguraciju i nadzor uređaja koji je zasnovan na Android platformi.

Peto poglavlje je sačinjeno od opisa provera koje su urađene nakon realizacije funkcionalnosti. U ovom poglavlju su opisani i dobijeni rezultati, kao i testne platforme.

Šesto poglavlje sadrži kratak pregled onoga što je urađeno u ovom radu i kakvi su dalji pravci razvoja.

U sedmom poglavlju je dat spisak korišćene literature tokom izrade ovog rada.

2. Teorijske osnove

U ovom poglavlju dati su opis i namena Android platforme, koncept digitalne televizije, kao i opis modula za nadzor i konfiguraciju uređaja zasnovanog na Android platformi, odnosno opis osnovnih karakteristika TR-069 protokola.

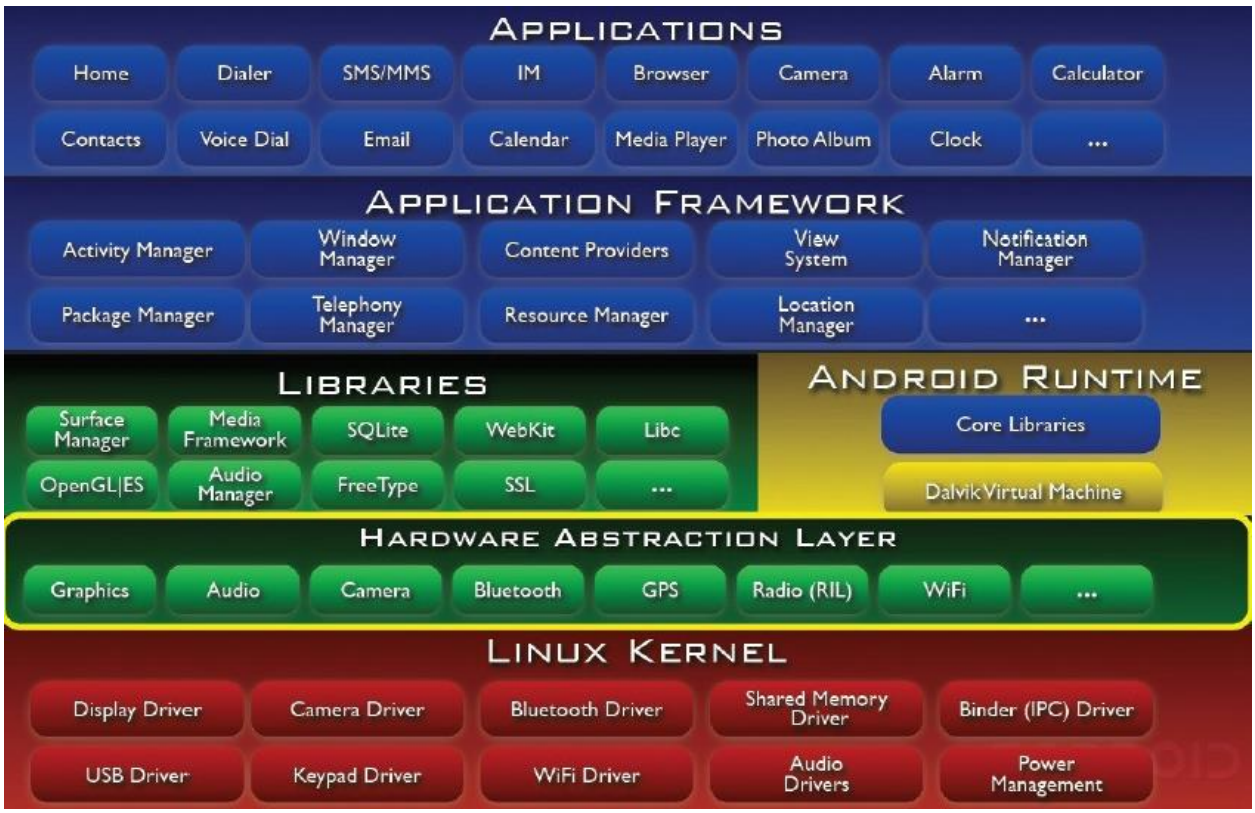
2.1 Android platforma

Android je prva otvorena, kompletna i besplatna programska platforma za mobilne telefone koja je sve više zastupljena i kao platforma za namenske uređaje čije su veličine memorije i procesorska moć ograničene (tablet računari, STB, TV…) [3]. Nove generacije pametnih TV zasnovanih uređaja sa Android platformom se sve više koriste od strane pružaoca usluga (eng. provider), odnosno operatera televizije zasnovane na internet protokolu (eng. Internet Protocol TeleVision - IPTV).

Za operatere bitan aspekt je kvalitet pružene usluge svojim kranjim korisnicima. Iz ovog zahteva je nastala potreba za nadgledanjem kvaliteta signala (eng. Quality of Service - QoS). Ovakav problem se rešava realizacijom proširenja programske podrške za TR-069 protokol uređaja sa Android operativnim sistemom. Proširivanje ovim funkcionalnostima nije jednostavan i brz postupak i zahteva složenu integraciju.

Na slici 2.1 je prikazana arhitektura Android platforme. Android stek može se podeliti u različite slojeve a to su:

- operativni sistem
- srednji sloj (engl. middleware)
- aplikativni sloj



Slika 2.1 - Arhitektura Android programskog steka

Linux jezgro (eng. Linux Kernel) - Android platforma ne predstavlja Linux distribuciju, ali je zasnovan na njegovom jezgru [5]. Linux jezgro je korišćeno kao industrijski dokazano, stabilno i u stalnom razvoju. Upravlja dodelom radne memorije, poseduje mogućnost upravljanja procesima, poseduje razdvojenost fizičkog sloja od programa i operativni sistem otvorenog koda (eng. Open Source). Upravljački blokovi uvedeni u okviru Android-a su:

- Binder – Komunikacija internih procesa (eng. Inter Process Communication - IPC),
- Ashmem – deljena memorija između procesa,
- Upravljački programi za upravljanje potrošnjom energije,
- Upravljački programi za ispise,
- Alarm i
- Upravljački programi za oslobađanje memorije.

Izvršno radno okruženje (eng. Android Runtime) – sastoji se od osnovnih biblioteka koje omogućavaju da se Android aplikacije pišu u Java programskom jeziku i Dalvik virtuelne mašine pomoću koje je moguće pokretanje Java .class / .jar datoteke posle prevođenja u .dex datoteku. Dalvik virtualna mašina se koristi zbog mogućnosti simuliranja uređaja sa sporim procesorima i relativno malom memorijom [5].

Biblioteke za apstrakciju fizičkog sloja (eng. Hardware Abstraction Libraries) – razdvajaju Android od fizičkog sloja i definišu spregu koju upravljačke jedinice implementiraju.

Okruženje za razvoj aplikacija (eng. Application Framework) – Java klase i sprege pružaju podršku Android aplikacijama i povezuju Android aplikacije sa nativnim slojem.

Aplikacije (eng. Applications) – pisane u programskom jeziku Java. Kompleksnije aplikacije koriste nativni sloj pomoću JNI poziva. Aplikacije mogu biti:

• Aktivnosti (eng. Activity) – grafički element koji najčešće odgovara jednom ekranu,

• Servisi (eng. Services) – uslužni procesi koji se izvršavaju u pozadini,

• Dobavljač sadržaja (eng. Content Provider) – elementi koji omogućuju deljenje podataka u sistemu,

• Primaoci emitovanih signala (eng. Broadcast Receivers) – elementi koji prihvataju emitovane signale (slaba baterija i slično).

JNI omogućava da se iz programskog jezika Java pozivaju nativne C funkcije. Jedan od razloga upotrebe JNI-a su bolje performanse, jer je nativni kod brži od Java koda. Takođe pomoću JNI sloja moguće je pristupiti sistemskim resursima, odnosno bibliotekama pisanim u programskom jeziku C. Mane su sledeće:

• Neoprezno programiranje prilagodnog sloja JNI može da izazove greške koje mogu da destabilizuju ceo sistem,

• Gubitak platformske nezavisnosti i

• Ne radi automatski oslobađanje memorije (eng. garbage collection), ako je zauzeta memorija u JNI-u, ona se mora osloboditi.

Ključna reč, koja označava da je metoda nativna, je native. Sve nativne metode su implementirane kao funkcije u dinamičkoj biblioteci. Da bi virtualna mašina prepoznala nativnu funkciju u dinamičkoj biblioteci, ona poseduje odgovarajući potpis u prototipu. Prototipovi nativnih funkcija se generišu pozivom javah alata, koji je standardni deo Java SDK. Ovaj alat kao ulazni parametar prima pun naziv klase, deklaraciju nativnih metoda i na osnovu njih generiše .h datoteku sa prototipovima svih nativnih funkcija.

Android servis je komponenta koja se izvršava u pozadini i koja najčešće nema interakciju sa korisnikom. Postoje dve vrste servisa:

• Sistemski servisi i

• Aplikativni servisi.

Sistemski servisi postoje u uređaju i do njih se dolazi preko odgovarajućih rukovaoca (menadžer, eng. manager), dok se aplikacioni servisi naknadno instaliraju i njih pokreću aplikacije ili primaoci emitovanih signala.

Postoji mnogo sistemskih servisa kao što su alarm, audio, kopiranje, rad sa mrežom, obaveštenja, lokacija i slično. Veza između aplikacije i sistemskog servisa se ostvaruje pomoću

menadžera, gde menadžer sadrži sve javne metode sistemskog servisa, koje klijentske aplikacije mogu da pozovu.

Lokalni servis (eng. Local Service) i udaljeni servis (eng. Remote Service) su dve podvrste aplikativnih servisa. Lokalni servisi se izvršavaju u istom memorijskom prostoru gde i aplikacija koja ih je pozvala. Postoje dva načina komunikacije sa lokalnim servisom:

- Metode *startService()* i *stopService()*, gde metoda *startService()* kreira servis i poziva njegove metode *onCreate()*, pa *onStartCommand()*, a metoda *stopService()* poziva *onDestroy()* metodu servisa, nakon čega se servis gasi,
- Metoda *bindService()* se može koristiti ako nema IPC metoda, već se samo startuje i zaustavlja servis.

Realizacija lokalnog servisa se vrši u tri koraka:

1. Kreiranje klase,
2. Prijava u manifestu i
3. Pisanje klijentskog koda.

Servisi se mogu i automatski startovati prilikom uključivanja uređaja. Udaljeni servisi se izvršavaju u posebnom memorijskom prostoru i komunikacija sa njima se odvija posredstvom IPC mehanizma. Redosled operacija kod udaljenog servisa:

1. Klijenti se vezuju na udaljeni servis,
2. Klijenti pozivaju metode servisa i
3. Klijenti se odvajaju od udaljenog servisa.

U Androidu *IPC* je implementiran korišćenjem *Binder* podsistema. *Binder* omogućuje komunikaciju među procesima i zasniva se na klijent-server arhitekturi. Klijent poziva metode servera. Procedura pisanja udaljenog servisa:

1. Definisanje *AIDL* datoteke sa definicijom metoda servisa,
2. Realizacija udaljenog servisa na osnovu *AIDL* datoteke,
3. Prijavljivanje servisa u manifestu i
4. Realizacija klijentskog koda:
    a) Vezivanje za servis,
    b) Poziv metode iz *AIDL* sprege i
    c) Odvajanje klijenta od servisa.

AIDL je deklarativni jezik za opis metoda servisa, zasnovan na IDL jeziku. Uglavnom se definišu sprege koje će implementirati serverski deo, odnosno servis. Sintaksa je slična Java sintaksi, sa sledećim podržanim tipovima:

- svi Javini primitivni tipovi,
- String,

• List,

• Map i

• CharSequence.

Jedna sprega je implementirana od strane jednog servisa. Ako su klijent i server različiti entiteti, potrebno je da se ista *AIDL* datoteka nalazi u oba entiteta, na istoj lokaciji. Implementacija udaljenog servisa - *onBind()* je ključna metoda za komunikaciju. Povratna vrednost ove metode je instanca *IBinder* klase. U njoj se kreira klasa naslednica klase *IRemoteServiceExample.Stub*, koja je mašinski generisana klasa. Nasleđivanje se vrši kako bi se realizovale sve metode navedene u *AIDL* sprezi. Ukoliko je posao u njoj dugotrajan on se realizuje preko *AsyncTask* klase. Prijava servisa u manifestu – Servis se prijavljuje unutar application oznake (eng. tag), kao *Service tag*. Obavezan atribut je *android:name* (puno ime servisa). Intent filter je podoznaka koji definiše akcije koje pokreću servis. Povezivanje klijenta na servis – Pozivom metode *bindService(Intent, Connection, Flag)* povezuju se klijentska aplikacija i servis. Ona nasleđuje *ServiceConnection* klasu koja poseduje dve povratne metode *onServiceConnected()* i *onServiceDisconnected(). onServiceConnected()* se poziva nakon uspostave veze sa servisom, a poziv *onServiceDisconnected()* nastupa prilikom prekidanja veze sa servisom. Povratni poziv (eng. Callback) – *Binder* omogućava povratni poziv što omogućava servisu pozive metoda na klijentu. Ako je posao koji servis obavlja dugotrajan, na serveru se kreira *AsyncTask* i kontekst izvršavanja se vraća klijentu pozivaocu. Klijent saznaje o završetku posla tako što ga server obaveštava povratnim pozivom. Realizacija ovog mehanizma zahteva dodavanje još jedne *AIDL* datoteke i kod klijenta i kod servera, koja će sadržati metodu koju će server pozvati kada se posao završi.

2.2 Digitalna televizija

Digitalna televizija predstavlja prenos audio i video zapisa, kao i dodatnih informacija u digitalnom formatu. Prednosti digitalne televizije u odnosu na analognu su pre svega kvalitetniji zvuk i slika. Takođe, digitalna televizija omogućuje prenos više TV servisa na jednoj frekvenciji i uopšte veću količinu informacija. Digitalni TV prijemnici uglavnom poseduju i pristup internetu, kao i mogućnost izvršavanja mnogobrojnih aplikacija pisanih u Javi, HTML-u i drugim programskim jezicima, što dovodi do postojanja interaktivne televizije. Takođe, kao posledica veće količine informacija koje se prenose tokom podataka u digitalnoj televiziji gledalac je u mogućnosti da bira između različitih video ili audio zapisa, jezika prevoda na jednom TV servisu, kao i da koristi TV aplikacije poput programskog vodiča ili ličnog video snimača. Digitalni televizijski prijemnik [6] je uređaj koji vrši konverziju televizijskog signala sa zemaljske, kablovske ili satelitske mreže sa ciljem prikaza sadržaja na standardnim LCD (engl.

Liquid Crystal Display), LED (engl. Light emitting diode) ili analognim televizorima. Prijemnik može biti ugrađen u TV uređaj ili odvojen uređaj tj. STB (engl. STB – Set Top Box) [7].

Koncept STB se pojavljuje 60-tih godina 20. veka kao uređaj za prevođenje UHF (engl. Ultra High Frequency) frekvencija na VHF (engl. Very High Frequency) frekvencije. Istorijski, prvi digitalni STB uređaji su primali signal sa kablovske mreže. Uvedeni su da nadoknade nedostatke izazvane razlikama između emitovanog i kablovskog signala [8]. Međutim, u digitalnoj eri uloga digitalnog STB uređaja nije samo prijem signala, već uređaj koji ima većinu osobina personalnog računara i multimedijalnog uređaja [9].

Osnovne funkcionalnosti STB uređaja su:

• Obrada emitovanog signala – Sposobnost da se obradi emitovani DVB signal predstavlja osnovnu funkcionalnost STB uređaja. Emitovani signal je u analognom obliku sa opsegom signala kanala od 8MHz. STB uređaj mora da odabere opseg radio frekvencije (engl. RF - Radio Frequencies) koja je vezan za odabrani kanal.

