



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Јована Михић

ИНТЕГРАЦИЈА ХМРР ПРОТОКОЛА У TR-069 ПОСЛУЖИТЕЉ НА АНДРОИД ПЛАТФОРМИ

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 2019



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:	Јована Михић		
Ментор, МН:	проф. др Илија Башичевић		
Наслов рада, НР:	Интеграција XMPP протокола у TR-069 послужитељ на Андроид платформи		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2019		
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)	7/50/18/1/18/0/0		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника		
Предметна одредница/Кјуине 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 побољшање постојећег решења за интеграцију модела за успостављање комуникационог канала кроз преводитеље мрежних адреса уређаја базираног на TR-069 комуникационом протоколу на Андроид платформи. Надограђен је комуникациони слој између TR-069 сервиса и Андроид апликација. У оквиру TR-069, клијентског сервиса додат је XMPP клијентски модул.		
Датум прихватања теме, ДП:			
Датум одbrane, ДО:			
Чланови комисије, КО:	Председник:	проф. др Небојша Пјевалица	
	Члан:	проф. др Марија Антић	Потпис ментора
	Члан, ментор:	проф. др Илија Башичевић	

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:	Jovana Mihić	
Mentor, MN:	Ilija Bašičević, PhD	
Title, TI:	Integration of XMPP protocol in the TR-069 server on the 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:	2019	
Publisher, PB:	Author's reprint	
Publication place, PP:	Novi Sad, Dositeja Obradovica sq. 6	
Physical description, PD: (chapters/pages/ref./tables/pictures/graphs/appendices)	7/50/18/1/18/0/0	
Scientific field, SF:	Electrical Engineering	
Scientific discipline, SD:	Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, S/KW:	DTV, STB, CPE, ACS	
UC		
Holding data, HD:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, N:		
Abstract, AB:	This document represents improvement of an existing solution for establishing communications channels thorough network addresses translators of devices based on TR-069 communication protocol on android platform model integration. Layer of communication between TR-069 services and android applications has been upgraded. XMPP client module has been added to client service within TR-069.	
Accepted by the Scientific Board on, ASB:		
Defended on, DE:		
Defended Board, DB:	President:	Nebojša Pjevalica, PhD
	Member:	Marija Antić, PhD
	Member, Mentor:	Ilija Bašičević, PhD
		Menthor's sign

Zahvalnost

Posebno se zahvaljujem Milanu Ivankoviću na stručnoj pomoći, izdvojenom vremenu i korisnim savjetima.

Zahvaljujem se mentoru dr Iliji Bašičeviću, tehničkom mentoru Goranu Stuparu i Darku Dejanoviću.

Takođe zahvaljujem se svojoj porodici i prijateljima na podršci tokom školovanja i tokom izrade ovog rada.

Želim da se zahvalim i institutu RT-RK na pruženoj prilici i omogućenim uslovima za izradu diplomskog rada.

SADRŽAJ

1.	Uvod.....	3
2.	Teorijske osnove	5
3.	Koncept rješenja	20
4.	Programsko rješenje.....	30
5.	Ispitivanje i verifikacija	35
6.	Zaključak.....	41
7.	Literatura	43

SPISAK SLIKA

- Slika 2.1 – Arhitektura Android programskog steka
- Slika 2.2 – Primjer okruženja TR-069 protokola
- Slika 2.3 – XMPP arhitektura
- Slika 2.4 – Tipovi strofi poruka
- Slika 2.5 – Razmjena IQ strofa
- Slika 3.1 – XMPP sekvenca poruke zahteva za vezu
- Slika 3.2 – Format poruke za zahtjev za povezivanje
- Slika 3.3 – Poruka pozitivnog odgovora na poruku
- Slika 3.4 – Poruka ako CPE nije podržan
- Slika 3.5 – Poruka ako zahtjev nije autentifikovan
- Slika 4.1 – Repozitorijum za čuvanje podataka o parametrima
- Slika 4.2 – Dijagram postavljanja vrijednosti parametra
- Slika 4.3 – Dijagram posavljavanja vrijednosti parametara
- Slika 5.1 – Sistem za testiranje
- Slika 5.2 – GenieACS poslužitelj
- Slika 5.3 – Primjer pozitivnog odgovora na poruku
- Slika 5.4 – Poruka prilikom greške kod logovanja
- Slika 5.5 – Primjer za neispravan format poslate poruke
- Slika 5.6 – Ploča na kojoj je vršeno testiranje

SPISAK TABELA

Tabela 5.1 Spisak testova ispitivanja parametara

Tabela 5.2 - Spisak testova primljenih poruka

Tabela 5.3 Spisak testova za komunikaciju sa ACS-om

SKRAĆENICE

NAT	- <i>Network Address Translation</i> , Preslikavanje mrežnih adresa.
STUN	- <i>Session Traversal Utilities for NAT</i> , Uslužni program za NAT.
BSD	- <i>Berkeley Software Distribution</i> , Operativni sistem za računar, verzija je UNIX®-a. Razvijen na Kalifornijskom univerzitetu Berkeley, 1970. godine.
JNI	- <i>Java native interface</i> , Sprega Java programskog jezika i C koda.
IDL	- <i>Android Interface Definition Language</i> , Protokol o komunikaciji korisnik poslužitelj.
TR-069	- <i>Tehnical Report 069</i> , Protokol za nadzor i upravljanje 069.
CPE	- <i>Customer-Premises equipment</i> , Korisnički uređaj koji podržava TR-069 protokol.
ACS	- <i>Auto Configuration Server</i> , Poslužitelj koji podržava TR-069 protokol.
IM	- <i>Instant Messaging</i> , Razmjena trenutnih poruka.
API	- <i>Application Programming Interface</i> , Interfejs za programiranje aplikacije.
SASL	- <i>Simple Authentication and Security Layer</i> , Okvir za autentifikaciju i sigurnost podataka u internet protokolima.
TLS	- <i>Transport Layer Security</i> , Kriptografski protokol koji obezbjeđuje kompletну komunikaciju sigurnosti preko mreže.
SSL	- <i>Secure Socked Layer</i> , Standardna tehnologija za održavanje sigurne internet veze i čuvanje svih osjetljivih podataka koji se šalju između dva sistema.
SOAP	- <i>Simple Object Access Protocol</i> , Protokol za poruke koji omogućava

distribuirane elemente aplikacije da komuniciraju.

XML

- *Extensible Markup Language*, Jezik koji definiše skup pravila za kodiranje dokumenata u formatu koji je čitljiv i za čovjeka i mašinski čitljiv.

1. Uvod

U posljednjih nekoliko godina tehnologija je unijela značajne promjene u naš život.

Ubrzan razvoj tehnologije i pojava prvih pametnih uređaja na tržištu doveli su do svakodnevnog povećanja broja uređaja, koje je uslovilo i njihov tehnološki razvoj. Niz savremenih aplikacija uveliko olakšava obavljanje svakodnevnih aktivnosti. Multimedijalni uređaji, kao što su telefoni i televizori, već uveliko su zasnovani na internetu. Uređaji sa ovakvim potrebama zahtjevaju od operatera (eng. *Service provider*) komunikacionih i IPTV (eng. *Internet Protocol Television*) usluga da obezbijedi besprijekoran rad svojih mreža, kako bi se korisnicima omogućila što bolja usluga.

Razvoj mrežnih tehnologija i brzine razmjene podataka je dovela do mogućnosti da se mreže sastavljene od uređaja potrošačke elektronike kontrolišu i nadgledaju sa udaljenih stanica. Poslužitelji (eng. *Servers*) uređaja potrošačke elektronike su sistemi zasnovani na računaru, čiji je osnovni zadatak komunikacija sa uređajima sa jedne strane i manipulacija podataka sa druge strane. Oni upravljaju podacima tako što ih obrađuju, skladište, grupišu i isporučuju drugim aplikacijama ili korisnicima radi dalje obrade ili grafičke prezentacije. Kovmuniciraju preko slojeva (eng. *API - Application Programming Interface*), a protokoli koji se koriste u sprežnim slučajevima definisani su od strane poslužitelja.

U ovom radu je opisan problem, koncept i realizacija XMPP komunikacionog protokola (eng. *Extensible Messaging and Presence Protocol*).