• Sigurnost – Obezbeđivanje zaštite signala od neovlašćenog pristupa. STB uređaj dekriptuje signal na osnovu PID-a (engl. Packet Identifier) koji definiše vezu između paketa i kanal. ECM (engl. Entitlement Control Messages) i EMM (engl. Entitlement Management Messages) paketi sa dodatnim CA nosiocima podataka se dobavljaju u toku demultipleksiranja i šalju CA modulu na potvrdu. Ako CA modul ne potvrdi da je sadržaj dostupan korisniku signal se ne dekriptuje, pri čemu je zadovoljena sigurnost na STB uređaju.

• Obrada audio i video podataka – Audio i video paketi se pomoću demultipleksera razdvajaju. MPEG2 video signal je enkodovan u Y/Cr/Cb formatu koji se pretvara u format koji se može prikazati na monitoru.

• Interakcija sa korisnikom – Korisnik može da odabere sadržaj koji želi da gleda pomoću daljinskog upravljača. Da bi se poboljšala usluga, operaterima se šalje povratna informacija o kanalima koji korisnik gleda.

2.3 Arhitektura TR-069 protokola

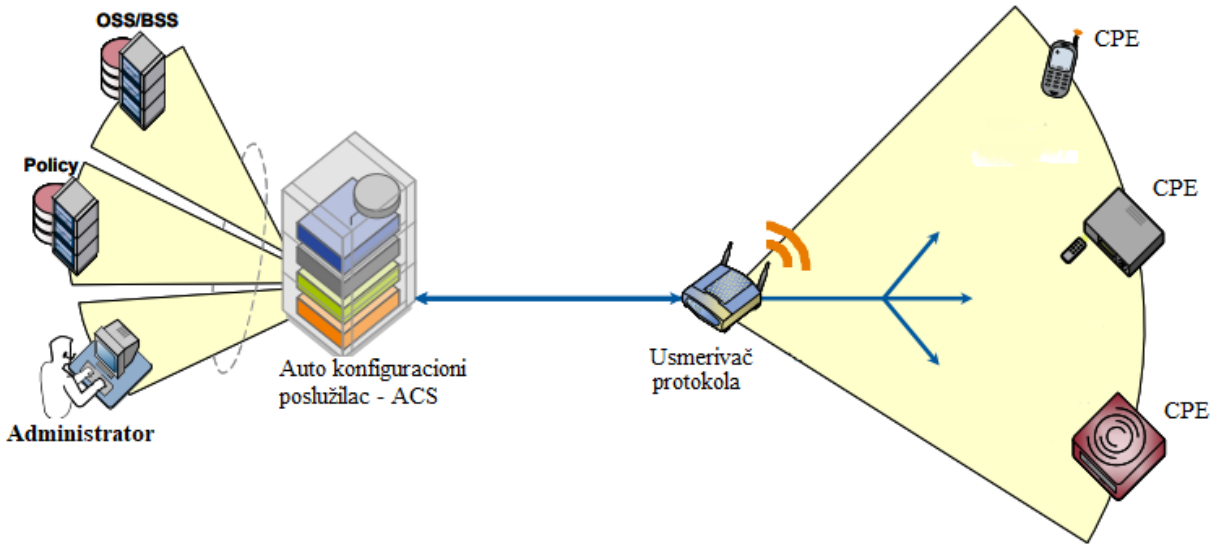
TR-069 (Tehnical Report 069) je napravljen od strane Širokopojasnog foruma (eng. Broadband forum) [1] i nazvan je CPE WAN upravljački protokol (eng. CWMP, Customer Premises Equipment Wide Area Network Management Protocol). Ovaj protokol se nalazi na ISO OSI aplikativnom nivou, i njegova funkcionalnost je da obavlja upravljanje i nadzor nad uređajima koji se nalaze kod krajnjih korisnika. CWMP definiše dva tipa uređaja: krajnji korisnički uređaj (eng.CPE, Customer Premises Equipment) i automatsko-konfiguracioni poslužilac (eng. ACS, Auto-Configuration Server). Korišćenjem ovog protokola, veza se uvek uspostavlja između krajnjeg uređaja i ACS poslužioca. Protokol se zasniva na dvosmernoj

SOAP/HTTP vezi. TR-069 protokolom je opisan samo način prenosa podataka između navedenih struktura u mreži, dok su definicije podataka koji se prenose date u modelima podataka pridruženih TR-069 standardu. TR-135 [2] model podataka namenjen je STB uređajima, kao i TR-106 [10] koji definiše osnovne podatke sistema potrebne za uspostavu TR-069 veze. Činjenica da se TR-069 zasniva na HTTP protokolu i da je nezavisan od tipa uređaja donele su mu široko polje primene. Neke od osnovnih funkcionalnosti koje pruža TR-069 protokol su:

- Automatska konfiguracija i dinamičko omogućavanje usluga,
- Rukovanje programskom podrškom krajnjeg uređaja,
- Nadgledanje statusa i performansi krajnjeg uređaja i njegova dijagnostika.

TR-069 protokol je našao svoju primenu u velikom broju namenskih uređaja. Sve ove platforme imaju jednu potrebu, a to je centralizovanje, nadgledanje i upravljanje datim jedinicama. Veliki broj statističkih analiza svoje funkcionalnosti zasnivaju na TR-069 protokolu. Korišćenjem ovog protokola moguće je izmeriti mnoge statistike, između ostalog:

- Merenje ponuđenog kvaliteta usluge,
- Statistiku gledanosti TV programa,
- Geografsku raspoređenost CPE uređaja i mnoge druge.



Slika 2.2 - : Primer okruženja TR-069 protokola

Na slici 2.2 predstavljen je primer okruženja TR-069 protokola. CPE uređaji mogu biti različitog tipa, a pritom ne postoje razlike u protokolu po kom se prenose podaci do ACS-a. Razlikuju se samo modeli podataka za svaki od datih CPE-ova. TR-069 standard definiše niz protokola koji se moraju koristiti i na CPE i na ACS strani prenosa podataka. Široko polje

primene TR-069 standarda je omogućilo primjenljivost na različite tipove uređaja bez potrebe za menjanjem načina prenosa poruka definisanih protokolom.

Za praćenje kvaliteta signala digitalne televizije postoji mnogo parametara. Ako se uzme u obzir broj korisnika koji bi koristili digitalne prijemnike, za praćenje kvaliteta signala bilo bi potrebno dosta resursa, što bi bilo veoma skupo, a možda čak i nemoguće ako bi broj korisnika dostizao nekoliko miliona. Zbog toga se moralo naći rešenje koje će dati što jasniju sliku o kvalitetu digitalnog signala. Studije Quo Vadis [11] i Mosquito [12] su predložile rešenje gde se od svih parametara izdvajaju tri i to odnos signal šum (eng. Signal to Noise Ratio – SNR) i bitska greška (eng. Bit Error Rate – BER) kao najvažniji i modulaciona greška (eng. Modulation Error Rate - MER. Studije su obavljene na osnovu DVB-T signala. BER predstavlja broj bitskih grešaka podeljen sa ukupnim brojem bita u posmatranom vremenskom periodu. Šum, ometanje, distorzija ili greške u sinhronizaciji dovede do bitske greške. BER nema mernu jedinicu i najčešće se izražava u procentima. SNR je mera odnosa željenog signala i pozadinskog šuma u njegovom prenosu i jedna je od najbitnijih mera prenosa i kvaliteta informacija. Generalizovano se može posmatrati kao odnos korisnih informacija prema irelevantnim i oštećenim informacijama. Jedinica za SNR je dB. Ove vrednosti se prate u parametrima modela TR-135.

CWMP definiše protokol koji koristi nekolicine standardnih internet protokola. Nivoi protokol steka prikazani su u tabeli 2.1.

Nivo protokola	Protokol
6	CPE/ACS aplikativni protokol
5	RPC
4	SOAP
3	HTTP(S)
2	SSL/TLS
1	TCP/IP

Tabela 2.1 - Prikaz korišćenih protokola po nivoima protokol steka

Na dnu protokol steka je standardni TCP / IP protokol. Zatim slede SSL / TSL protokoli koji obezbeđuju enkripciju standardnog mrežnog protokola (eng.Hypertext Transfer Protocol, HTTP) protokola koji se naziva HTTP Secure (https). Simple Object Access Protocol (SOAP) protokol je zasnovan na EXtensible Markup Language (XML) i koristi se za enkodovanje poziva udaljenih procedura (eng. Remote Procedure Call, RPC). Naznačene RPC metode su definisane od strane CWMP. Sam vrh protokol steka predstavljaju aplikacije koje koriste CWMP na strani ACS-a i CPE-a. Aplikacije nisu deo CPE WAN protokola upravljanja. CPE u svakom trenutku može inicijalizovati veza sa ACS-om koristeći predefinisani jednoobrazni lokator resursa

(eng. Uniform Resource Locator, URL) ACS-a. Sesija inicijalizovana od strane CPE-a započinje uspostavom konekcije sa ACS-om i slanjem Inform RPC metode. Razlozi uspostave sesije mogu biti sledeći:

- Prvi put kada se CPE uspostavi vezu za pristupanje mreži;
- Pri ponovnom pokretanju CPE-a;
- Kada istekne vreme zadato PeriodicInformInterval parametrom – periodično prijavljivanje na ACS;
- Kada CPE primi zahtev za uspostavu veze od strane ACS-a;
- Kada se promeni URL ACS-a;
- Kada se promeni vrednost parametara na čije praćenje je ACS pretplaćen;
- Kada se uspešno ili neuspešno izvrši preuzimanje ili otpremanje datoteka;
- Kada je sesija neuspešno okončana;

TR-069 protokol takođe definiše i mehanizam za uspostavu veze (eng. Connection Request) kojim je ACS u mogućnosti da inicira prevremeno prijavljivanje CPE-a na ACS, odnosno prevremeno slanje Inform RPC zahteva. Definisani su različiti mehanizmi za uspostavu veze:

- TR-069 Connection Request – primenjuje se kada su ACS i CPE u istoj lokalnoj mreži (eng. Local Area Network, LAN), odnosno kada su IP adrese ACS-a i CPE-a u istom domenu.
- STUN - primenjuje se kada ACS i CPE nisu u istoj LAN mreže pri čemu IP adrese nisu iz istog domena. Način korišćenja STUN mehanizma opisan je u okviru TR-181 specifikacije.
- Extensible Messaging and Presence Protocol (XMPP) - primenjuje se istim slučajevima kao i STUN mehanizam. Definisan je u okviru verziji 5 TR-069 protokola.

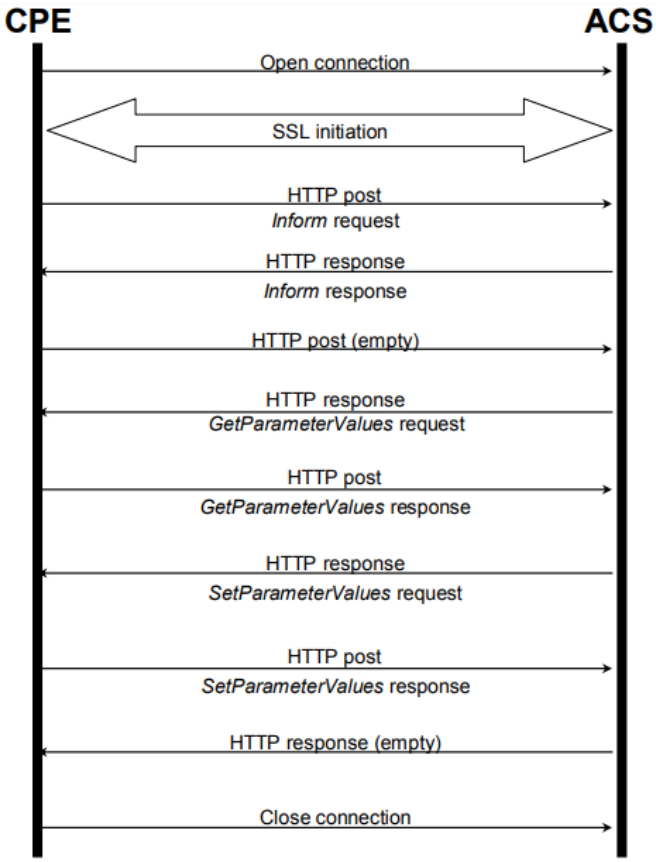
Skup RPC metoda je definisan TR-069 specifikacijom i njihov broj se menja kroz verzije specifikacije, kako dodavanjem novih tako i izbacivanjem postojećih iz prethodnih verzija. Funkcionalnosti koje su omogućene upotrebom RPC metoda su sledeće:

- Prosleđivanje strukture modela podataka od strane CPE-a.
- Prosleđivanje i postavljanje vrednosti parametara iz modela podataka.
- Prosleđivanje i postavljanje vrednosti atributa parametara iz modela podataka.
- Preuzimanje i otpremanje datoteka sa navedene URL adrese.
- Ponovo pokretanje CPE-a sa ili bez vraćanja CPE-a u inicijalno stanje.

ACS u svakom trenutku može zahtevati početak sesije upotrebom mehanizma za zahtevanje veze (eng. Connection Request). Mehanizam za zahtevanje veze koristi HTTP GET tip poruke koja se šalje na URL koji je određen od strane CPE-a. CPE mora izvršiti autentifikaciju ACS-a ili će u suprotnom zahtev biti odbijen. Ukoliko je autentifikacija uspešna

CPE, šalje HTTP poruku sa statusom "200" (OK) ili "204" (No Content). Nakon uspešne autentifikacije i poslatog odgovora CPE mora u roku od trideset sekundi da inicijalizuje sesiju na standardan način, u suprotnom ACS može ponovo da pokuša sa mehanizmom za zahtev konekcije.

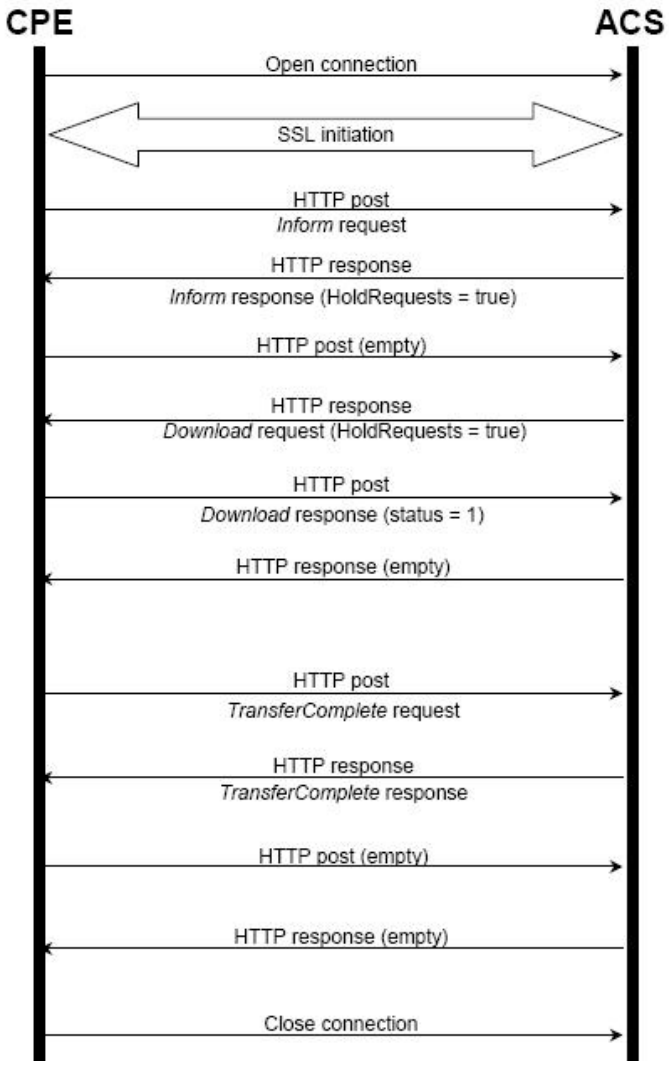
TR-069 uvek sesija započinje slanjem Inform RPC metode od strane CPE-a. Sve poruke unutar sesije su HTTP POST ili HTTP RESPONSE poruke koje predstavljaju odgovore na HTTP zahteve. ACS nakon primljene Inform poruke mora odgovoriti sa InformResponse porukom koja sadrži zaglavlje sa definisanom verzijom CWMP-a koju podržava ACS. Ukoliko se verzija koju podržava ACS ne poklapa sa verzijom na CPE-u, CPE mora koristi željenu verziju koju zahteva ACS. Potom CPE šalje praznu HTTP POST poruku nakon čega ACS može početi sa slanjem svojih zahteva za CPE. Jedna jednostavna TR 069 sesija ilustrativno je prikazana na slici 2.3.