Prelaskom sa analognog na digitalno emitovanje televizijskog i radio signala došlo je do primata digitalne televizije u većini zemalja. Bitna karakteristika digitalne televizije je kvalitet signala (eng. *QoS – Quality of Service*). Digitalni signal, iako bolji od analognog, podložan je raznim tipovima šuma. Kod IPTV-ja zahtjevi za kvalitetom signala su još više izraženi zbog mehanizma prenošenja signala. Kvalitet signala kod IP televizije zavisi od propusnosti mreže

koju korisnik ima, mrežne opreme i smetnji u prenosu [1]. Zbog različitih smetnji dolazi do gubitka na kvalitetu signala, gubitka paketa, mogući su kvarovi na digitalnim prijemnicima i drugo. Zato je operatorima potrebna povratna informacija o dešavanjima na uređajima kako bi mogli da reaguju na promjene. Povratnu informaciju operatoru omogućuju korisnici programske podrške. Korisnici programske podrške su u mogućnosti da preuzmu podatke o signalu koji digitalni televizijski prijemnik dobija, o statusu svake od komponenti digitalnog prijemnika i da te podatke proslijede poslužitelju (eng. *Server*), kao i da omogući oporavak od greške u toku rada.

Ukoliko postoji aktivna veza između interneta i uređaja, povratna informacija se šalje periodično. U suprotnom se podaci 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) koji garantuje siguran prenos. Protokol definiše postojanje TR-069 poslužitelja 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). Takođe, standard dozvoljava pravljenje sopstvenog modela podataka uz praćenje standardom precizno definisanih pravila.

Rad je organizovan u sedam poglavlja:

- Prvo poglavlje sadrži kratak uvod u rad.
- Drugo poglavlje sadrži opis Android platforme, osnove TR-069 i XMPP protokola.
- U trećem poglavlju dat je opis procesa projektovanja aplikacije.
- Četvrto poglavlje sadrži opis rješenja.
- Način ispitivanja i prikaz rješenja je sadržan u petom poglavlju.
- Unutar šestog poglavlja sadržan je kratak pregled šta je urađeno u ovom radu i kakvi su dalji pravci razvoja.
- Sedmo poglavlje sadrži spisak korištene literature tokom izrade rada.

2. Teorijske osnove

U okviru ovog poglavlja date su osnove strukture TR-069 korisničke biblioteke, namjena Android platforme i osnovni pojmovi o XMPP komunikacionom protokolu.

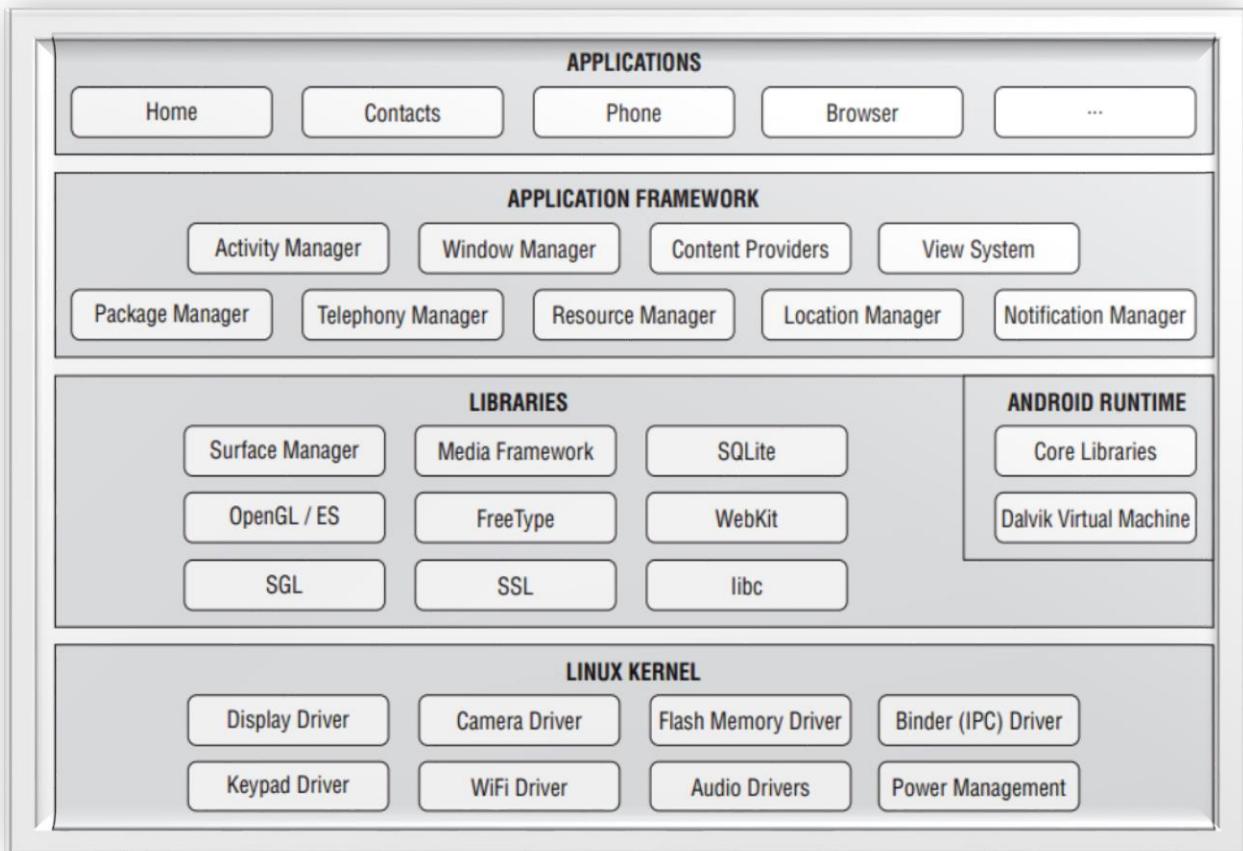
1. Android platforma

Android platforma je trenutno najrasprostranjenija platforma za mobilne telefone, zasnovana na Linux jezgru. Prilagođena je da se može koristiti na većini mobilnih uređaja, uključujući pored mobilnih telefona i tablet računare, laptop računare. Našla je primjenu i u ostalim namjenskim uređajima kao što su: pametni satovi, pametni televizori, prijamnici digitalnog TV signala [2].

Android platforma je stek programske podrške i sastoji se od:

- Jezgra Linux operativnog sistema (eng. Linux kernel).
- Sistemskih biblioteka (eng. Libraries).
- Izvršnog radnog okruženja (eng. Android Runtime).
- Okruženja za razvoj aplikacija (eng. Application Framework).
- Aplikacija (eng. Application).

Na slici 2.1. je prikazana arhitektura Android platforme.



Slika 2.1 - Arhitektura Android programskog steka

Linux jezgro (eng. *Linux Kernel*) – Android platforma ne predstavlja Linux distribuciju, ali je zasnovana na njegovom jezgru [3]. Linux jezgro upravlja dodjelom radne memorije, posjeduje mogućnost upravljanja procesima, posjeduje razdvojenost fizičkog sloja od programa i operativnog sistema otvorenog koda (eng. Open Source). Upravljački blokovi u okviru Android-a su:

- Binder – Komunikacija internih procesa (eng. *IPC - Inter Process Communication*).
- Ashmem – djeljena memorija između procesa.
- Upravljački program za upravljanje potrošnjom energije.
- Upravljački programi za ispis.
- Alarm.
- Upravljački programi za oslobađanje memorije.

Izvršno radno okruženje (eng. *ART – Android Runtime*) – je okruženje za izvršavanje aplikacije koje koristi Android platformu. ART omogućava da se Java bajtkod prevodi u nativne (C/C++) instrukcije i potom izvršava u radnom okruženju uređaja.

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

Sistemske biblioteke (eng. *Libraries*) – implementirane su u programskim jezicima C/C++ i obezbjeđuje osnovne funkcionalnosti Android platforme.

Okruženje za razvoj aplikacija (eng. *Application Framework*) - napisano je u programskom jeziku Java. Java klase i sprege pružaju podršku Android aplikacijama i povezuju Android aplikacije sa nativnim slojem.

Aplikacije (eng. *Applications*) – su 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/prozoru sa kojim korisnici mogu komunicirati kako bi nešto uradili. Jedna aplikacija može imati više aktivnosti i sve one su međusobno nezavisne (eng. *Loosely coupled*).
- **Servisi** (eng. *Services*) – predstavljaju komponente koje se izvršavaju u pozadini. Servisi su namjenjeni operacijama koje se dugo izvršavaju.
- **Dobavljači sadržaja** (eng. *Content Provider*) – primarna namjena ove komponente je da omogući dijeljenje podatka između aplikacija/procesa. Da bi podaci neke aplikacije bili dostupni nekoj drugoj, potrebno je da ta aplikacija definije spregu prema svojim podacima.
- **Primaoci emitovanih signala** (eng. *Broadcast Receivers*) – osnovna namjena ove komponente je da prihvata Intent objekte koji se šalju širom sistema metodom *sendBroadcast()*. Prilikom registrovanja prijamnika definiše se i *IntentFilter* objekat kako bi on dobijao samo određene *Intent* objekte.

Prvi sloj aplikacija se sastoji od nekoliko osnovnih komponenti:

- **Home** – prikazuje aplikacije, programe i prečice. Takođe podržava promjenjivu pozadinu.
- **Phone** – podržava klasične telefonske funkcije kao i kontrolu poziva, konferencijske razgovore, sporedne usluge i laku integraciju sa aplikacijom Contacts.
- **Web Browser** – je pretraživač baziran na WebKit-u sa svim njegovim mogućnostima.
- **Email** – omogućava upravljanje i reprodukciju sadržaja kodiranih na razne načine.
- Ostale komponente koje se nalaze u ovom sloju su **Alarm Clock, Calculator, Calendar, Camera, Contancts, Setting, Voice Diler** i ostale.

Sljedeći sloj predstavlja sloj biblioteka u Android platformi:

- **Surface Manage** – prikaz prozora za pojedine aplikacije. Biblioteka za upravljanje grafičkom spregom ujedno je zadužena i za pravilno iscrtavanje različitih aplikacionih komponenti u vremenu i prostoru.
- **OpenGL/ES** – biblioteka koja služi za grafički 2D i 3D prikaz.
- **Media Framework** – odgovoran za reprodukciju i rad sa multimedijalnim datotekama.
- **FreeType** – biblioteka koja služi za vektorsku rasterizaciju fonta.
- **SSL** (eng. *Secure Socket Layer*) – omogućuje sigurnosnu komunikaciju preko interneta.
- **SQLite** – Biblioteka koja pruža programsku podršku za manipulisanje bazama podataka. Aplikacije je mogu koristiti kao način da skladište podatke.
- **Libc** (eng. *System C library*) – implementacija standardne sistemske biblioteke programskog jezika C izvedene iz operativnog sistema BSD.
- **WebKit** – jezgra web pretraživača koji podržava Javascript i ostale standarde na mobilnim uređajima.

Okruženje za razvoj aplikacija (eng. *Application Framework*) sadrži komponente koje koriste sve aplikacije uređaja:

- **Activity Manager** – upravljanje životnim ciklusom aplikacije.
- **Package Manager** – sadrži informacije o aplikacijama koje su instalirane na sistemu.
- **Windows Manager** – upravlja aplikacijskim prozorima.
- **Telephony Manager** – API koji se koristi pri izradi aplikacija za upravljanje pozivima.
- **Content Provider** – organizovan za zajedničko korištenje podataka od strane više aplikacija.
- **Resource Manager** – služi za pohranu dijelova aplikacija koji nisu kod (npr. slike).
- **Location Manager** – upravljanje lokacijskim temeljima usluge.
- **Notification Manager** – upravlja obavještenjima i događajima (npr. dolazeće poruke, nadolazeći sastanak).
- **View System** – sadrži bazu gotovih grafičkih prikaza i alata.

JNI (eng. *Java Native Interface*) predstavlja programsku platformu koja omogućava *Java* programima da pozivaju i da budu pozvani od strane drugih programa i biblioteka koje su napisane u drugim programskim jezicima, najčešće C/C++ i asembler.

JNI se koristi u situacijama kada nativni *Java* kod ne može da odgovori na specifične zahtjeve programera ili kada je potrebno koristiti specifične funkcije operativnog sistema. Takođe se može koristiti i kada je potrebno modifikovati ili koristiti već postojeću aplikaciju koja je napisana u drugom programskom jeziku. Osim u ovim slučajevima *JNI* se koristi i kada su performanse zvršavanja neke funkcije od presudnog značaja jer se C i asemblerski kod izvršavaju mnogo brže od nativnog *Java* koda.

Ipak *JNI* osim ovih prednosti sa sobom donosi i neke mane.

Prvo, sav programski kod koji *Java* izvršava preko *JNI*-a mora biti unaprijed preveden upravo za operativni sistem na kome će se finalni program izvršavati. To ustvari znači da ovakav *Java* program gubi prenosivost, po čemu je *Java* i poznata. Ovaj problem se može djelimično riješiti tako što će se sav kod koji se izvršava uz pomoć *JNI*-a unaprijed prevesti za sve poznatije operativne sisteme.

Drugi problem koji *JNI* donosi je gubitak sigurnosti koju *Java* pruža. Dok je *Java* *type-safe* i siguran jezik, nativni jezici poput C-a i C++-a nisu. Kao rezultat toga, treba povećati oprez kada se razvijaju takve aplikacije. Metoda u nativnom dijelu koda koja ne radi kako bi trebalo, može srušiti cijelu aplikaciju.

Nepisano pravilo, bilo bi dobro koristiti *JNI* u najmanjem mogućem broju klasa i čim više izolirati nativni kod od ostatka aplikacije.

Ključna riječ 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 mora da posjeduje određeni opis u prototipu. Prototipovi nativnih funkcija se generišu sa pozivom *javah* alata, koji je standardni dio *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.

Primjer: *Javah -jni imeKlase*

gdje je *imeKlase* naziv klase u kojoj su zapisane definicije. U nativnoj funkciji se nalazi odgovarajući dio koji odgovara svakom primitivnom tipu podatka u *Javi*, ali i nekim *Java* klasama.

Alat *javac* je takođe standardni Java alat. Na taj način se generiše datoteka koja sadrži prototip nativne funkcije napisane u C programskom jeziku.

Mnoge standardne Java biblioteke direktno zavise od JNI platforme – kada je potrebno korisniku omogućiti pristup datotekama i grafičkim ili zvučnim funkcijama operativnog sistema. Osim standardnih Java biblioteka, postoje i mnoge nestandardne koje dodaju još više funkcionalnosti i pritom se oslanjaju na JNI, na primjer za pristup USB uređajima i slično.

Android servis

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

- Sistemski servisi.
- Aplikativni servisi.

Postoji mnogo sistemskih servisa kao što su kopiranje, alarm, rad sa mrežom, obavještenja i slično. Oni postoje u uređajima i do njih se dolazi preko odgovarajućih rukovaoca.

Aplikativni servisi se naknadno instaliraju.

Lokalni servisi su podvrsta aplikativnih servisa i izvršavaju se u istom memorijskom prostoru gdje i aplikacija koja ih je pozvala. Postoje dva načina komunikacije sa lokalnim servisima:

- Metode **startService()** i **stopService()**. Metoda **startService()** kreira servis i poziva njegove metode **onCreate()** i **onStartCommand()**.
- Metoda **bindService()**.

Realizacija lokalnog servisa se vrši u tri koraka:

- Kreiranje klase.
- Prijava u manifestu.
- Pisanje korisnikovog koda.

IPC (eng. *Inter Process Communication*) je u Androidu implementiran korištenjem **Binder** podsistema. Binder omogućava komunikaciju između procesa i zasniva se na korisnik-poslužitelj arhitekturi. Korisnik poziva metode poslužitelja.

Procedura udaljenog poslužioca je:

- Definicija udaljenog poslužitelja na osnovu AIDL datoteke.
- Realizacija udaljenog poslužitelja na osnovu AIDL datoteke.
- Prijavljanje poslužioca u manifestu.
- Realizacija korisnikovog koda:

- Vezivanje za poslužitelja.
- Poziv metode AIDL sprege.
- Odvajanje korisnika od poslužitelja.

AIDL (eng. *Android Interface Definition Language*) je deklarativni jezik za opis metoda poslužitelja, zasnovan na IDL (eng. *Interactive Data Language*) jeziku. Uglavnom se definišu sprege koje će implementirati dio poslužitelja, odnosno servis. Sintaksa je slična Java sintaksi sa sledećim tipovima:

- Svi Javini primitivni tipovi.
- String.
- List.
- Map.
- CharSequence.

Jedna sprega je implementirana od strane jednog servisa. Ako su korisnik i poslužitelj 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 vrijednost ove metode je instanca *IBinder* klase. U njoj je kreirana klasa nasljednica klase *IRemoteServiceExampleStub*, koja je mašinski generisana klasa.

Prijava servisa u manifestu – Servis se prijavljuje unutar *application* oznake, kao *service* tag. Obavezan atribut je *android:name* (puno ime servisa). *Intent filter* je podoznaka koja definiše akciju koja pokreće servis.