Slika 2.3 – Jednostavna TR-069 sesija

Dok ima zahteva za CPE, ACS mora u SOAP porukama zahteva da postavlja zaglavlje HoldRequests koje sadrži logičku tačnu vrednost. U svakom trenutku ACS može pomoću zaglavlja HoldRequests obavestiti CPE da postoje zahtevi i CPE mora biti spreman da u narednoj poruci obradi zahtev od ACS-a. Kada nema više zahteva za CPE, ACS šalje praznu HTTP POST poruku ili postavlja *HoldRequests* zaglavlje na negativnu logičku vrednost. CPE

može slati svoje zahteve ACS-u ukoliko *HoldRequests* ima negativnu vrednost (nepostojanje *HoldRequests* zaglavlja se tretira kao da je postavljeno na negativnu vrednost) ili ukoliko je ACS poslao praznu HTTP poruku. TR-069 ACS sesija se zatvara ukoliko ACS i CPE nemaju više zahteva. Na slici 2.4 prikazan je primer TR-069 sesije kada ACS postavlja *HoldRequests* zaglavlje na logički tačnu vrednost i kada CPE šalje zahteve ka ACS-u.



Slika 2.4- TR-069 sesija

Jedna od osnovnih RPC metoda jeste Inform metoda. Inform metoda se prosleđuje isključivo od strane CPE-a ka ACS-u i njenom upotrebom CPE obaveštava ACS o svom prisustvu. Inform zahtev predstavlja začetak sesije unutar koje ACS i CPE prosleđuju različite RPC zahteve u jednom ili drugom smeru. Sesija inicijalizovana od strane CPE-a započinje uspostavom veze sa ACS-om i slanjem Inform RPC metode. Protokol zahteva da CPE poznaje jednoobrazni lokator resursa (eng. Uniform Resource Locator, URL) ACS-a kako bi iniciranje sesije bilo moguće. U tabeli 2.2 i 2.3 su prikazane neke od obaveznih/opcionih RPC metoda koje svaki sistem koji implementira TR-069 protokol treba da podrži.

Ime metode	Opis	CPE	ACS
GetRPCMethods	Spisak svih RPC metoda koji podržava CPE	Obavezna	Neobavezna
SetParameterValues	Postavljanje vrijednosti zastavljenih parametara navedenih vrijednosti	Obavezna	Obavezna
GetParameterValues	Dostavljanje vrednosti zahtevanih parametara	Obavezna	Obavezna
GetParameterNames	Dostavljanje imena svih parametara i objekata iz modela podataka	Obavezna	Obavezna
SetParameterAttributes	Postavljanje zahtevanih atributa parametra na navedene vrednosti	Obavezna	Neobavezna
GetParameterAttributes	Dostavljanje vrednosti atributa zahtevanog parametra	Obavezna	Neobavezna
AddObject	Dodavanje novog objekta u modelu podataka koji podržava CPE	Obavezna	Neobavezna
DeleteObject	Uklanjanje postojećeg objekta u modelu podataka koji podržava CPE	Obavezna	Neobavezna
Reboot	Ponovno pokretanje CPE-a	Obavezna	Neobavezna
Download	Preuzimanje navedenih datoteka sa određenog URL-a	Obavezna	Obavezna
Upload	Otpremanje navedenih datoteka sa određenog URL-a	Neobavezna	Neobavezna
FactoryReset	Ponovno pokretanje CPE-a sa tim da se CPE vraća u inicialno stanje	Neobavezna	Neobavezna
GetQueuedTransfers	Dostavljanje zahteva o prethodno poslatom Download ili Upload zahtevu	Neobavezna	Neobavezna
GetAllQueuedTransfers	Dostavljanje zahteva o svim poslatim Download ili Upload zahtevima koji još nisu izvršeni	Neobavezna	Neobavezna
ScheduleInform	Zahtev da CPE pošalje Inform metodu u određenom vremenskom periodu	Neobavezna	Neobavezna
SetVouchers	Postavljanje vaučera na CPE	Neobavezna	Neobavezna
GetOptions	Dostavljanje informacije trenutno postavljenih na CPE-u i njihovo stanje	Neobavezna	Neobavezna

Tabela 2.2 - CPE RPC metode, opis, CPE odgovor, ACS poziv

Ime metode	Opis	ACS	CPE
GetRPCMethods	Spisak svih RPC metoda koje podržava ACS	Obavezna	Neobavezna
Inform	Inicijalizacija sesije, poruka sadži podatke na osnovu kojih se svaki CPE identifikuje na ACS-u	Obavezna	Obavezna
TransferComplete	Obaveštava ACS o završetku prenosa datoteke koji je inicijalizovan preko Download ili Upload zahteva od strane ACS-a	Obavezna	Obavezna
AutonomousTransferComplete	Obaveštava ACS o završetku prenosa datoteke koji nije zahtevan od strane ACS-A	Obavezna	Neobavezna
RequestDownload	CPE zahteva da ACS pošalje Download zahtev sa navedenim podacima	Neobavezna	Neobavezna
Kicked	Omogućava ACS-u da navede URL na koji bi Interent pretraživač trebao da se preusmeri	Neobavezna	Neobavezna

Tabela 2.3 - ACS RPC metode, opis, CPE poziv, ACS odgovor

Model podataka (eng. Data Model) ima veoma važnu ulogu u realizaciji TR-069 protokola. Model podataka je jedini element protokola koji je zavisan od tipa uređaja i kao takav jednini način da se uređaji razlikuju i identifikuju u okviru TR-069 protokola. Dok TR-069 specifikacija propisuje način komunikacije između klijentske strane i poslužioca, TR-106 specifikacija [10] opisuje način na koji se definišu modeli podataka, odnosno sadrži *XML* šemu (eng. XML Schema Definition, XSD) koja definiše pravila kreiranja modela podataka. Pored toga TR-106 specifikacija propisuje i skup elementarnih parametara kako bi protokol pravilno funkcionisao i kako bi se svaki CPE mogao jedinstveno identifikovati u ekosistemu. Široko polje primjene TR-069 standarda je omogućila primjenljivost na različite tipove uređaja bez potrebe za mijenjanjem načina prenosa poruka definisanih protokolom. Neki od modela podataka su navedeni u tabeli 2.4.

Model podataka	Tip uređaja
TR-089	Internet konvertori protokola i usmjerivači
TR-104	VoIP uređaji
TR-106	Osnovni model podataka za TR-069
TR-135	IPTV i STB uređaji
TR-140	NAS uređaji
TR-143	Model korićen za dijagnostiku CPE uređaja
TR-157	Model komponenata datog CPE uređaja
TR-181	Model mrežnih podataka uređaja
TR-192	FAP uređaij

Tabela 2.4 - Modeli podataka podržani TR-069 protokolom

Model podataka predstavlja skup parametara organizovanih u obliku strukture podataka tipa stabla (eng. Tree). Stablo je organizovano tako da parametar uvek predstavlja njen list dok su čvorovi stabla objekti koji kao svoje potomke mogu da sadrže nove objekte ili parametre. Objekti mogu biti definisani kao višestruki, što znači da može postojati više instanci jednog objekta. Dodavanje i uklanjanje višestrukih objekata je moguće izvršiti upotrebom RPC poziva. Svaki parametar pored svoga naziva i vrednost ima definisan svoj tip i ograničenja, npr. maksimalnu ili minimalnu vrednost parametra. Takođe parametar sadrži i atribute čijom upotrebom se može podesiti politika praćenja rada uređaja kao i prava pristupa. Definisani su sledeći atributi:

•Nivo obaveštavanja – definišu se tri nivoa obaveštavanja.

1. Onemogućeno obaveštavanje - promena vrednosti parametra se ACS-u nikada ne prijavljuje osim na eksplicitan zahtev ACS-a.
2. Pasivno obaveštavanje - promena vrednosti parametra se ACS-u prijavljuje pri prvoj narednoj sesiji.
3. Aktivno obaveštavanje - promena vrednosti parametra se ACS-u prijavljuje čm se desila, pri čemu je potrebno inicirati sesiju.

• Lista prava pristupa – sadrži listu pretplatnika koje imaju prava da menjaju vrednost parametra. Lista se ne odnosi na ACS koji uvek ima pravo upisa vrednosti ukoliko je parametar definisan kao upisiv.

Modeli podataka definisani TR-106 modelom podataka, su stukturirani kao stablo, u kom postoje sledeći tipovi čvorova:

- Korenski čvor,
- Objekat,
- Višestruki objekat,
- Parametar.

Korenski čvor omogućava objedinjavanje više modela podataka u jedno stablo ukoliko je to potrebno za određeni uređaj. Objekat predstavlja čvor u stablu koji ne može biti list, već se samo koristi za bolje struktuiranje podataka. Višestruki objekat je takođe objekat, s tim što se može više puta instancirati i pritom mu se pridružuje indeks pomoću koga mu se može jednoznačno pristupiti. Parametri predstavljaju listove u stablu i samo njima je pridružena vrednost. Pored vrednosti parametrima se pridružuje i niz atributa pomoću kojih se vrši konfiguracija uređaja od strane ACS servera. Atributi su:

- Nivo notifikacije (Notification), definiše pod kojim okolnostima CPE treba da javi promenu vrednosti ACS-u;
- Lista pristupa (AccessList), definiše listu entiteta koji smiju da promijene vrednost parametra.

Pored atributa, modelom se definišu i podešavanja koja se ne mogu promeniti od strane ACS servera. Ta podešavanja su:

- Mogućnost upisa vrednosti parametra,
- Ograničenja vrednosti parametara, definisana enumeracijom ili graničnim vrijednostima parametara,
- Obaveza slanja vrednosti parametra prilikom iniciranja TR-069 sesije,
- Tip podataka datog parametra,
- Mogućnost nadgledanja promjene vrednosti parametra u realnom vremenu,
- Početna podrazumevana vrednost parametra,
- Ograničenje broja višestrukih čvorova kod istog pretka,
- Da li je čvor definisan u tom profilu modela podataka,
- Jedinica mere date vrednosti parametra.

Tipovi podataka parametara koje definiše standard su:

- Niz karaktera (String),
- Celobrojni (Integer),
- Celobrojni prošireni (Long),
- Neoznačeni celobrojni (Unsighend Integer),
- Neoznačeni prošireni celobrojni (Unsigned Long),
- Logički (Boolean),
- Vremenski (dataTime),

- Binarni (Base64),
- Heksadecimalni (hexBinary).

TR-069 protokol definiše i tipove operacija koje se mogu izvršiti nad stablom, a to su:

- Dodavanje i brisanje čvora je moguće samo ukoliko je dati čvor višestruki objekat,
- Dodavanje vrednosti je moguće samo ukoliko je čvor parametar,
- Postavljanje vrednosti je moguće ukoliko je čvor parametar i moguća je promena njegove vrendosti,
- Postavljanje vrednosti atributa nad parametrom.

Takođe se uz svaki čvor pridružuje semantički opis tog čvora definisan stringom koji je razumljiv čoveku. Važno je naglasiti da se prilikom pristupa bilo kom čvoru u stablu mora navoditi njegova cela putanja od korenskog čvora, gde se kao separator između imena čvorova navodi karakter tačka. Primer je:

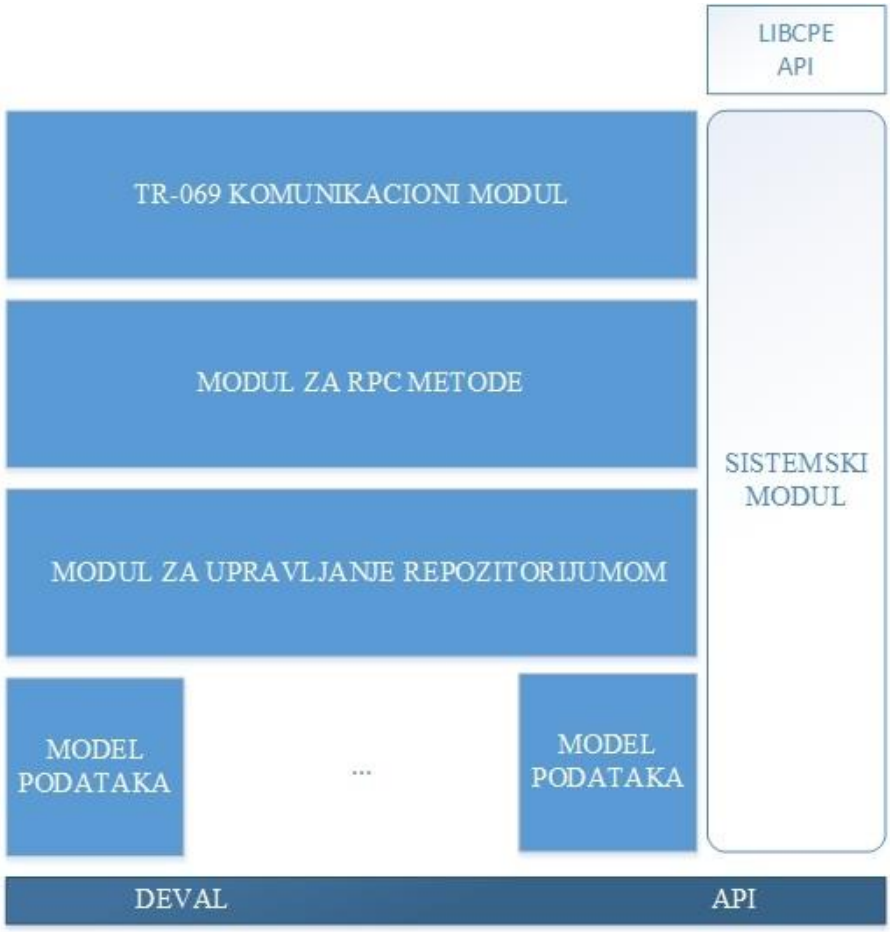
- KorenskiČvor.Objekat.VišestrukiObjekat.Indeks.Objekat.Parametar

3. Koncept rešenja

Ovo poglavlje sadrži analizu zadatog problema i koncept rešenja koji je prikazan i objašnjen kroz primer rada modula za nadzor i konfiguraciju uređaja. Prikazano je postojeće rešenje od koga je počeo razvoj sadašnjeg. Rešenje je identično prilagođeno za dve nove platforme sa BG4-ct Synaptic - 2GB RAM čipom i Amlogic 805x - 1GB RAM čipom. Rešenje je nadograđeno određenim specifičnim funkcinalnostima kao i onih propisanih standardom. Nadograđen je komunikacioni sloj između TR-069 servisa i Android aplikacija. Dobavljanje i ažuriranje programske podrške, odnosno instaliranje aplikacije prilagođeno je Android Nougat-u. Ažuriranje programske podrške je implementirano na dva načina, ažuriranje prilikom sledećeg ponovnog pokretanja uređaja (eng. Reboot) ili nasilno ažuriranje (eng. Force upgrade). Nakon skidanju date programske podrške uređaj se sam ugasi, primeni izmene i opet pokrene. U okviru TR-069 servisa dodat je STUN klijentski modul. Implementirani su parametri iz TR-135 modela i TR-181 modela podataka. Implementirano je prikupljanje i podizanje ispisa (eng, Logs) sa strane klijenta na stranu poslužioca. Pored dodavanja novih funkcinalnosti, izvršena je i reformacija i prilagođavanje postojećih.