Povezivanje korisnika na servis – Pozivom metode *bindService* (*Instant*, *Connection*, *Flag*) povezuje se klijentska aplikacija i servis. Ona nasljeđuje *ServiceConnection* klasu koja posjeduje dvije potvrđne metode *onServiceConnected()* i *onServiceDisconnected()*.

Metoda *onServiceConnected()* se poziva nakon uspostave veze sa servisom, a poziv *onServiceDisconnected()* nastupa prilikom prekidanja veze sa servisom.

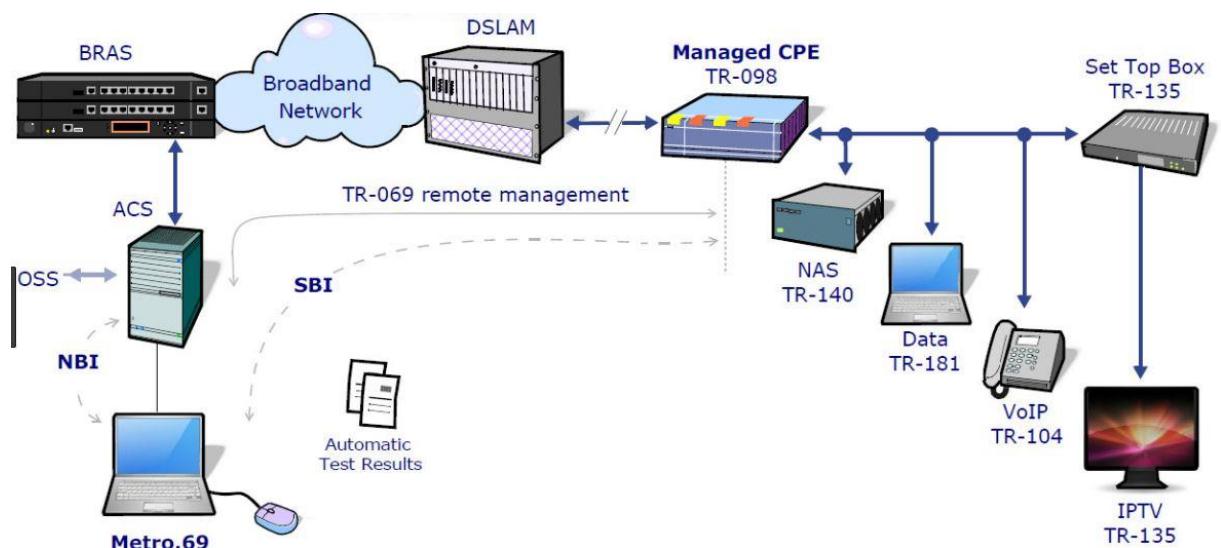
Povratni poziv – *Binder* je mehanizam za međuprocesnu komunikaciju, što daje servisu mogućnost poziva metoda na strani korisnika. Ako je posao koji servis obavlja dugotrajan, na servisu se kreira *AsyncTask* i kontekst izvršavanja se vraća korisniku pozivaocu. Korisnik saznaće o završetku posle tako što ga servis obavještava povratnim pozivom. Realizacija ovog mehanizma zahtjeva dodavanje još jedne AIDL (eng. *Android Interface Definition Language*) datoteke i kod korisnika i kod servisa, koja će sadržati metodu koju servis poziva kada se posao završi.

2. TR-069 komunikacioni protokol

TR-069 (eng. *Technical Report 069*) definiše protokol na ISO OSI aplikativnom nivou. Korištenjem ovog protokola veza se uspostavlja između krajnjeg korisnika uređaja (eng. *CPE – Customer Premises Equipment*) i automatsko-konfiguracionog poslužitelja (eng. *ACS – Auto-Configuration Server*). Protokol se zasniva na bidirekcionoj SOAP/HTTP vezi. Protokolom je opisan 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 model podataka namijenjen je STB uređajima, a TR-106 definiše osnovne podatke za uspostavu TR-069 veze [5].

Neke od osnovnih funkcionalnosti koje pruža TR-069 protokol su:

- Nadgledanje statusa i performansi krajnjeg uređaja i njihova dijagnostika.
- Automatska konfiguracija i dinamičko omogućavanje usluga.
- Rukovane programskom podrškom krajnjeg uređaja.



Slika 2.2 Primjer okruženja TR-069 protokola

Široka primjena TR-069 protokola je omogućila da CPE uređaji mogu biti različitog tipa, pri tome ne postoji razlika u protokolu po kome se prenose podaci do ACS-a, nego se razlikuju 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.

Poruke koje se prenose putem TR-069 protokola su standardizovane, gdje je za svaku od njih definisana sintaksa poruke, način rukovanja, kao i rukovanje u slučaju greške.

Modeli podataka su standardizovani kao stablo u kome postoje sljedeći tipovi čvorova:

- Korijenski čvor.
- Objekat.
- Višestruki objekat.
- Parametar.

Stablo je struktuirano tako da parametar uvijek predstavlja njen list, dok su čvorovi stabla objekti koji kao potomke mogu da sadrže nove objekte ili parametre. Objekti mogu biti definisani kao višestruki tj. može postojati više instanci jednog objekta koji se numerišu u rastućem redoslijedu.

Korijenski čvor omogućava da se objedini više modela podataka u jedno stablo. Objekti predstavljaju čvor u stablu koji ne može biti list, koristi se za bolje struktuirane podatke.

Parametrima se pridružuje i niz atributa kojim se vrši konfiguracija uređaja od strane ACS poslužitelja.

Atributi su:

- Nivo notifikacije.
- Lista pristupa.

Modelima se definišu podešavanja koja se mogu promijeniti od strane ACS servisa, ta podešavanja su:

- Mogućnost upisa vrijednosti parametara.
- Ograničenja vrijednosti parametara, definisana enumeracija ili granična vrijednost parametara.
- Obaveza slanja vrijednosti parametara prilikom iniciranja TR-069 sesije.
- Tip podataka datog parametra.
- Mogućnost nadgledanja promjene vrijednosti parametara u realnom vremenu.
- Početna podrazumijevanja vrijednosti parametara.
- Ograničenja broja višestrukih čvorova ispod pretka.
- Da li je čvor definisan u tom profilu modela podataka.
- Jedinica mjere date vrijednosti parametara.

Tipovi podataka definisani standardom su:

- Niz karaktera (eng. *String*).
- Cjelobrojni (eng. *Integer*).
- Cjelobrojni prošireni (eng. *Long*).
- Neoznačeni cjelobrojni (eng. *Unsigned Integer*).
- Neoznačeni cjelobrojni prošireni (eng. *Unsigned Long*).
- Logički (eng. *Boolean*).
- Vremenski (eng. *dateTime*).
- Binarni (eng. *Base64*).
- Heksadecimalni (eng. *hexBinary*).

Prilikom pristupa bilo kom čvoru u stablu mora se navoditi njegova cijela putanja od korijenskog čvora, gdje se kao separator između imena čvorova navodi tačka. Kao npr. *KorijenskiČvor.Objekat.VišestrukiObjekat.Indeks.Parametar*

3. XMPP komunikacioni protokol

XMPP (eng. *Extensible Messaging and Presence Protocol*) komunikacioni protokol baziran je na XML-u.

X – eXtensible: definisan u otvorenom standardu, koristi otvoren sistem pristupa i razvoja.

M – Messaging: dio XMPP protokola koji korisnik vidi (npr. razmjena poruka u stvarnom vremenu).

P – Presence: identifikatori koji na poslužitelju definišu stanje XMPP entiteta, te spremnost entiteta na primanje poruka.

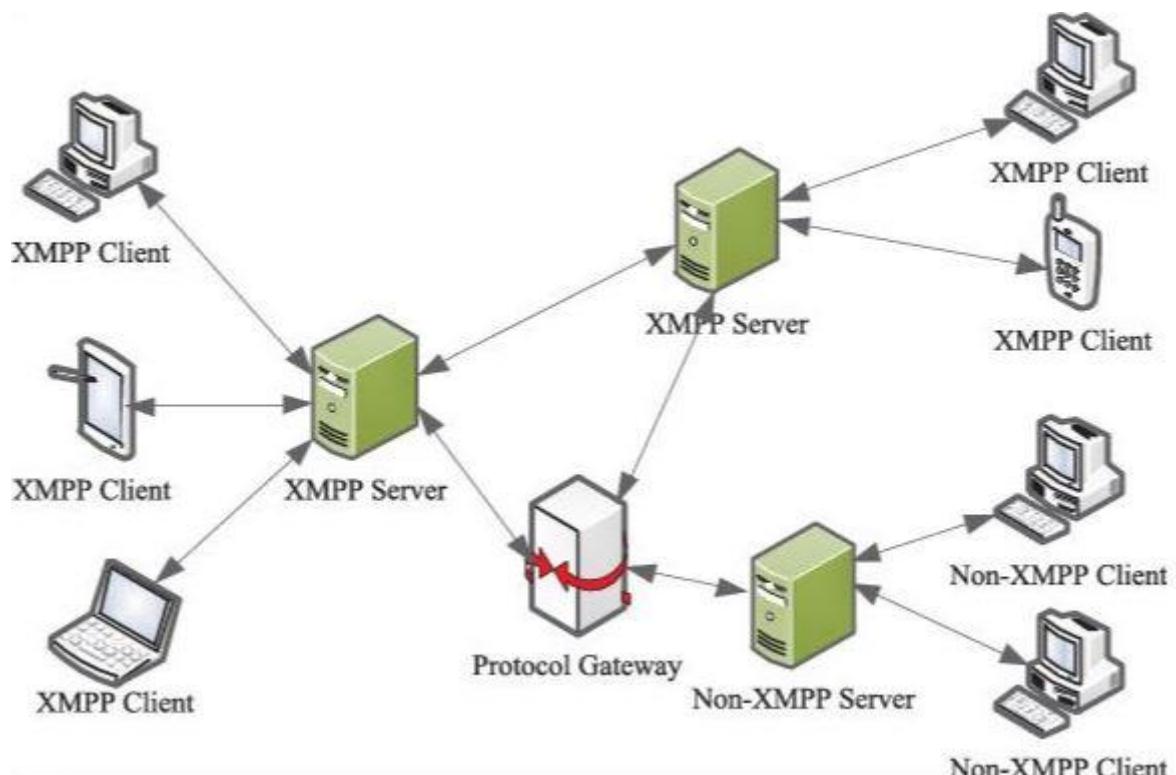
P – Protocol: XMPP je protokol, odnosno skup standarda koji omogućavaju međusobnu komunikaciju dva sistema. Široko je rasprostranjen na cijelom internetu.

XMPP protokol se ranije nazivao *Jabber*, kreiran je 1999. od strane *Jabber open-source* zajednice za potrebe baznih poruka (eng. IM, *Instant Messaging*), prikazivanje informacija, održavanje liste kontakata u približno realnom vremenu. Dizajniran je da bude fleksibilan sa mogućnostima proširenja. XMPP protokol je takođe korišten za sisteme za signalizaciju VoIP, video sadržaja, prenos datoteka (eng. *File Transfer*), video igre, IoT (eng. *Internet of Things*) aplikaciju kao pametna mreža i servise društvenih mreža. Za razliku od većine IM protokola, XMPP je definisan kao otvoreni standard i koristi pristup otvorenih sistema razvoja i aplikacija

po kojima bilo ko može implementirati XMPP servise i povezivati sa rješenjima drugih organizacija [6].

Jabber je besplatna XML platforma otvorenog koda slična Linux-u. Za razliku od ostalih sistema brzog slanja poruka, Jabber protokol funkcioniše na principu *e-mail* odnosno koristi distribuiranu arhitekturu. Dodaje sufiks za svaku adresu nakon “@” znaka, čime adresira poslužitelja. Takav način adresiranja omogućava Jabber poslužiteljima čitanje adresa, pa se zna kako doći do drugih adresa iz različitih sistema. Temeljne tehnologije formirane su pod nazivom **XMPP** (eng. *Extensible Messaging and Presence Protocol*) na IETF-u 2004. godine.

Otvoreni XMPP protokol se nadograđuje i proširuje dodacima. Format poruke koristi se za prenos različitih vrsta poruka između proizvoljnih entiteta u realnom vremenu. Poruke se šalju putem XMPP poslužitelja (JID-ova). Svaki JID (eng. *Jabber ID*) sadrži i domenski dio, za određivanje poslužitelja na kojeg se korisnik spaja (npr. *user@mydomain.org*).



Slika 2.3 XMPP arhitektura

Arhitektura XMPP umrežavanja je slična arhitekturi elektronske pošte: svi su u mogućnosti da uspostave sopstvenu XMPP infrastrukturu poslužitelja bez potrebe glavnog centralnog poslužitelja [12].

Decentralizovana korisnik-poslužitelj arhitektura omogućava razdvajanje obaveza. Strana poslužitelja obično je zadužena za sigurnost, autentifikaciju korisnika, enkripciju poruka i slično. To omogućava programerima koji razvijaju aplikacije za korisnike da se mogu više usredotočiti na korisničko iskustvo. Za razliku od Web arhitekture, u XMPP arhitekturi poslužitelji su

međusobno povezani. Kada korisnik šalje poruku drugom korisniku koji se nalazi na drugom domenu, on se povezuje na svoj poslužitelj koji se tada direktno povezuje na poslužitelj na kome se nalazi primatelj. Poruka na taj način ne prolazi kroz posredničke poslužitelje kao što je slučaj kod e-mail arhitekture.

Primjeri XMPP protokola su *CommuniGate Pro*, *Apache Vysper*, *iChat server*, *Opendire*, *Jerry Messenger*, *Siemens OpenScape*, *Tigase*, *Wokkel*.

Prednosti XMPP:

- **Decentralizovana arhitektura**

Arhitektura XMPP umrežavanja je slična arhitekturi elektronske pošte, svako je u mogućnosti da uspostavi sopstvenu XMPP infrastrukturu poslužitelja bez potrebne za glavnim centralnim poslužiteljem.

- **Otvoreni standardi**

Nema direktnе povezanosti sa proizvođačima.

- **Istorija**

XMPP tehnologije su još u upotrebi od 1999. Godine. Višestruke implementacije XMPP standarda postoje za korisnika, za komponente prijamnika i biblioteke kodova.

- **Bezbijednost**

XMPP poslužitelje možemo izdvojiti iz javne XMPP mreže. Implementirani su solidni bezbjednosni protokoli u jezgro XMPP specifikacije.

- **Fleksibilnost**

Proizvoljene funkcionalnosti se mogu nadograditi na XMPP.

Nedostaci:

- Nema optimizovan **QoS** (eng. *Quality of Service*).

Protokol podataka XMPP protokola uglavnom čini oko 70% na poslužitelju, a od toga 60% ponavljanja.

- Komunikacija bazirana na tekstu XML formata.

S obzirom da XML zapisi imaju poseban format, XMPP protokol ima malo veći mrežni paket prilikom slanja poruka na adrese više primalaca.

- S obzirom na dužinu jednog XML fajla kojim je kodiran XMPP protokol, ne može da se obezbijedi modifikacija binarnih podataka. Sa tim, upotreba ekstremnog protokola za transfer fajlova kao HTTP je neminovna.

- Nema mogućnost *end-to-end* enkripcije

Prenos XML-a

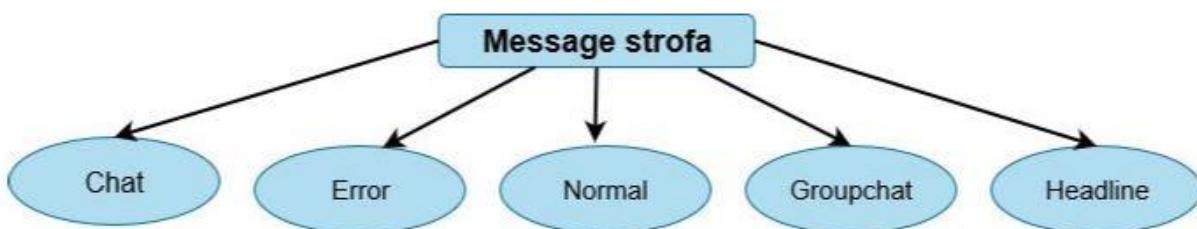
XMPP je tehnologija za prenos (eng. Streaming) XML-a. Kada se pokreće sesija sa XMPP poslužiteljem, otvara se dugotrajna TCP vezu i onda se uspostavlja XML prenos do poslužitelja (poslužitelj takođe započinje prenos u suprotnom smjeru). Nakon uspostave XML toka podataka sa poslužiteljem, korisnik i poslužitelj mogu da razmjene tri specijalna XML isječka putem toka podataka: `<message/>`, `<presence/>` i `<iq/>`. Ovi isječci se nazivaju XML "strofama" (eng. *Stanza*), predstavljaju osnovne tekstualne strukture XMPP-a i nakon uspostave XML toka podataka može se slati neograničen broj XML strofa preko istog toka [6].