3.1 TR-069 klijent – postojeće rešenje

TR-069 klijent koji je korišćen u razvoju dinamičkog sistema za prevođenje TR-069 klijentske biblioteke razvijan je na institutu za Računarsku tehniku i računarske komunikacije. Njegova struktura i moduli od kojih je sačinjen prikazani su na slici 3.1.



Slika 3.1 - Moduli u TR-069 klijentskoj biblioteci

Sistemski modul pruža prilagodni sloj koji pušta u rad i zaustavlja TR-069 klijentske biblioteke. Isto tako, u okviru njega se obavlja zauzimanje resursa, učitavaju modeli podataka i uspostavlja se veza sa poslužiocem. On predstavlja osnovni modul u CPE klijentu i omogućava dostupnost funkcionalnosti biblioteke drugim aplikacijama.

TR-069 komunikacioni modul je zadužen za komunikaciju između poslužioca i uređaja. U sklopu modula su funkcionalnosti:

- Čekanje i obrada događaja koji će pokrenuti TR-069 sesiju;
- Izvođenje TR-069 sesije;
- Izvođenje prenosa datoteka;
- Omogućavanje sprege za rukovanje programskom podrškom krajnjeg uređaja; (eng. Firmware upgrade);
- Parsiranje HTTP zaglavlja i proveru ispravnosti;
- Pakovanje, validacija i parsiranje SOAP struktura;
- Pozivanje odgovarajuće RPC metode;
- Slanje i prijem HTTP poruka.

Prilikom prijema svakog HTTP paketa vrši se parsiranje HTTP zaglavlja, a potom provera ispravnosti SOAP paketa. Ukoliko je primljen, SOAP zahtev određuje koji tip RPC metode se poziva i vrši se parsiranje SOAP paketa i poziv RPC metode sa dobijenim podacima. Ukoliko dobije zahtev za vrednosti nekog od parametara ili objekata, pristupa modulu za rukovanje repozitorijumima preko RPC metoda i vraća ih ACS poslužiocu.

Modul za rukovanje repozitorijumima (eng. Data Repository Engine) obezbeđuje CRUD (eng. Create, Read, Update, Delete) funkcionalnosti nad repozitorijumima definisanim na osnovu TR-069 podržanih modela podataka. Uređenje struktura u ovom modulu omogućuje lako proširivanje novim modelima podataka. Svaki model podataka se prilikom inicijalizacije biblioteke instancira kao stablo i poveže u listu repozitorijuma, tako da se dodavanje novih repozitorijuma može izvršiti bez izmena u kodu biblioteke. Da bi se pristupilo parametru ili objektu u repozitorijumu, potrebno je obezbediti putanju do parametra/objekta definisanog u modelu podataka i instancu napravljenog repozitorijuma.

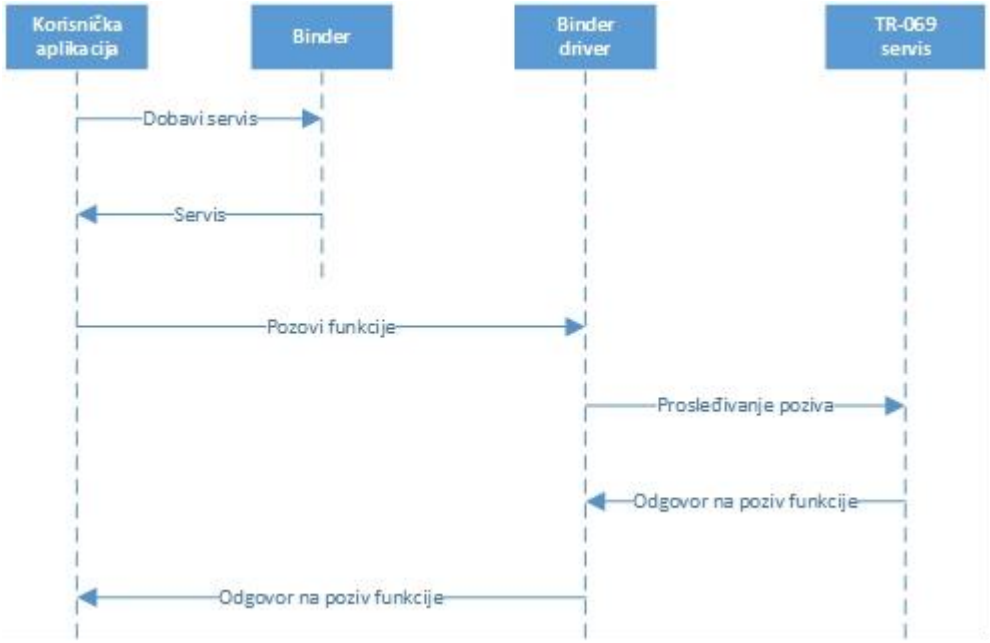
Moduli podatka (TR-106, TR-135, TR-181 repozitorijum) predstavljaju prilagodni API konkretnih krajnjih uređaja prema biblioteci. Zasnovani su na odgovarajućim modelima podataka. Ovi moduli predstavljaju funkcije koje su potrebne za integraciju TR-069 klijenta u uređaj. U njima treba da se nalaze minimalni skupovi funkcija za svaki parametar i za svaki objekat sa više instanci.

LibCpe API i *DEVAL*(eng Device Abstraction Layer) API datoteke koje se isporučuju korisnicima biblioteke i sadrže kompletan skup funkcija kojima se može iskoristiti potpuna funkcionalnost TR-069 klijentske biblioteke. *DEVAL* API je unija funkcija za sve repozitorijume koji postoje na konkretnom uređaju, odnosno za sve modele koji su potrebni za modelovanje uređaja u koji se TR-069 klijentska biblioteka ugrađuje. *LibCpe* API je skup funkcija za pokretanje i zaustavljanje rada klijentske biblioteke.

3.2 TR-069 komunikaciona sa Android aplikacijama

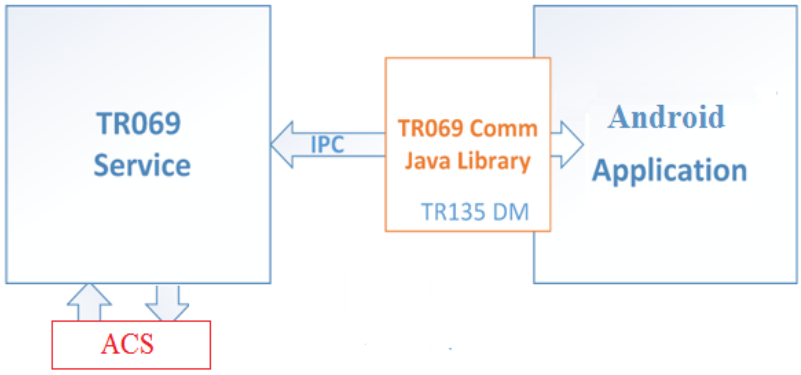
Android definiše svoj mehanizam za međuprocesnu komunikaciju, *Binder*. Ovim mehanizmom je omogućeno da aplikativna komponenta iz jednog Android procesa može da pozove metodu druge aplikativne komponente koja se nalazi u drugom Android procesu. Da bi se ovo izvelo potrebno je definisati aplikativnu programsku spregu kojom je opisano koje sve funkcionalnosti nudi aplikativna komponenta. U tu svrhu Android je ponudio jezik za definisanje sprege, *AIDL* (eng. Android Interface Definition Language). Komunikacija među procesima je orijentisana kao klijent – poslužilac. *Binder* mehanizam je iskorišćen za realizaciju TR-069

komunikacionog mehanizma. Na slici 3.2 predstavljena je komunikacija između krajnje korisničke aplikacije i TR-069.



Slika 3.2- Komunikacija između korisničke aplikacije i TR-069 servisa

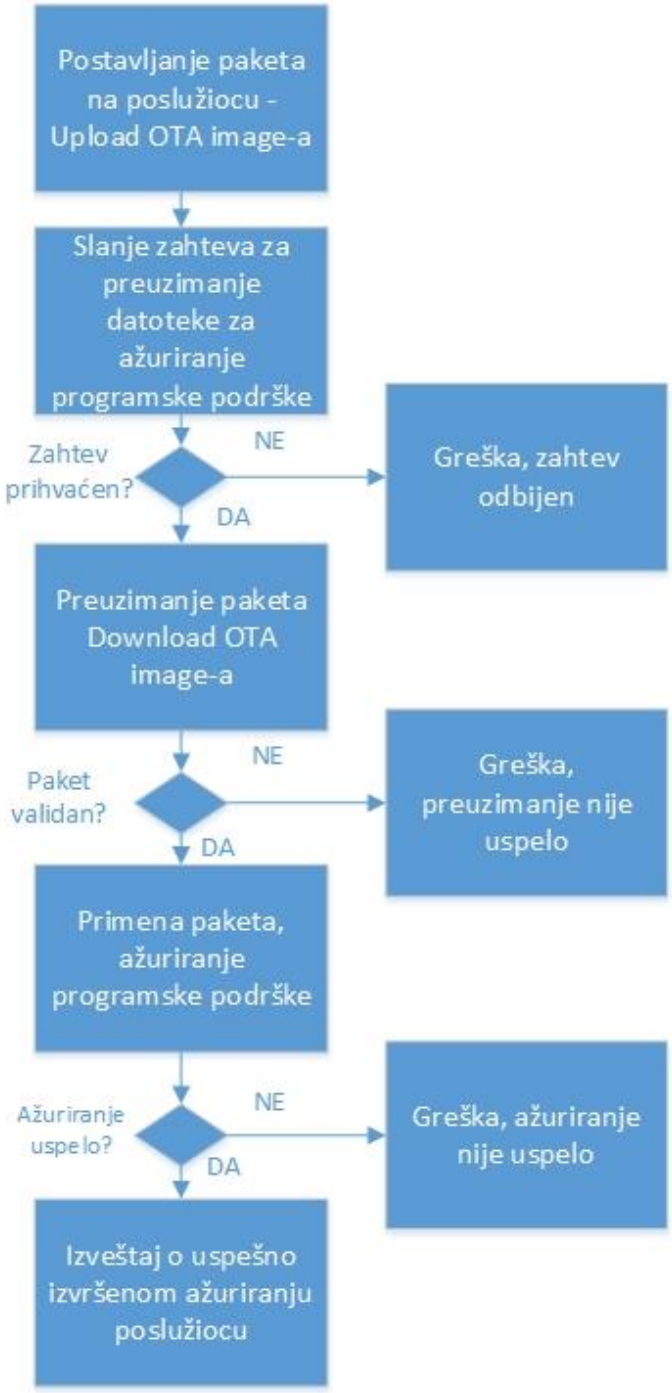
AIDL je sprega za povezivanje koja omogućava komunikaciju između Android aplikacije i TR-069 agentskog servisa. Deklaracija deljenih funkcija se nalaze u *AIDL* datoteci. U zasebnoj Android aplikaciji ili servisu se postavljaju specifični STB parametri za srednji sloj (eng. Middleware) iz TR-135 modela. Implementacija je izvršena na ovaj način jer su informacije o TR-135 parametrima dostupne samo u aplikaciju za emitovanje IPTV DTV sadržaja. Korisnik Android aplikacije se povezuje na TR-069 agentski servis preko TR069ClienServiceCommunacion biblioteke, odnosno pomoću *AIDL* sprege. Na slici 3.3 je prikazana arhitektura TR-069 agentskog servisa sa Android aplikacijom. *TR069ClienServiceCommunacion* je sprega za povezivanje koja omogućava komunikaciju između Android aplikacije i TR-069 servisa. Android aplikacija koristi sistemske resurse uređaja i obezbeđuje funkcionalnost krajnjem korisniku.



Slika 3.3- Primer komunikacije TR-069 servisa i Anroid aplikacije

3.3 TR-069 ažuriranje programske podrške

Funkcionalnost ažuriranja programske podrške je jedna od funkcija šireg sistema Set-Top-Box uređaja i jedna od osnovnih funkcinalnosti propisanih TR-069 protokolom. Da bi se postigla što jasnija slika o celom procesu ažuriranja, kompletan proces obuhvata putanju od poslužioca do krajnjeg uređaja, uključujući prijem paketa za ažuriranje, njegovo slanje do krajnjeg uređaja i primenu na krajnjem uređaju. Opšti izgled procesa ažuriranja dat je na slici 3.4.



Slika 3.4 – Opšti prikaz ažuriranja programske podrške

Rukovanje poslužiocem u konkretnom slučaju, omogućeno je korišćenje Web tehnologije, tj. administrator može pristupiti ACS-u preko Web-stranica. Administrator može da vidi spisak STB uređaja koji su aktivni i nalaze se na mreži. Paket se nakon postavljanja na poslužilac može poslati bilo kojem aktivnom uređaju. Izborom uređaja i potvrđivanjem da želimo da se izvrši ažuriranje programske podrške, zapravo započinje proces ažuriranja.

Poslužilac šalje ciljanom uređaju zahtev za preuzimanje paketa za ažuriranje programske podrške. Slanje tog zahteva ne nastupa odmah, već u prvoj sledećoj komunikacionoj sesiji između CPE-a i ACS-a. Vremenski razmak između sesija je definisan u modelu podataka. Trajanje jedne sesije nije određeno, ono zavisi od količine podataka koja treba da se razmeni između CPE-a i ACS-a. Prilikom slanja zahteva za preuzimanje paketa, RPC metoda *Download* se poziva sa određenim argumentima. Najvažniji argumenti ove metode su:

- Tip paketa (datoteke) - da li je u pitanju paket za ažuriranje, Web-sadržaj ili konfiguraciona datoteka
- Lokacija resursa - (eng. URL-Unified Resource Locator) putanja odakle CPE, treba da preuzme paket za ažuriranje
- Veličina paketa - veličina datoteke iskazana u bajtima.
- Ime paketa na strani CPE-a - putanja do direktorijuma na platformi i ime koje treba biti dodeljeno paketu kada se uspešno preuzme (npr / koren / neki_direktorijum / paket_za_ažuriranje)
- Odlaganje - vreme u sekundama kojim se označava koliko vremena treba tačno da prođe između trenutka kada CPE primi zahtev i trenutka kada krene da preuzima paket sa zadate putanje

CPE na ovaj zahtev odgovara obaveštenjem da li je proces uspeo ili nije uspeo, sa dodatkom izveštaja o eventualnoj grešci. Kada CPE otpočne sa preuzimanjem, pokreće se vremenska kontrola koja traje određeno definisano vreme. Na isteku vremenske kontrole, ukoliko CPE nije uspeo da preuzme i primeni paket ACS-u se šalje odgovor preko metode *DownloadResponse* sa vrednošću 1 u suprotnom 0 ako je sve prošlo bez grešaka. Vrednost 1 označava da se komunikaciona sesija nastavi, dok će se preuzimanje i primenjivanje paketa nastaviti u pozadinskoj niti. Moguće je da se ACS-u vrati SOAP izveštaj o grešci ukoliko je preuzimanje paketa nemoguće. Ukoliko ceo proces prođe kako treba ACS-u se šalje izveštaj o tome preko metode *TransferComplete*. Isto tako, ukoliko dođe do greške prilikom ažuriranja programske podrške, u metodi *TransferComplet* se šalje izveštaj o greškama, kako bi administrator mogao da pravovremeno odreaguje na problem koji je izazvao grešku.