Nekoliko faktora određuje značenje strofe:

- Nazivi elemenata strofe - to su poruka, prisutnost ili strofa za zahteve i odgovore. Svaki tip strofe se preusmjerava drugaćije na poslužitelja i prihvata drugaćije od korisnika.
- Vrijednost tipa atributa, koja varira u zavisnosti od vrste date strofe. Ova vrijednost diferencira način na koji će se tip strofe obraditi od strane primaoca.
- Elementi naslijednici koji definišu veličinu transportne strofe. Veličina transportne strofe može da se predstavi korisniku ili da se procesuira po nekom automatizmu što je i određeno specifikacijom koja definiše imenski prostor (eng. *namespace*) za veličinu transporta.

Poruka

XMPP <message/> : strofa je osnovna "gurajuća" (eng. "*store and push*") metoda za dobijanje informacija sa jednog mesta na drugo. S obzirom da su poruke nepotvrđene, tj one su na neki način "ispali i zaboravi" mehanizam za brzo dobijanje informacija sa jednog mesta na drugo. Na primer, prenosi poruke između dvije krajnje tačke. Razlikujemo pet vrsta ovih strofi s obzirom na atribut *type*.



Slika 2.4 Tipovi strofi poruka

Tip strofi:

- *Chat* ova poruke se šalje na entitete za jedan na jedan.
- *Error* se koristi kada entitet koji je primio poruku primjeti neku vrstu pogreške, posle čega vraća korisniku strofu ovog tipa.
- *Normal* je tip koji označava obične poruke koje šalje korisnik. Korisnik može, a ne mora dobiti odgovor na njih.
- *Groupchat* se koristi za poruke namjenjene sobama za razmjenu poruka sa više korisnika.
- *Headline* se koristi za obavještenja.

```
<message from='niepin@tml.hut.fi/laptop'
  to='peter@hut.fi'>
<body>Hello</body>
</message>
```

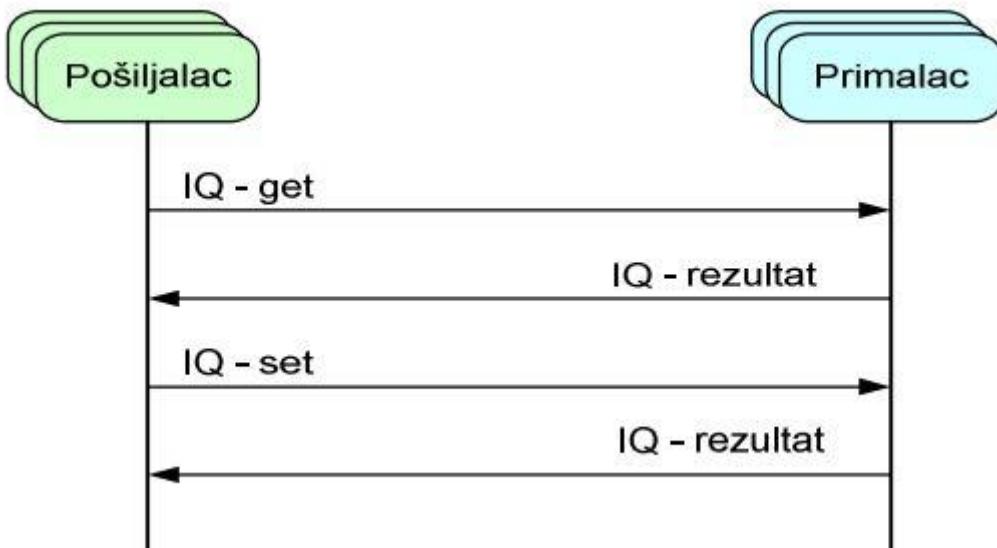
Prisutnost

XMPP <presence> : strofa se koristi za prikazivanje prisutnosti entiteta na mreži. Pod pojmom prisutnosti podrazumijevaju se na mreži (eng. *online*) i nije na mreži (eng. *offline*) stanja, ali i neka dodatna poput odsutan (eng. *away*) i ne smetaj (eng. *do not disturb*). Kako bi jedan klijent mogao vidjeti prisutnost drugoga, mora za to biti autoriziran, to jest, mora poslati zahtjev.

```
<presence from='niepin@tml.hut.fi/laptop'
  to='peter@hut.fi'>
```

Strofa za zahtjeve i odgovore (eng. *IQ, Info/Query*)

<iq> : to je mehanizam odgovora na zahtjev, sličan HTTP, koji omogućava entitetima da postavljaju zahtjeve i primaju odgovore jedni od drugih, na primjer prenos datoteka.



Slika 2.5 Razmjena IQ strofa

Ova strofa sadrži samo jedan dodatni element koji definiše šta primaoc treba učiniti kada primi zahtjev. Kao i kod strofe za poruke, i ovdje postoji atribut *id* koji služi za praćenje zahtjeva. Tip atributa koji se može koristiti u strofi IQ ima četiri vrijednosti koje se mogu koristiti:

- *get* pošiljalac traži informacije.
- *set* pošiljalac šalje informacije primaocu, ili postavlja zahtjev primaocu.
- *result* je obavezni odgovor primaoca set ili get strofe koji vraća tražene informacije ili odgovor na zahtjev.
- *error* se vraća pošiljaocu get ili set strofe kada primalac nije mogao obraditi njegov zahtjev.

```

<iq type='get'
    from='niepin@tml.hut.fi/laptop'>
    <query xmlns='jabber:roster'/>
<iq>

```

3. Koncept rješenja

U okviru ovog poglavlja data je analiza zadatog problema, a najveći akcenat je stavljen na XMPP protokol.

Otkrivanjem novih protokola, *Broadband Forum TR-069* programi su se okrenuli prema rješenju koje koristi razmjenu poruka koristeći protokol XMPP (eng. *Extensible Messaging and Presence Protocol*).

Komunikacija između klijentskih procesa se u slučaju korištenja XMPP tehnologije odvija prema modelu korisnik-poslužitelj. Klijentski procesi ne komuniciraju direktno, već posredno pomoću poslužitelja. Takođe, arhitektura je decentralizovana što omogućava klijentskim programima jednostavnost tj. ne moraju brinuti o sigurnosti mreže i podataka koji se njom prenose. Pošto ACS prosljeđuje zahtjev za povezivanje preko XMPP-a, nema razloga za brigu o prevodu mrežnih adresa (eng. *NAT, Network Address Translation*). Kako bi klijent mogao započeti komunikaciju s drugim klijentom, prvo se mora otvoriti stalna TCP (eng. *Transmission Control Protocol*) veza između XMPP poslužitelja i korisnika [10]. Protokol UDP (eng. *User Datagram Protocol*) ne podrazumjeva stalnu vezu, zbog čega je XMPP bolje rješenje za prenos podataka od STUN (eng. *Session Traversal Utilities for NAT*) koji koristi UDP protokol.

CWMP (eng. *CPE WAN Management Protocol*) može se koristiti za daljinsko upravljanje CPE povezanim preko LAN kroz Gateway. Kada ACS upravlja priključenim uređajem preko NAT Gateway-a ili Gateway-a sa omogućenim zaštitnim zidom (gdje se uređaj ne može direktno kontaktirati od strane ACS-a), CPE WAN Management Protocol se još uvijek može koristiti za upravljanje uređajima, ali uz ograničenja koja se odnose na zahtjev za povezivanje.

Procedure definisane u aneksu [5] omogućavaju ACS-u da pokrene sesiju sa bilo kojim uređajem, uključujući uređaje koje ACS ne može direktno kontaktirati. Ovo omogućava ekvivalentnu funkcionalnost zahtjeva za povezivanje, ali koristi različit mehanizam baziran na XMPP (eng. *Extensible Messaging i Presence Protocol*) da bi se odgovorilo ovom scenariju.

Procedura

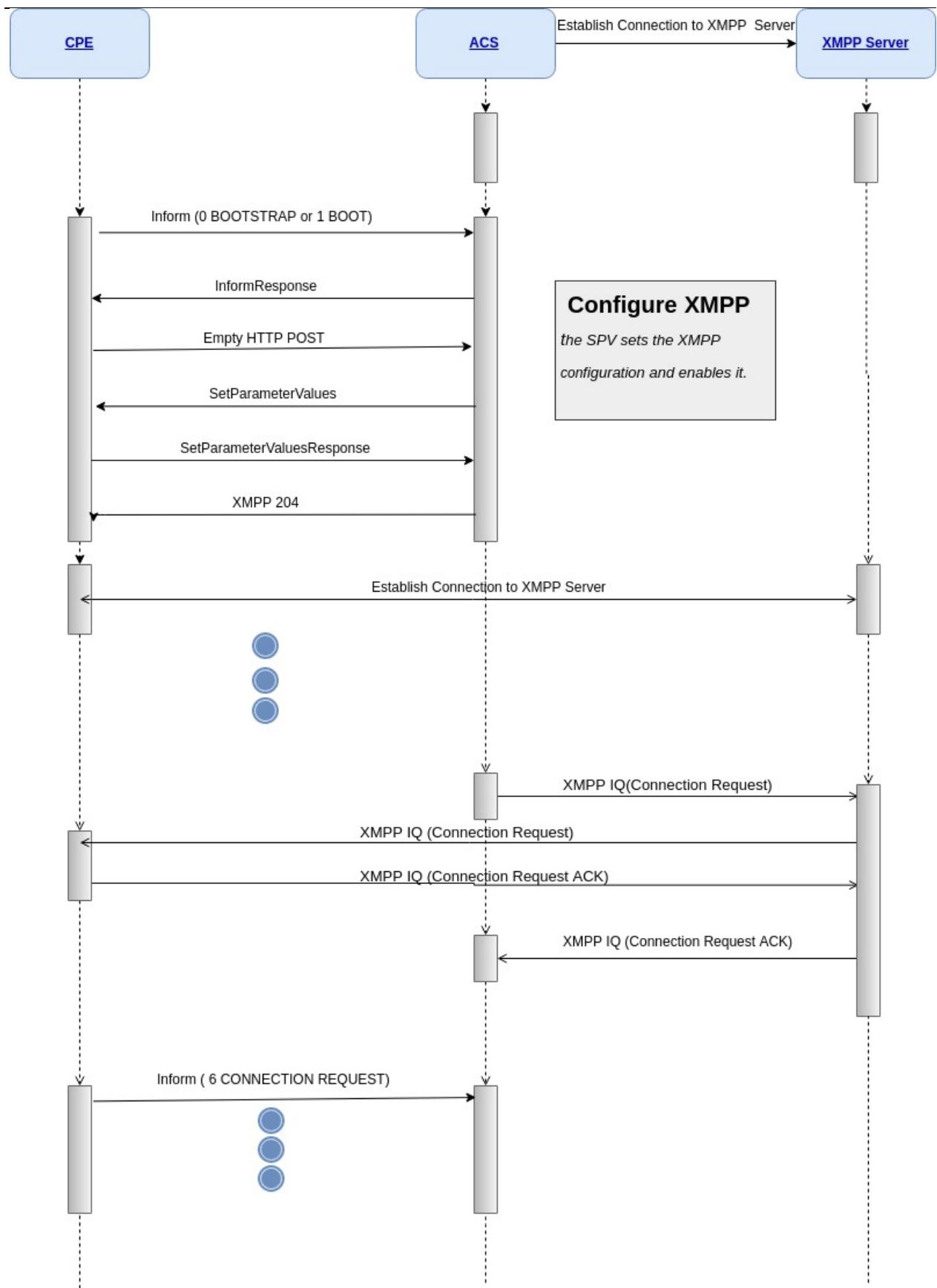
Da bi se prilagodio mogućnostima ACS-a da izda ekvivalentne zahtjeve za povezivanje potrebno je sljedeće:

- CPE mora biti u stanju da uspostavi sigurnu i autentifikovanu vezu sa XMPP poslužiteljem
- CPE mora biti u stanju da održava vezu sa XMPP poslužiteljem

Da bi se postigle navedene stavke aneksu [15] definiše odrešeni skup poruka koje treba proslijediti preko standardne XMPP mreže .

Zahtjev za povezivanje sa CPE se sumira na sljedeći način:

- ACS uspostavlja vezu sa XMPP poslužiteljem.
- ACS omogućava korištenje XMPP-a u uređaju konfiguriranjem XMPP-a.
- Uređaj uspostavlja XMPP vezu sa navedenim XMPP poslužiteljem.
- Kad god ACS želi uspostaviti vezu sa CPE-om, on može poslati XMPP Connection Request za XMPP poslužitelj.
- XMPP poslužitelj šalje XMPP IQ strofu na odgovarajući uređaj.
- Uređaj izdaje informacioni zahtjev ACS-u koji je naveden u njegovom parametru ManagementServerURL.



Slika 3.1 XMPP sekvenca poruke za vezu

CPE zahtjevi

CPE koji je u skladu sa ovim aneksom mora da podržava *XMPBasic:1* i *XMPConnReg:1* [16].

Kad god parametar *ManagementServer.Connection* referencira omogućen primjer table *XMPConnection*, CPE aneks mora da prati procedure definisane u XMPP za povezivanje sa XMPP poslužiteljem [15].

CPE mora:

- Odrediti javnu IP adresu XMPP poslužitelj.
- Otvoriti tok podataka (eng. *Stream*) na XMPP poslužitelju i prihvati XML tok podataka iz XMPP poslužitelja.
- Koristiti TLS za uspostavljanje, šifrovanje i sigurnu TCP komunikaciju sa XMPP poslužiteljem.
- Koristiti SASL (eng. *Simple Authentication and Security Layer*).
- Uvjeriti se da parametar *ManagementServer.ConnReqJabberID* sadrži istu vrijednost kao *JabberID* polje koje se nalazi u okviru XMPP instance. Instanca veze na koju upućuje parameter *ConnReqXMPPConnection*.
- Slušanje XMPP zahtjeva za povezivanje poruka i sa XMPP poslužiteljem.
- Održavanje TCP veze sa XMPP poslužiteljem korištenjem "*whitespace keepalive*" mehanizma.
- Reagovanje na te poruke kada stignu.
- Ako se veza sa XMPP poslužiteljem izgubi, ponovo je potrebno da se uspostavi.

Kada god parametar *ManagementServerConnReqXMPPConnection* referencira dostupnu instancu XMPPConnection table, CPE koji prati zahtjev ovog aneksa treba da:

- Uspostavi Connection Request XMPP konekciju prije uspostavljanja CWMP Session, gdje su "1 BOOT" ili "13 WAKEUP" kodovi događaja koji se dostavljaju. Tokom događaja promjene u parametru *ConnReqJabberID*, ovo će dozvoliti CPE-u da dostavi "4 VALUE CHANGE" kod događaja uz "1 BOOT" ili "13 WAKEUP" kod događaja i ukloniti potrebu za CPE da u istom momentu uspostavi drugu CWMP sesiju za dostavljanje pomenutih događaja kodova koji mijenjaju vrijednost.

XMPP veze ekripcije

Ukoliko se parametar *ManagementServerConnReqXMPPConnection* odnosi na omogućenu instancu XMPP-a, tabele veze CPE prateći zahtjeve ovog aneksa mora dogоворити сигруну vezu u slučaju da je TLS posmatran kao "*mandatory-to-negotiate*". XMPP veza mora biti ili u TLS enkripcioni XMPP kanal ili povezan preko WebSocket Secure konekcije [17].

XMPP kanal ovjere

Ukoliko se parametar *ManagementServerConnReqXMPPConnection* odnosi na omogućenu instancu XMPP table povezivanja, CPE prateći zahtjeve ovog aneksa se mora ovjeriti sa XMPP poslužiteljem, nakon što se uspostavi XMPP veza. XMPP veza se ovjerava korištenjem **SASL** (eng. *Simple Authentication and Security Layer*) protokola.

Parametri kao što su korisničko ime i šifra XMPP veze se koriste kao potvrda za SASL proceduru ovjere [16].

XMPP zahtjev povezivanja

U skladu sa aneksom, CPE mora uvijek slušati u slučaju da su poslate XMPP poruke kao zahtjev za povezivanje, koje dolaze sa dozvoljene liste Jabber ID-ova i koje su generisane od strane XMPP poslužitelja određene za *XMPP.Connection* instance koja se odnosi u sklopu parametra *ManagementServerConnReqXMPPConnection*.

Kada CPE primi XMPP poruku zahtjeva za povezivanje, poruka mora da je važeća i ovjerena.