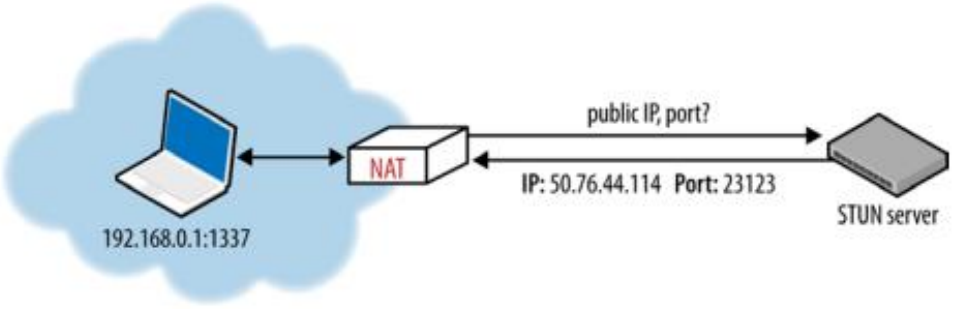
Opisani postupak primenljiv je na bilo koji krajnji uređaj, razlika je samo u programsim podrškama koje pokreću ti krajni uređaji i funkcionalnosti tih uređaja. Poslednje korake

postupka, samo ažuriranje programske podrške karakteristično je za zadati krajni uređaj, u ovom slučaju uređaj zasnovan na Android Nougat platformi.

U zavisnosti od zahteva pristiglog sa strane poslužioca razlikujemo dva načina ažuriranja. Prvi podrazumeva tiho ažuriranje (eng. Silent Upgrade), nakon pristiglog zahteva i dobavljanja datoteke za ažuriranje programske podrške, samo ažuriranje se obavlja tek nakon što korsinik naredni put ponovno pokrene uređaj (eng. Restart). Drugi slučaj podrazumeva nasilno ažuriranje (eng. Force Upgrade), nakon dobavljene datoteke sam uređaj se ugasi, ponovi pokrene i primeni ažuriranje programske podrške. Dodatna mogućnost, koja se pruža je da se ovim putem umesto ažuriranja programske podrške, ažurira odnosno instalira određena aplikacija (APK - Android application package datoteka). Postupak je idnetičan kao kod ažuriranja programske podrške, s tim da se razlikuje krajnji korak, samo instaliranje se obavlja na Android Nougat specifičan način za instaliranje aplikacija.

3.4 STUN klijent

Uloga STUN poslužioca i klijenta ogleda se u otkrivanju adrese i porta preko kojih je moguće vršiti interakciju sa uređajem. Pored otkrivanja adrese, STUN mehanizam ima ulogu da održava vezu na toj adresi i portu i da ne dozvoli da se ta povezanost (eng. Binding) raskine.



Slika 3.5 - STUN klijent-poslužilac komunikacija

Za STUN klijenta iskorišćeno je i prilagođeno postojeće JSTUN rešenje [13]. Rešenje je imlementirano kao modul u okviru TR-069 servisa na strani uređaja.

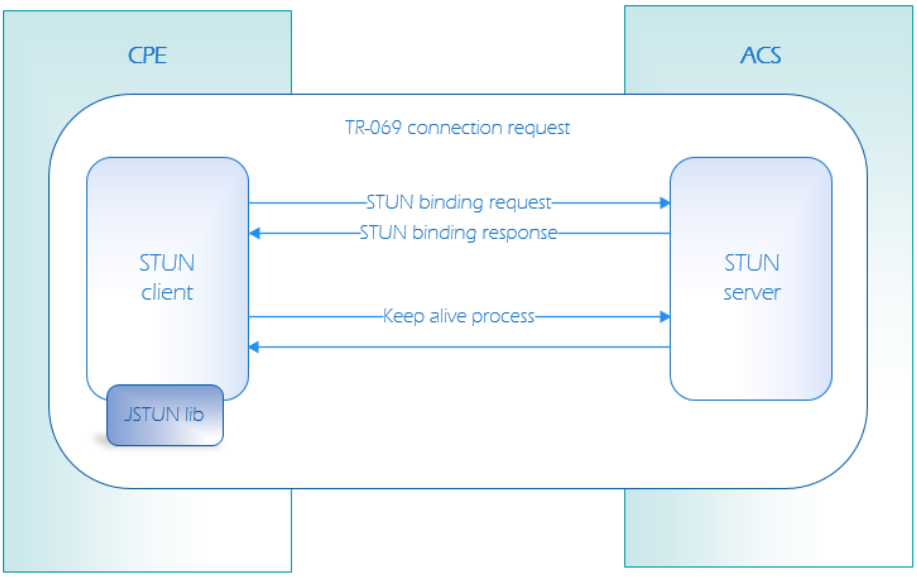
STUN modul predstavlja klijenta čija je uloga da otkrije da li je translacija adrese i porta na snazi. Nakon otkrivanja translacije klijent ima ulogu da održi vezu na usmerivaču protokola na istoj adresi i portu. STUN klijent modul se može podeliti na dve celine:

- Upravljački deo
- Deo zadužen za generisanje zahteva i obradu odgovora

Upravljački deo je zadužen za pokretanje i zaustavljanje klijenta kao i za upravljanje klijentom tokom rada. Deo zadužen za rukovanje STUN porukama je zadužen za generisanje STUN zahteva i obradu STUN odgovora.

Prva faza u radu STUN klijenta je otkrivanje postojanja translacije adrese i porta. STUN klijent šalje zahtev poslužiocu i po sadržaju odgovora od poslužioca zaključuje da li se uređaj nalazi iza usmerivača protokola. Drugu fazu čini održavanje veze.

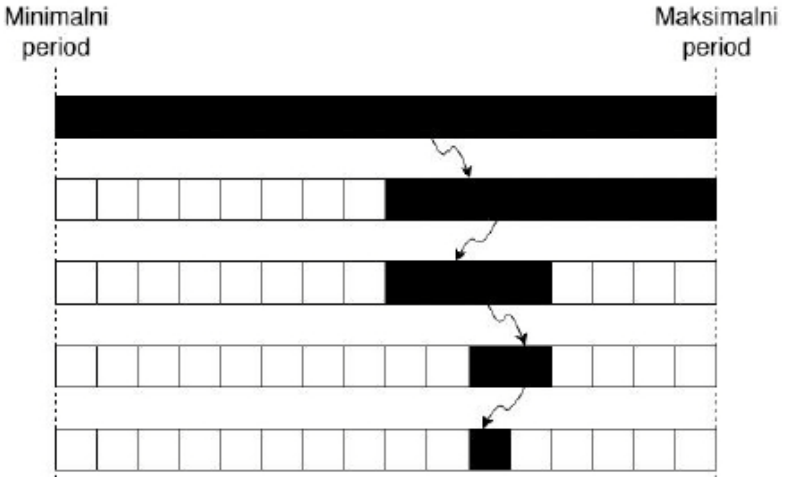
STUN klijent sa primarnog porta periodično šalje poruke koje za cilj imaju da osveže vezu koju je nastala otkrivanjem translacije adrese i porta. Učestalost slanja tih zahteva je definisana parametrima minimalnog i maksimalnog vremena perioda održavanja veze, koji su definisani u TR-181 modelu podataka. Ukoliko su vrednosti ovih parametara različiti STUN klijent mora pristupiti eksperimentalnom otkrivanju tačne vrednosti perioda održavanja veze.



Slika 3.6 – Održavanje veze STUN klijenta i server

Važan deo STUN mehanizma je period održavanja veze (eng. Keep alive period), u protokolu definisan pomoću dva parametra koji definišu maksimalan i minimalan period održavanja veze. Ukoliko ta dva parametra nemaju istu vrednost, mora se pristupiti eksperimentalnom određivanju tačne vrednosti perioda održavanja veze. Metoda binarne pretrage se koristi za određivanje tačne vrednosti. Proces binarne pretrage se zasniva na metodi polovljenja intervala čije su granice parametri minimalnog i maksimalnog perioda održavanja veze, jer slanje zahteva koji služi za osvežavanje veze ne sme biti ređe od vrednosti koju nosi parametar za maksimalni period održanja veze i ne sme biti češći od parametra minimalnog perioda održavanja veze. Proces binarne pretrage je završen kada se pronađe vrednost za koju veza ostaje održana i koja se nalazi na granici tako da svaka vrednost koja je veća od utvrđene

rezultuje prekidom veze između klijenta i poslužioca. Na slici 3.7 je prikazan postupak određivanja tačne vrednosti nad intervalom pomoću binarne pretrage.



Slika 3.7 - Metoda binarne pretrage

Sama metoda binarne pretrage je vremenski zahtevna te iziskuje određenu količinu vremena i sprovodi se pri pokretanju uređaja. Osim sporosti metode za određivanje tačne vrednosti vremenskog intervala između zahteva za vezivanje koji će omogućiti da veza ostane netaknuta postoji još nekoliko potencijalnih problema koji ovakav vid određivanja perioda za održavanje veze čini manje pouzdanim. Ukoliko se usmerivač protokola iz nekog razloga isključi i ponovo pokrene ili je predviđen za ponovno pokretanje (eng. Restart), tada će izračunati period održavanja veze biti manji nego što je to stvarno slučaj. Iz tog razloga je preporučljivo mehanizam pronalaženja tačne vrednosti ponoviti nekoliko puta, ali je to sa stanovišta vremena teško prihvatljivo.

3.5 TR-069 prikupljanje i podizanje ispisa na ACS poslužilac

ACS može koristiti Upload RPC metodu da izazove da CPE otpremi određenu datoteku na određenu lokaciju. Ako datoteka ne može biti uspešno otpremljena, CPE ne traba da ponovo pokuša otpremanje datoteke, već treba da prijavi neuspeo pokušaj ACS-u preko *Upload response* (ako već nije poslat) ili *TransferComplete* poruku. Nakon što je ACS obavešten o neuspehu dopremanja datoteke, ACS naknadno može pokušati ponovo pokrenuti otpremu izdavanjem novog zahteva za *Upload*. Ako CPE primi jedan ili više zahteva za otpremanje pre izvršenja prethodno traženog otpremanja, CPE mora da izvrši svaki od njih u što je moguće bliže zahtevanom vremenu (na osnovu vrednosti argumenta *DelaySeconds* i vremena zahteva). Potrebno je postavljanje zahteva u redove čekanja i CPE mora biti u stanju da radi sa najmanje

tri zahteva. Za svaki izvršen prenos, CPE mora poslati posebnu *TransferComplete* poruku. Otpremanje ne mora biti u istom redu u kome su primljeni zahtevi jer *DelaySeconds* parameter može biti različit u zahtevima. Na primer, ACS bi mogao da zahteva *Upload* sa *DelaySeconds* parametrom jednakim jedan sat, a zatim pet minuta kasnije traži drugi *Upload* sa *DelaySeconds* parametrom jednakim jedan minut. U ovom slučaju, CPE bi izvršio drugi *Upload* pre prvog.

Implemanirano je prikupljanje i otpremanje datoteka sa ispisima CPE uređaja na ACS. Vrši se otpremanje pet datoteka sa izvučenim specifičnim delovima ispisa (eng. Log):

- Drop Box log
- Main Buffer log
- Event Buffer log
- System Buffer log
- Recovey log

4. Programsko rešenje

U ovom poglavlju će detaljno biti opisano programsko rešenje modula za nadzor i konfiguraciju uređaja zasnovanog na TR-069 protokolu. Biće prikazane osnovne izmene početnog rešenja razvijenog na RT-RK institutu. Za razvoj predloženih programskih rešenja korišćena su dva programska jezika:

• C programski jezik: Ovaj programski jezik je korišćen za razvoj Android nativnog dela.

• Java programski jezik: Ovaj programski jezik je korišćen za razvoj Android aplikativnih komponenti.

Glavne izmene obuhvataju implemetaciju specifičnih parametara, integraciju STUN klijentske biblioteke, otpremanje datoteke sa ispisima CPE uređaja na ACS poslužilac, ažuriranje programske podrške. Biće prikazan skup funkcija koje su potrebne za realizaciju funkcionalnosti kao i opis njihovoih parametara. Zbog sličnosti implementacije parametara neće se ponavljati opis istih funkcija za različite parametre.

4.1 Implementacija parametara modela podataka

Implementirani su TR-135 model podataka sa specifičnim STB parametrima i TR-181 modeli podataka sa specifičnim mrežnim parametrima. Prema opisanoj arhitekturi TR-069 protokla biće prikazana sprega poziva od funkcija najnižih slojeva, preko JNI sprege do Java poziva i Android komunikacione biblioteke.

Funkcije za postavljanje parametara su identične, razlikuju se samo po tipu podataka, pravu pisanja/ćitanja (eng. Read/Write) dataog parametra i na osnovu broja višestrukih čvorova u stablu modela podataka. Postavljanje vrednosti parametra će biti prikazano na primeru jednog *Read* parametra i jednog *Read/Write* parametra.

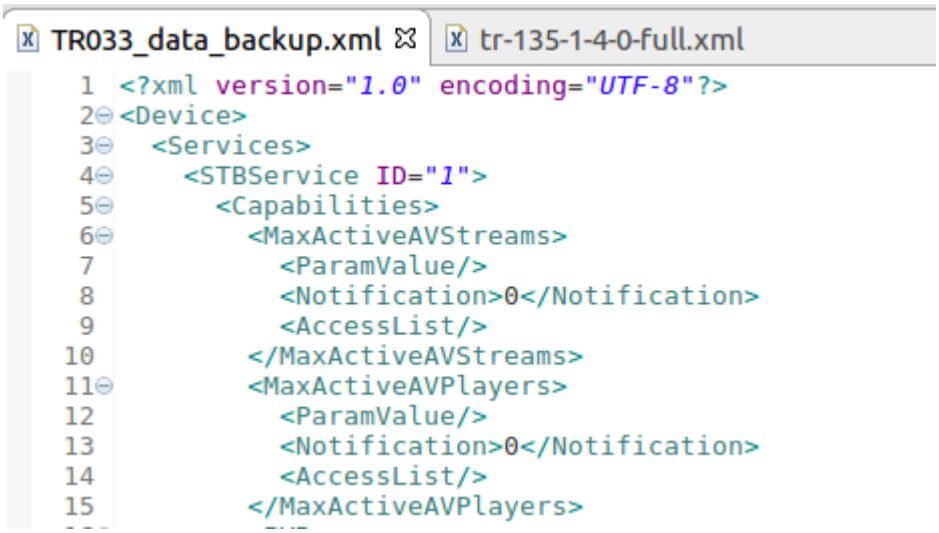
Za Read parameter biće prikazan *MaxActiveAVStreams* iz TR-135 modela podataka.

TR033_data_backup.xml

Ova datoteka predstavlja repozitorijum u kome se na osnovu modela podataka formira stablo parametara. Identična datoteka se kreira na CPE uređaju i u nju se upisuju vrednosti parametara. Na slici 4.1 je prikazan deo repozitorijuma za čuvanje podataka formiranog na osnovu modela podataka. Osim što definiše koji sve parametri postoje za određeni model uređaja, model podataka takođe definiše sve informacije o tom parametru, kao što su:

- Da li je moguće menjati vrednost podatka,
- Kog tipa je podatak,
- Opis datog parametra,
- Kom objektu pripada dati parametar.

Na osnovu ovih informacija formira se repozitorijum, u obliku *XML* datoteke, za čuvanje podataka o parametrima iz odgovarajućeg modela podataka, konkreno u ovom slučaju iz TR-135 modela podataka.



Slika 4.1 – Repozitorijum za čuvanje podataka o parametrima

Za TR-181 model podataka dobavljanje vrednosti parametara vršeno je dirketno iz sistema, ili konfigurisanjem sa strane poslužioca. Za TR-135 model implementirana je sprega za komunikaciju sa drugim Android aplikacijama kako bi postavljanje parametara moglo da se obavi iz aplikacije koja je zadužena za reprodukciju DTV sadržaja. Na slici 4.2 prikazan je poziv funkcija za postavljanje vrednosti (eng. Set functions) od deklaracije u komunikacionoj biblioteci do najnizeg sloja TR-069 sitemske biblioteke. Dat je prikaz kom sloju pripada funkcija i u kojoj datoteci odnosno klasi se nalazi.

Slika 4.2 – Dijagram postavljanja vrednosti parametra

Komponenta *libCPE* je pozicionirana na najnižem nivou Android steka, a to je nivo sistemskih biblioteka. U ovom sloju su implemetirane glavne funkcinalnosti TR-069 klijenta, komunikacija, parsiranje i obrada zahteva, implementacija RPC metoda, upravljanje repoziturijumima i čuvanje podataka.

libcpe_tr033_repository.c

tr033_error_t tr033_set_capabilities_max_active_avstreams(unsigned stb_service_index, int max_active_avstreams)

- Opis: Funkcija za mapiranje parametra u stablu repozitorijuma i postavljanje vrednosti parametra pozivom funkcije ***data_repo_set_param_value***
- Parametri:
  - stb_service_index - Indeks višestrukok čvora
  - max_active_avstreams – Vrednost parametra

libcpe_data_repository_engine.c

repo_err_t data_repo_set_param_value(unsigned setter_id, const char *URL, const char *value, repo_param_type_t data_type, char *error_desc)

- Opis: Postavljanje vrednosti parametra u repozitorijumu
- Parametri:
  - setter_id - Identifikator entiteta koji postavlja vrednost parametra
  - URL - Putanja do parametra u stablu modela podataka
  - Value - Nova vrednost parametra
  - Data_type - Tip podatka za novu vrednost parametra
  - Error_desc - Detaljni opis greške ukoliko dođe do nje tokom postavljanja vrednosti parametra

JNI modul je implementiran u programskom jeziku C. On sadrži deklaraciju i definiciju nativnih funkcija koje se pozivaju iz TR-069 agentskog servisa i koje pozivaju funkcije iz *libCPE* biblioteke. U ovom modulu su implementirane nativne metode koje se povezuju sa pozivima metoda *libCPE* biblioteke, a one se pozivaju iz Java koda. Metode implementirane u JNI modulu su:

- Inicijalizacija klijenta,
- Deinicijalizacija klijenta,
- Pokretanje klijenta,
- Zaustavljanje klijenta,
- Promena vrednosti parametara,
- Dobavljanje vrednosti parametara,
- Dodavanje višestrukog objekta,
- Brisanje višestrukog objekta,
- Registrovanje povratnog poziva i
- Odjavljivanje povratnog poziva.

Ove metode su standardne, ne zavise od modela podataka (od parametara čije vrednosti menjaju ili dobavljaju i od višestrukih objekata koji treba da se dodaju ili brišu). Sve nativne funkcije imaju isti početak imena, koje se dobija na sledeci način:
Java_(ime paketa)_(ime klase)_(ime funkcije)(JNIEnv *env, jobject obj, ...);

- ime paketa – paket u kom se nalazi klasa,
- ime klase – klasa u kojoj se nalaze funkcije,
- ime funkcije – naziv funkcije u Java delu,
- env –referenca na Java okruženje za izvršavanje nativnih funkcija i
- obj –Java objekat čija je nativna funkcija članica klase

libcpe_jni_tr033_wrapper.c

int Java_insight_tr069_client_LibCPENative_tr135SetCapabilitiesMaxActiveAvstreams(JNIEnv *env __attribute__((unused)), jobject obj __attribute__((unused)), jint stb_service_index, jint value)

• Opis: JNI funkcija za postavljanje vrednosti parametra u kojoj se sa identičnim parametrima poziva nativna funkcija - ***tr033_set_capabilities_max_active_avstreams***.

TR-069 servis je realizovan kao Android servis koji se oslanja na TR-069 sistemsku biblioteku. Životni vek ovog servisa počinje pri uključivanju Androida kako bi bio dostupan sistemu za sve vreme njegovog rada. Realizovan je u Java programskom jeziku.

LibCPENative.java

public native int tr135SetCapabilitiesMaxActiveAvstreams(int stbServiceIndex, int maxActiveAvstreams)

• Opis: Deklaracija JNI funkcije za postavljanje vrednosti parametra.

ITR069ClientServiceAPI.aidl

TR135Status setCapabilitiesMaxActiveAvstreams(int stbServiceIndex, int maxActiveAvstreams)

• Opis: Deklaracija funkcije u *AIDL* datoteci koja omogućuje komunikaciju između Android aplikacije i TR-069 Android servisa.

TR069ClientServiceAPI.java

public TR135Status setCapabilitiesMaxActiveAvstreams(int stbServiceIndex, int maxActiveAvstreams)

• Opis: Implementacija funkcije iz *AIDL* datoteke u kojoj se vrši poziv funkcije ***tr135SetCapabilitiesMaxActiveAvstreams*** iz ***LibCPENative*** klase.

Za Read/Write parametre biće prikazan *PreferredSubtitlingLanguage* parametar iz TR-135 modela podataka. Za razliku od *Read* parametra vrši se implementacija 4 funkcije na najnižem nivou, za postavljanje i dobijanje vrednosti, kao i funkcije za registraciju i poništavanje registracije povratne funkcije o promeni vrednosti parametra na strani poslužioca. Na slici 4.3 prikazan je tok poziva prilikom regisracije i poziva povratinh funkcija po slojevima.

Slika 4.3 – Dijagram poziva povratinih funkcija o promeni vrednosti parametra

libcpe_tr033_repository.c

tr033_error_t tr033_set_preferred_subtitling_language(unsigned stb_service_index, char * preferred_subtitling_language)

- Opis: Funkcija za mapiranje parametra u stablu repozitorijuma i postavljanje vrednosti parametra pozivom funkcije ***data_repo_set_param_value.***
- Parametri:
  - stb_service_index - Indeks višestrukok čvora
  - preferred_subtitling_language – Vrednost parametra

tr033_error_t tr033_get_preferred_subtitling_language(unsigned stb_service_index,char **value)

- Opis: Funkcija za mapiranje parametra u stablu repozitorijuma i dobavljanje vrednosti parametra pozivom funkcije ***data_repo_get_param_value.***
- Parametri:
  - stb_service_index - Indeks višestrukok čvora
  - value – Povratna vrednost parametra

libcpe_data_repository_engine.c

repo_err_t data_repo_get_param_value(unsigned getter_id, const char *URL,char **value, repo_param_type_t *data_type)

- Opis: Dobavljanje vrednosti parametra.
- Parametri:
  - getter_id - Identifikator entiteta koji dobavlja vrednost parametra
  - URL - Putanja do parametra u stablu
  - Value - Dobavljena vrednost parametra
  - Data_type - Tip podatka dobavljene vrednosti parametra

Zbog sličnosti *Set/Get* funkcije i sličnog puta poziva neće se opet ponavljati navedni tok kao kod *Read* parametara.

libcpe_tr033_repository.c

tr033_error_t tr033_reg_preferred_subtitling_language(unsigned stb_service_index, tr033_parameter_setter set_func)

- Opis: Registracija povratne funkcije o promeni vrednosti parametra na strani poslužioca.
- Parametri:
  - stb_service_index - Indeks višestrukok čvora
  - set_func - Putanja do parametra u stablu

tr033_error_t tr033_unreg_preferred_subtitling_language(unsigned stb_service_index)

- Opis: Deregistracija povratne funkcije o promeni vrednosti parametra na strani poslužioca
- Parametri:
  - stb_service_index - Indeks višestrukok čvora

deval_specific.c

tr033_error_t tr135_params_register_callbacks()

tr033_error_t tr135_params_unregister_callbacks()

- Opis: Funkcije DEVAL sloja u kojima se vrši registracija i deregistacija povratinh funkcija o promeni vrednosti parametra pozivom opisanih funkcija iz **libcpe_tr033_repository.c**.

tr033_error_t set_preferred_subtitling_language(const char *value)

• Opis: Funkcije DEVAL sloja koja se prosleđuje kao parameter opisanoj funkciji za registraciju iz *libcpe_tr033_repository.c* datoteke i koja služi kao odgovor na povratnu funkciju o promeni vrednosti parametra. Poziva se Java funkcija pomoću ***call_java_func_string_int*** funkcije iz *deval_init.c* datoteke. U zavisnosti od broja parametara mogu se pozvati sledeće funkcije:

- deval_err_t call_java_func_int(const char *func_name);
- deval_err_t call_java_func_string_int(const char *func_name, const char *value);
- deval_err_t call_java_func_string_int_int(const char *func_name, const char *value, const unsigned value_int);
- deval_err_t call_java_func_string_int_int_int(const char *func_name, const char *value, const unsigned value0_int, const unsigned value1_int);

Prvi parametra predstavlja ime Java funkcije koja je pozivana a ostali predstavljaju argumente pozvane funkcije.

LibCPECallbacks.java

public int setPreferredSubtitlingLanguage(String value)

• Opis: Funkcija koja služi kao odgovor na povratnu funkciju o promeni vrednosti parametra. Kao parameter prima novu vrednost koja treba da se postavi u sistemu.

ITR069ServiceCallback.aidl

int setPreferredSubtitlingLanguage(int stbServiceIndex, String value);

• Opis: Deklaracija funkcije u *AIDL* datoteci koja omogućuje komunikaciju između Android aplikacije i TR-069 agentskog servisa. Povratna funkcija treba da obavesti Android aplikaciju o promeni vrednosti parametra na strani poslužioca.

TR069ClientServiceAPI.java

public int setRWPreferredSubtitlingLanguage(int stbServiceIndex, String value)

• Opis: Funkcija u koja služi kao okidač povrtane funkcije iz *AIDL* datoteke.

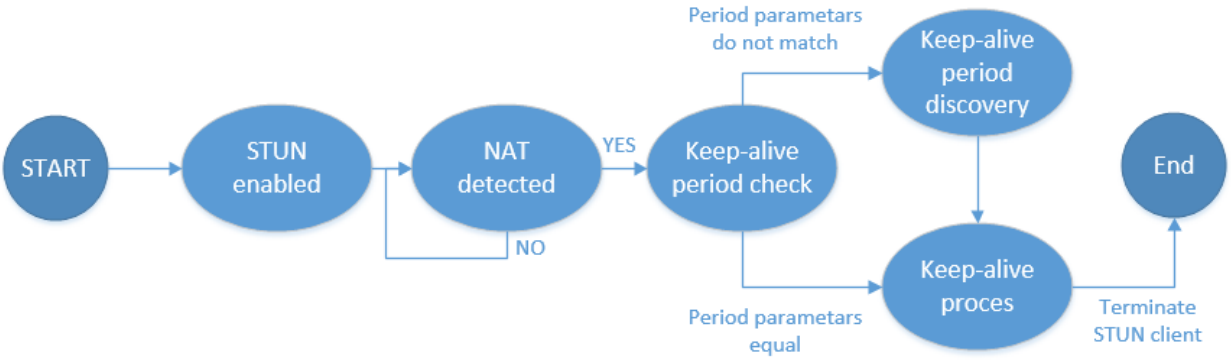
• Parametri:

- stbServiceIndex - Indeks višestrukok čvora
- value – Nova vrednost

4.2 STUN klijent

STUN klijent predstavlja modul koji u sprezi sa STUN poslužiocem vrši otkrivanje translacije adrese i porta, kao i održavanje veze na usmerivaču protokola. Tok rada STUN klijenta je definisan pomoću mašine stanja koja je prikazana na slici 4.6 i koja sadrži sledeća stanja:

- Inicijalno(početno) stanje
- Provera postojanja STUN poslužioca
- Analiza postavke mreže
- Analiza parametara perioda održavanja veze
- Eksperimentalno određivanje parametra održavanja veze
- Odražavanje veze
- Krajnje stanje



Slika 4.4 – Automat stanja STUN klijenta

ManagementServer.java

U ovoj klasi implementirane su povratne funkcije o promeni vrednosti nekog od STUN parametara koji pripadaju TR-181 modelu podataka. Konfiguracije se vrši od strane administrator na ACS poslužiocu. Parametri funkcija predstavljaju novo-postavljne vrednosti.

- ***public void onSetSTUNEnable(boolean stunenable)***
- ***public void onSetSTUNServerAddress(String stunserverAddress)***
- ***public void onSetSTUNServerPort(long stunserverPort)***
- ***public void onSetSTUNUsername(String stunusername)***
- ***public void onSetSTUNPassword(String stunpassword)***
- ***public void onSetSTUNMaximumKeepAlivePeriod(int seconds)***
- ***public void onSetSTUNMinimumKeepAlivePeriod(long seconds)***

TR181Monitor.java

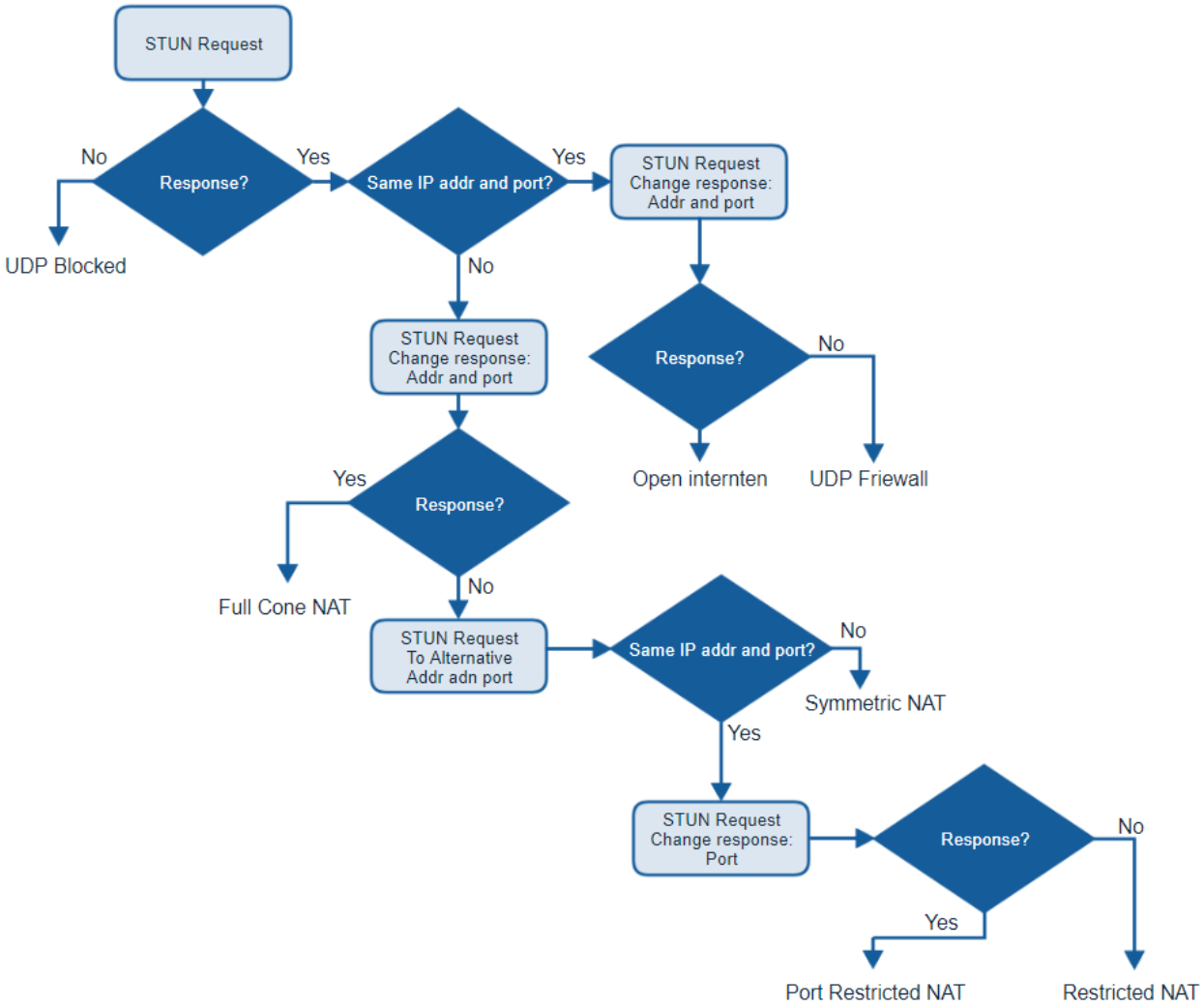
public void restartSTUNClient()

• Opis: Funkcija za ponovno pokretanje STUN klijenta. Poziva se na svaku promenu nekog od STUN parametara.

STUNClient.java

public class STUNClient

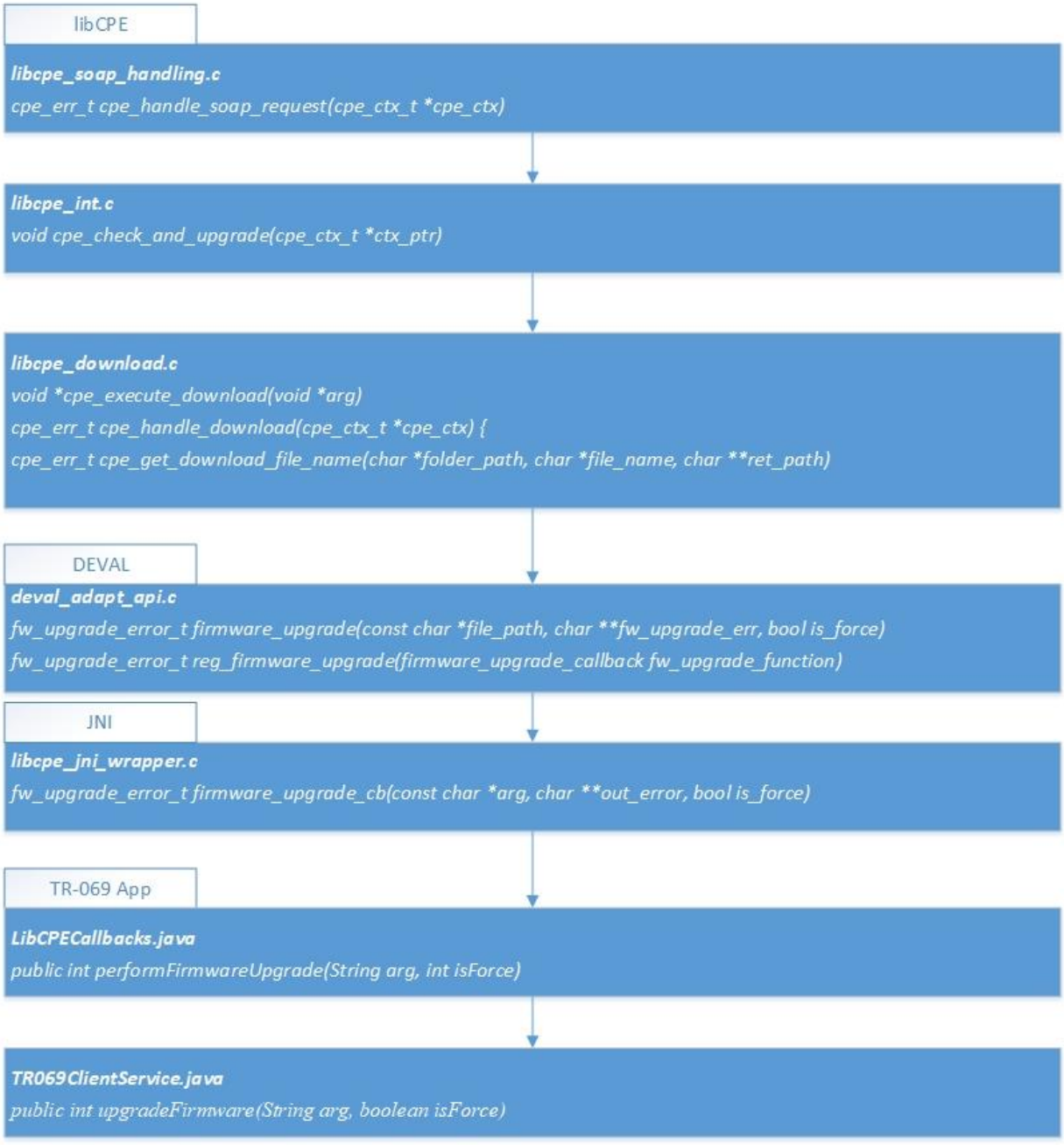
U ovoj klasi implementirane su glavne funkcinalsnot STUN klijenta. Na slici 4.5 prikazano je imlementirano upravljanje STUN klijetnom. Funkcinalnost je ostvarena oslanajem na pozive iz uvčene JSTUN bibilioteke.



Slika 4.5 - Dijagram logike STUN klijenta

4.3 Ažuriranje programske podrške

Ažuriranje programske podrške predstavlja jednu od glavnih funkcionosti TR-069 protokola. Na slici 4.6 prikazan je poziv svih potrebnih funkcija za obavljanje ažuriranja od najnizih slojeva ka najvišim.



Slika 4.6 – Dijagram poziva funkcija za ažuriranje programske podrške

TR069ClientService.java

public int upgradeFirmware(String arg, boolean isForce)

• Opis: Funkcija za ažuriranje programske podrške uređaja.

• Parametri:

- arg – Putanja na kojoj se nalazi skinuta OTA image datoteka
- isForce – Indikator da li se radi o nasilnom ažuriranju ili odloženom

U zavisnosti da li se radi o naslilnom ili odloženom ažuriranju koriste se sedeće metode:

•***RecoverySystem.installPackage(getApplicationContext(), file)***

Funkacija koja se koristi prilikom nasilnog ažuriranja, sa parametrima jedinstvenog Contexta aplikacije i dadoteke za ažuriranje, odnosno OTA image.

• ***RecoverySystem.processPackage(getApplicationContext(), file, null)***

• ***RecoverySystem.scheduleUpdateOnBoot(getApplicationContext(), file)***

Date metode kosiste se kod odloženog ažuriranja, odnosno kada se ažuriranje obavlja na sledećem gašenju uređaja.

Dodatna mogučnost predstavlja instaliranje aplikacija, odnosno *APK* datoteka sa strane poslžioca na CPE uređaje.

public static boolean installApk(String arg)

• Opis: Funkcija za instaliranje *APK* datoteka, kao parameter prima putanju do datoeke. Instaliranje se vrši pomoću androidovih klasa *PackageManager* i *PackageInstaller*.

LibCPECallbacks.java

public int performFirmwareUpgrade(String arg, int isForce)

• Opis: Povratne funkcije za obaveštavanje o pokretanju ažuriranja programske podrške, nakon uspešnog dobavljanja sa poslužioca.

public int performApplicationInstall(String arg)

• Opis: Povratne funkcije za obaveštavanje o pokretanju instaliranja *APK* datoeke, nakon uspešnog dobavljanja sa poslužioca.

libcpe_jni_wrapper.c

fw_upgrade_error_t firmware_upgrade_cb(const char *arg, char **out_error, bool is_force)

• Opis: JNI povratne funkcije za obaveštavanje o ažuriranja programske podrške, nakon uspešnog dobavljanja sa poslužioca.

apk_install_error_t apk_install_cb(const char *arg, char **out_error)

• Opis: JNI povratne funkcije za obaveštavanje o pokretanju instaliranja *APK* datoeke, nakon uspešnog dobavljanja sa poslužioca.

deval_adapt_api.c

fw_upgrade_error_t firmware_upgrade(const char *file_path, char **fw_upgrade_err, bool is_force)

• Opis: Funkcija za upravljanje ažuriranja programske podrške.

fw_upgrade_error_t reg_firmware_upgrade(firmware_upgrade_callback fw_upgrade_function)

• Opis: Funkcija za registrovanje povrtane funkcije za ažuriranje programske podrške.

• Parametri:

- fw_upgrade_function _callback - funkcija koja se poziva posle okidanja.

apk_install_error_t perform_apk_install(const char *file_path, char **apk_install_err)

• Opis: Funkcija za upravljanje instlarianja datoteka.

apk_install_error_t reg_apk_install(apk_install_callback apk_install_function)

• Opis: Funkcija za registrovanje povrtane funkcije za instaliranje datoteke.

libcpe_int.c

void cpe_check_and_upgrade(cpe_ctx_t *ctx_ptr)

• Opis: Funkcije za proveru validnosti ažuriranja programske podrške. U funkciji se proverava da li zahtev sadrži datoteku za ažuriranje ili instaliranje i na osnovu toga poziva se odgovarajuća funkcija.

• Parametri:

- ctx_ptr – Kontekst sa parsiranim podacima zahteva

libcpe_download.c

void *cpe_execute_download(void *arg)

cpe_err_t cpe_handle_download(cpe_ctx_t *cpe_ctx)

cpe_err_t cpe_get_download_file_name(char *folder_path, char *file_name, char **ret_path)

• Opis: Funkcije za rukovanje dobavljanja željene datoteke sa ACS poslužioca. Dobavljanje se obavlja upotrebo funkcija iz *LibCurl* biblioteke.

libcpe_soap_handling.c

cpe_err_t cpe_handle_soap_request(cpe_ctx_t *cpe_ctx)

• Opis: Funkcija za rukovanje i parsiranje zahteva primljenih od ACS poslužioca.

4.4 Otpremanje ispisa

Na osnovu zahteva poslatog sa ACS poslužioca otprema se željena datoteka sa ispisima CPE uređaja. Na slici 4.7 prikazan je skup funkcija potrebnih za otpremanje datoteka.

Slika 4.7 – Dijagram poziva funkcija za otpremanje datoteka sa ispisom

VendorLogFilesEnumerator.java

public static void initVendorLogFiles()

• Opis: Funkcija za inicijalizaciju odnosno pravljenje datoteka za skladištenje ispisa.

libcpe_soap_handling.c

cpe_err_t cpe_handle_soap_request(cpe_ctx_t *cpe_ctx)

• Opis: Funkcija za rukovanje i parsiranje zahteva primljenih od ACS poslužioca.

libcpe_upload.c

cpe_err_t cpe_handle_upload(cpe_ctx_t *cpe_ctx)

• Opis: Funkcija za rukovanje otpremnja datoteke na ACS poslužilac.

void *cpe_execute_upload(void *arg)

• Opis: Funkcije za otpremanje željene datoteke na ACS poslužioca. Otpremanje se obavlja upotrebo funkcija iz *LibCurl* biblioteke.

cpe_err_t cpe_get_upload_file_path(cpe_ctx_t *ctx_ptr, const char *upload_file_type, char **file_path)

• Opis: Funkcija za upis ispisa CPE uređaja u datoteke.

static cpe_err_t cpe_get_log_file_path(int index, char **out_path

• Opis: Funkcija za dobavljanje putanja datoteka za ispis.

• Parametri:
  - index – Indeks tražene datoeke (1-5)
  - out_path – Putanja do datoteke

static cpe_err_t cpe_create_adb_log_file(const char *name, char *out_path)

• Opis: Funkcija za prikupljanje ispisa.

• Parametri:
  - name – Ime za filtriranje ispisa
  - out_path – Putanja do datoteke

static cpe_err_t cpe_create_dropbox_log_archive(char *out_path)

• Opis: Funkcija za prikupljanje i arhiviranje *Drop Box* ispisa.

5. Ispitivanje i verifikacija

U okviru ovog rada obavljena su ispitivanja realizovanih modula. U svrhu ispitivanja TR-069 programske podrške sproveden je niz ispitnih slučajeva. Ispitivanje je vršeno u više iteracija kako bi rezultati bili što precizniji. Ispitni slučajevi su imali za cilj ispitivanje robusnosti razvijene programske podrške. Ispitivanje je vršeno u više etapa:

- Ručno ispitivnje, proveravanjem vrednosti parametra na uređaju i poslužiocu
- Ručno ispitivanje ažuriranja programske podrške
- Ručno ispitivanje STUN klijenta
- Ručno ispitivanje otpremanja datoetka sa ispisima na poslužilac
- Testna aplikacija za povezivanje na TR-069 servis
- Programsko ispitivanje vrednosti parametara

Testiranje je vršeno na dve nove platform. Jedan od DTV prijemnika koji je korišćen pri izradi ovog rada je Android platforma bazirana na Synaptics BG4-CT čipsetu, Slika 5.1. Komponente koje ova platforma poseduje su:

- Video dekoder UHD HEVC H.265, AVC/H.264, MPEG-1/H.261/H.263
- HDMI 2.0 (HDCP 2.2 & CEC supported) izlaz
- Audio dekoder MPEG-1 lazers 1,2, AAC LC, AAC HE Level 2, AAC HE Level 4, MP3
- RAM memorija – 2GB DDR-L
- Procesor Marvell BG4-CT 4x1,5GHy
- Flash memorija – eMMC 8GB
- Ethernet utičnicu 10/100 Full-duplex
- Wifi i BlueTooth

Slika 5.1 – Platforma bazirana na Synaptics BG4-CT čipsetu

Drugi DTV prijemnika koji je korišćen pri izradi ovog rada je Android platforma bazirana na Amlogic S805X čipsetu. Slika 5.2. Komponente koje ova platforma poseduje su:

- Video dekoder sa podrškom za HEVC/H.265 1080P60,HD MPEG-4,H.264,MPEG-2
- Audio dekoder sa podrškom za MPEG-1, Layer I & II; MPEG-2 Layer II; HE-AAC
- Čip S805X Quad-Core A53 up to 1.2GHz(DVFS),11000DMIPS
- USB 2.0
- Ethernet konekcija (10/100M)
- 2x2 802.11 b/g/n/ac Wi-Fi konekcija
- RAM 1GByte DDR3
- Flash 8GByte EMMC
- HDMI 2.0b with HDCP2.2
- Bluetooth BT 4.1
- Power 12V/1.5A
- Rezolucija - 480i/480p,576i/576p/720p/1080p, Support HDR10 output

Slika 5.2 – Platforma bazirana na Amlogic S805X čipsetu

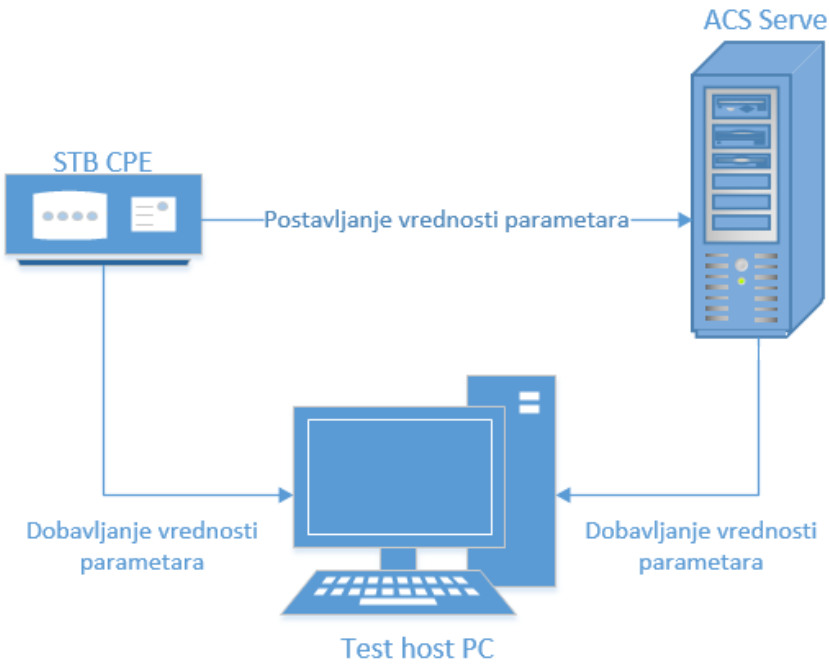
5.1 Ispitivanje vrednosti parametara

Ispitivanje vrednosti parametara je vršeno ručno upoređivanjem vrednosti parametara na uređaju i poslužiocu. Napravljena je Android aplikacija koja se *Bainder* mehanizmom zakači na TR-069 servis radi testiranja postavljanja vrednosti parametara iz druge aplikacije. Takođe testirano je i obaveštavnje druge aplikacije o promeni vrednosti parametara na poslužiocu. U okviru aplikacije postavljane su vrednosti parametara iz TR-135 modula pre nego što je komunikaciona biblioteka implemetirana u okviru apliakcije za prikaz DTV sadržaja.

Na slici 5.3 prikazano je instrumentizovano testiranje vrednosti parametara. Testna skripta je imlementirano u Pyton programskom jeziku. Repozatorijum sa vrednostima parametara se dobavlja sa CPE uređaja, a povezivanjem na ACS dobavljaju se vrednosti parametara na poslužiocu i vrši se njihovo upoređivanje. Testiranjem u više iteracija utvrđeno je poklapanje vrednosti parametara na CPE uređaju i ACS poslužiocu, prema tome sve funkcije za upravljanje parametrima su ispravno implementirane i funkcionalne.

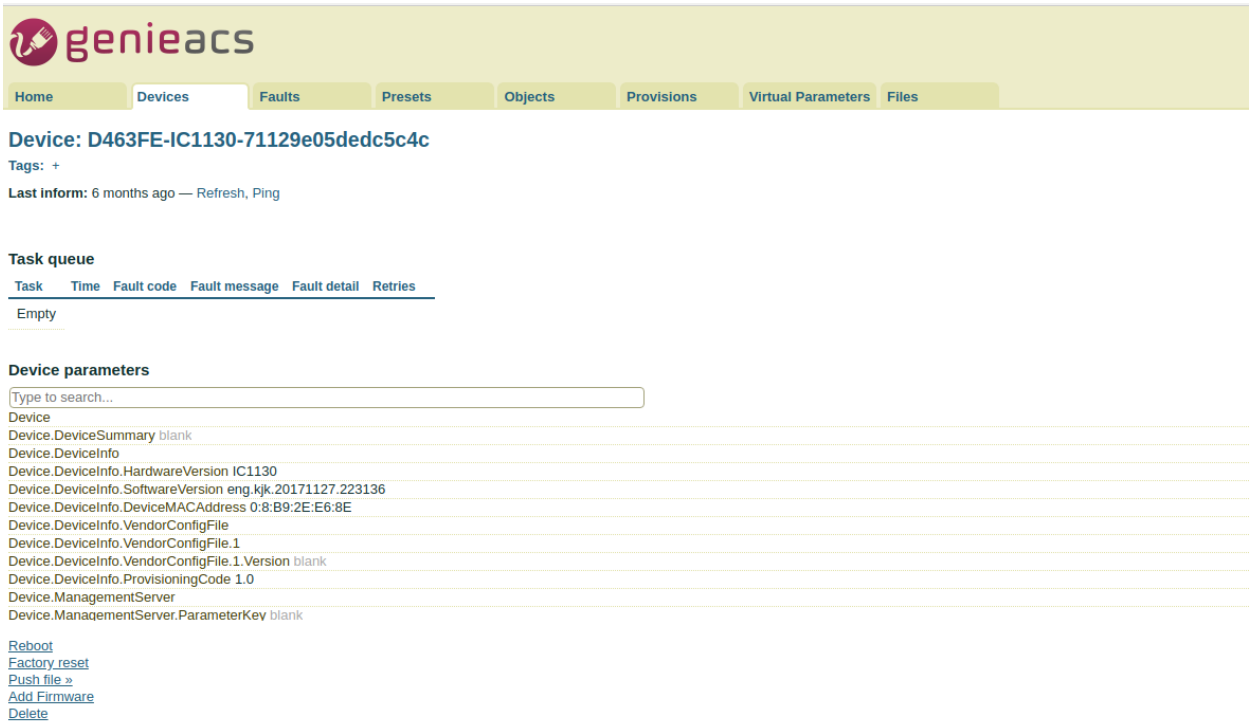
Test	Rezultat
Ručno testiranje parametara iz TR-135 i TR-181 modela poadataka	✓
Instrumentalizovano testiranje parametara iz TR-135 i TR-181 modela poadataka	✓

Tabela 5.1 - Spisak testova ispitivanja parametara



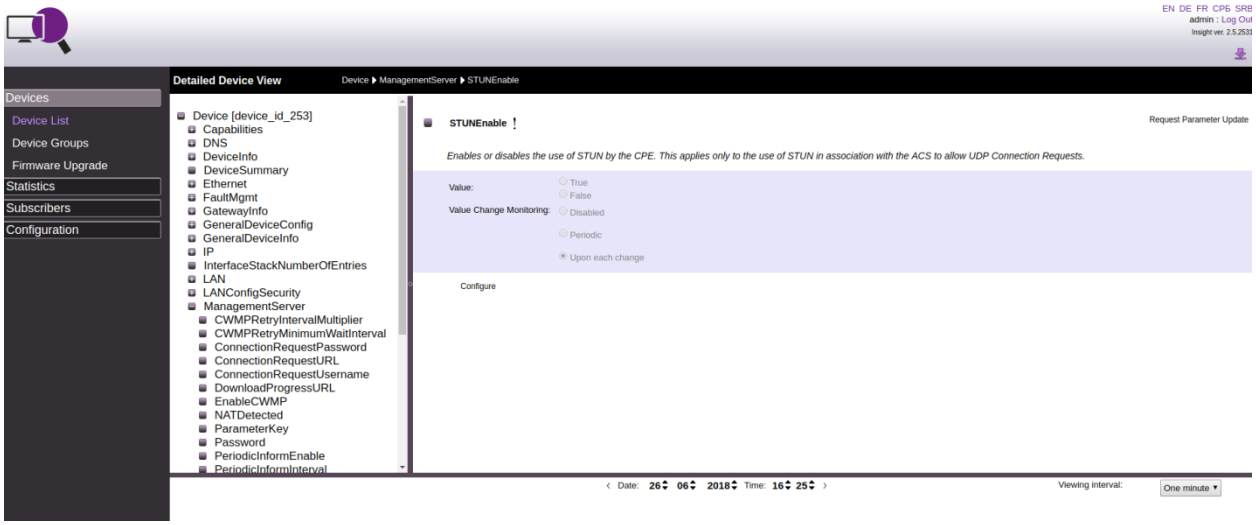
Slika 5.3 – Sistem za testiranje

Prilikom testiranja korišćen je javni GenieACS [14] poslužilac otvorenog koda (eng. Open Sorce). GenieACS je izabran zbog kompatibilnosti sa standardom i ciljanim posližiocem operatera. Okruženje GenieACS poslužioca prikazano je na slici 5.4.



Slika 5.4 – GenieACS poslužilac

Prilikom ručnog tesiranja korišćen je Insihgt ACS poslužilac razvijen na RT-RK institutu i kompatibilan je sa početnim rešenjem CPE klijenta. Okruženje Insight ACS poslužioca prikazano je na slici 5.5.



Slika 5.5 – Insihgt ACS poslužilac

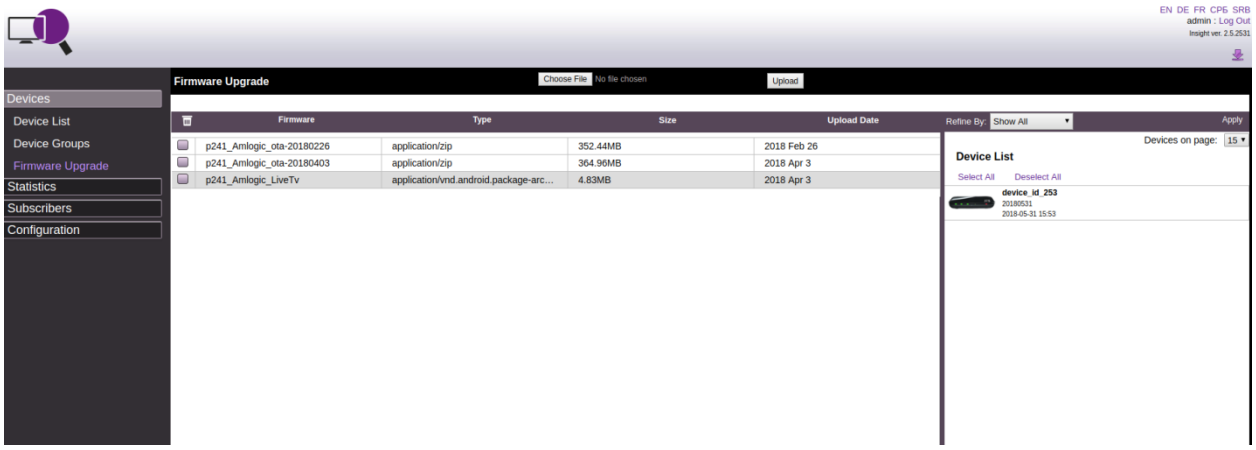
5.2 Ažuriranje programske podrške

Početno rešenje odudarlo je od standarda pa je konstantnim testiranjem usklađeno sa standardom i ACS poslužiocem. Načinjene su promene oko obaveštavanja poslužioca o ispravnom završetku dobavljanja datoteke (OTA imag-a) i o uspešnosti primene odnosno ažuriranja programske podrške. Testiranjem je utvrđena potpuna funkionalnost. Testirani su sledeći slučajevi prikazani u tabeli 5.2.

Test	Rezultat
Odloženo ažuriranje preko *Eterneth* mreže	✓
Odloženo ažuriranje preko *Wi-Fi* mreže	✓
Nasilno ažuriranje preko *Eterneth* mreže	✓
Nasilno ažuriranje preko *Wi-Fi* mreže	✓
Brisanje datoteke nakon ažuriranja	✓
Komunikacija sa ACS serverom	✓
Instaliranje aplikacija preko *Eterneth* mreže	✓
Instaliranje aplikacija preko *Wi-Fi* mreže	✓

Tabela 5.2 – Spisak testova ažuriranje programske podrške

Takođe izvršeno je i testiranje funkcionalnosti instaliranja aplikacija odnosno *APK* datoteka preko ACS poslužioca. Instaliranje se obavlja u pozadini bez potrebne dozvole korsnika CPE uređaja. Testiranjem je utvrđena potpuna funkinalnost. Testiranje je vršeno pomoću Insihgt ACS poslužioca, na slici 5.6 prikazano je okurženje za ažuriranje programske podrške.



Slika 5.6 - Okurženje za ažuriranje Insihgt ACS poslužioca

5.3 Testiranje otpremanja datoteka

Izvršeno je ručno testiranje funkcionalnosti otpremanja datoteka sa ispisom sa CPE uređaja na ACS poslužilac. Na ACS poslužiocu se ručno pokreće slanje zahteva za dobavljanje ispisa. Nakon toga TR-069 klijent na strani CPE uređaja parsira zahtev i obavlja otpremanje zeljene datoteke sa ispisima. Testiranjem je utvrđena potpuna funkinalnost.

5.4 Testiranje STUN klijenta

Implementacija zahteva za povezivanje je testirana kroz niz testova u kojima su pokrivena sva okruženja u kojima će se koristiti zahtev za povezivanje.

Testno okruženje možemo podeliti u tri kategorije:

- Krajnji uređaj i poslužilac su u istoj mreži
- Između krajnjeg uređaja i poslužioca se nalazi usmerivač protokola
- U mreži u kojoj se nalazi krajnji uređaj prisutan je proxy

Izvršeni su i testovi otkrivanja translacije adrese i porta, i održavanja veze tokom perioda vremena od 48 sati. Ustanovljeno je da proxy nema uticaj na funkcionisanje STUN mehanizma. Sa druge strane, ukoliko zaštitni zid mreže blokira slanje UDP poruka, tada STUN mehanizam neće raditi. U tabeli 5.2 su prikazani rezultati testiranja STUN klijenta.

STUN Test	Rezultat
Otkrivanje translacije	✓
Otkrivanje translacije *proxy*	✓
Održavanje veze	✓
Održavanje veze *proxy*	✓
Zaštitni zid blokira UDP	✓

Tabela 5.3 – STUN testovi

6. Zaključak

Osnovni zadatak rada bio je realizacija TR-069 klijenta. Rešenje je realizovano upotrebom programskih jezika C i Java. Rešenje poseduje malu zavisnost od eksternih biblioteka i iz tog razloga je portabilno i prilagodljivo svakoj Android platformi. Rešenje ne narušava koncept već postojećeg rešenja klijenta za krajnji uređaj i ACS poslužioca. Tokom razvoja rešenja dolazilo je do definisanja novih zahteva, koji se najviše tiču modularnosti, lakoće održavanja i proširivosti programske podrške. Tokom realizacije ispunjeni su svi zahtevi.

Dalji razvoj može da se odvija u dva smera. U jednom će se razvijati komunikacioni deo radi podržavanja novih profila TR-069 standarda, a u drugom će se pružati podrška za nove modele podataka podržane od strane TR-069 protokola, kao i integracija biblioteke na druge krajnje uređaje. Poseban aspekt obuhvata razvoj profila i proširivanju modela podataka za određene uređaje, koji trenutno nisu podržani TR-069 protokolom. Izazov bi bio zameniti postojeću XML bazu podataka sa SQL bazom što bi znatno ubrzalo postupak upisa i ispisa podataka o parametrima.

7. Literatura

[1] TR-069 (CPE Wan Management Protocol), Broadband Forum, http://www.broadband-forum.org/cwmp

[2] TR-135 Data Model for a TR-069 Enabled STB, Broadband Forum, http://www.broadband-forum.org/technical/download/TR-135_Amendment-3.pdf

[3] Professional Android Application Development, Autor: Reto Meier ,ISBN-10: 0470344717

[4] Frank Ableson, Charlie Collins, Robi Sen, Unlocking Android, A Developer's Guide

[5] Embedded Android: Porting, Extending, and Customizing, Autor: Karim Yaghmour, ISBN-13: 978-1-449-30829-2

[6] W. Fischer, Digital Video and Audio Broadcasting Technology, A Practical Engineering Guide, Springer; 3rd ed. 2010

[7] Pekowsky, S.; Jaeger, R., "The set-top box as "multi-media terminal"," Consumer Electronics, IEEE Transactions on , vol.44, no.3, pp.833,840, Aug 1998

[8] O'Driscoll G., "The Essential Guide to Digital Set-top Boxes and Interactive TV," Printice Hall PTR, Ireland, 2000.

[9] Datta T., "Digital Covergence: the Set Top Box and Its Digital Interface," White paper,Enthink, July 1998

[10] TR-106 Data Model Template for TR-069-Enabled Devices, Broadband Forum, http://www.broadband-forum.org/technical/download/TR-106_Amendment-6.pdf

[11] Quality Of Video and Audio for Digital Television Services (Quovadis) project, Final report, European Commision Cordis..

[12] Management of service quality in television operations (Mosquito) project, European Commision Cordis, http://cordis.europa

[13] "JSTUN" - Java Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translation (NAT), by Thomas King, https://jstun.javawi.de

[14] GenieACS - Open Source solution for auto-configuration server, https://genieacs.com

[15] Bjelica, M. Z.; Papp, I.; Teslic, N., "Diagnostic CWMP Client for Set-Top Box Devices", ICCE, 2014