XMPP poruka kao zahtjev za povezivanje je važeća samo kada su ispunjeni sljedeći uslovi:

- XMPP poruka kao zahtjev za povezivanje mora biti dostavljena u intervalu XMPP IQ strofe preko XML toka podataka koji je prethodno TLS oslobođen.
- XMPP poruka kao zahtjev za povezivanje mora biti dobro formatirana.
- "od" (eng. "from") adresa sadržana u sklopu XMPP IQ strofe mora da odgovara jednoj od adresa sadržanih u sklopu bazne liste *ManagementServer.ConnReqAllowedJabberIDs* parametra (osim ako je vrijednost tog parametra prazna, što znači da su sve adrese dozvoljene)
- Vrijednost korisničkog imena u sklopu zahtjeva za konekciju mora da odgovara vrijednosti *ManagementServer.ConnectionRequestUsername* parametra.

XMPP poruka za zahtjev za povezivanje je ovjerena ako i samo ako vrijednost šifre u sklopu zahtjeva "connectionRequest" odgovara vrijednosti ManagementServer.ConnectionRequestPassword parametra.

Nakon što CPE primi, uspješno potvrdi i odgovori na XMPP zahtjev za povezivanje, mora da uspostavi vezu sa predefinisanim ACS-om prateći generičke Connect Request zahtjeve i dalje kvalifikativno sljedećim tačkama:

- CPE bi trebalo da ograniči broj zahtjeva za povezivanje za određene CWMP krajnje tačke koje on prihvata u toku određenog vremena u cilju daljeg smanjenja mogućnosti odbijanja napada na servis. Ako CPE izabere da odbije zahtjev za povezivanje iz ovog pomenutog razloga, CPE mora da odgovori zahtjevu za povezivanje sa sledećom XMPP greškom:
error code 503 sa servis je nedostupan (service-unavailable).
- Ako se CPE već nalazi u sesiji sa ACS-om sa najmanje jednim CWMP krajnjom tačkom kada dobije jedan ili više zahtjeva za povezivanje, ne smije da prekine nikakvu sesiju protiv bilo koje krajnje tačke CWMP prerano. Zbog toga CPE mora da preduzme jednu od sljedećih alternativnih radnji:
 - Odbaci svaki zahtjev za povezivanje odgovaranjem sa *error code 503*.
 - Prateći završetak trenutne krajnje tačke CWMP sesije, pokreće tačno jednu novu sesiju u tom vremenu (bez obzira koliko zahtjeva za povezivanje je primljeno u toku prethodne sesije) koje uključuje "6 CONNECTION REQUEST" EventCode u Inform-u. Zahtjevi za povezivanje koji nisu prihvaćeni, moraju biti odbijeni (eng. *error code 503 i "service-unavailable"*). U slučaju da je nova sesija za krajnje tačke CWMP trenutno u sesiji, CPE mora momentalno pokrenuti nakon što je završena postojeća sesija i sve promjene iz te sesije su primjenjene.
 - Ako zahtjev za povezivanje nije ni za jednu krajnju tačku CWMP trenutno u sesiji, CPE može da pokrene novu sesiju sa zahtjevima krajnjih tačaka CWMP dok je trenutna sesija i dalje aktivna.

Ovaj zahtjev važi za zahtjeve za povezivanje koji su primjenjeni u bilo kom vremenskom intervalu kada je CPE posmatran u sesiji sa bar jednom krajnjom tačkom CWMP, uključujući period u kojem je CPE u procesu uspostavljanja sesije.

Ako je XMPP zahtjev za povezivanje neuspješno potvrđen i ovjeren, CPE mora da vrati IQ strofu sa skupom atributa koji su postavljeni na "error".

ACS zahtjevi

ACS prateći zahtjeve aneksa bi trebalo da ima mogućnost omogućavanja upotrebe XMPP-a na CPE, podržavajući ovaj aneks konfiguriranjem XMPP.Connection objekta, konfiguriranjem *ManagementServer.ConnReqXMPPConn* parametra koji se odnosi na XMPP.Connection objekat i opcionalno konfigurišući parametar *ManagementServer.ConnReqAllowedJabberIDs* koji sadrži listu dozvoljenih Jabber ID-ova koji pokreću XMPP zahtjev za povezivanje.

ACS prilagođen ovom aneksu mora da ima pristup XMPP vezi koja se čvrsto drži sljedećih pravila [16]:

- Poveže na XMPP poslužiteljem koji je sposoban da komunicira sa CPE kojem ACS namjerava da izda zahtjev za povezivanje.
- Otvori XML tok podataka na XMPP poslužitelja i prima XML tok podataka od XMPP poslužitelja.
Napomena: XML tokovi podataka su neusmjereni i ovaj XMPP mehanizam zahtjeva za povezivanje zahtjeva upotrebu dva XML toka podataka preko jedne TCP veze.
- Koristi TLS da otvorи enkripciju i ostvari TCP vezu sa XMPP poslužiteljem.
- Koristi SASL za autentifikaciju sa XMPP poslužiteljem.
- Šalje XMPP poruku kao zahtjev za povezivanje CPE-u koji podržava ovaj aneks.

XMPP veza enkripcije

ACS prateći pravila ovog aneksa mora imati pristup bezbjednoj vezi ako je TLS posmatran kao "*mandatory-to-negotiate*", XMPP zahtjev je enkriptovan korištenjem STARTTLS produžetka TLS protokola.

XMPP zahtjev za vezu

ACS u skladu sa ovim aneksom mora da pošalje XMPP poruku zahtjeva za vezu putem XMPP poslužitelj kojem ACS ima pristup.

Format XMPP poruke zahtjeva za vezu je definisan u aneksu K2.3. ACS se mora pridržavati sljedećih pravila kada šalje XMPP poruku zahtjeva za vezu:

- XMPP poruka zahtjeva za vezu mora biti dostavljena u sklopu XMPP IQ strofe preko XML toka podataka, koji je TLS obezbjeđen u skladu sa aneksom i provjeren.

- XMPP poruka zahtjeva za vezu mora biti dobro XML formatirana [18].
- "od" (eng. "from") adresa sadržana u sklopu XMPP IQ strofe mora da se poklapa sa jednom od adresa koje se nalaze unutar bazne liste parametara *ManagementServer.ConnReqAllowedJabberIDs* (osim ako je vrijednost tog parametra prazna, što znači da su sve adrese dozvoljene).
- Vrijednost korisničkog imena u sklopu "connectionRequest" mora da odgovara vrijednosti *ManagementServer.ConnectionRequestUsername* parametru CPE-a gdje je XMPP zahtjev za vezu poslat.
- Vrijednost šifre u sklopu "connectionRequest" mora da odgovara vrijednosti *ManagementServer.connectionRequestPassword* parametra gdje je XMPP zahtjev za vezu poslat.

Protok poruka

Adrese koriste notaciju (L@D/R) gdje:

- L predstavlja lokalni dio (*XMPP.Connection.{i}.Username*).
- D predstavlja domenski dio (*XMPP.Connection.{i}.-Domain*).
- R predstavlja izvršni dio (*XMPP.Connection.{i}.resource*).

gdje je *XMPP.Connection.{i}* instanca koja je referencirana od strane *ManagementServer.ConnReqXMPPConn* parametra.

Zahtjev za povezivanje

Poruka zahtjeva za povezivanje je formatirana u sklopu XMPP IQ strofe (eng. *Stanza*) i definisana je sa Broadband Forum connectionRequest kao što možemo vidjeti na slici 3.2. Ovaj element upotrebljava postojeće *ConnectionRequestUsername* i *ConnectionRequestPassword* parametar sadržane u *ManagementServer* objektu CPE-a gdje je XMPP poruka zahtjeva za vezu adresirana. Kako je XMPP veza enkriptovan i ispravan (autentikovan) kanal, nema potrebe da ove vrijednosti budu dodatno enkriptovane.

```
<iq from="[ACS-identity]" to="L@D/R" id="cr001" type="get">
  <connectionRequest xmlns="urn:broadband-forum-org:cwmp:xmppConnReq-1-0">
    <username>username</username>
    <password>password</password>
  </connectionRequest>
</iq>
```

Slika 3.2 Zahtjev za povezivanje

Odgovor na uspješan zahtjev

Pozitivan odgovor XMPP korisnika sadržan u sklopu CPE-a je prazan rezultat IQ strofe ACS-u kao što vidimo na slici 3.3.

```
<iq from="L@D/R" to="[ACS=identity]" id="cr001" type="result" />
```

Slika 3.3 Odgovor na uspješan zahtjev

Odgovor na neuspješan zahtjev za vezu

Neuspješan odgovor je greška IQ strofe koja ima vezu sa greškom koja je povezana sa elementom koji izvršava grešku.

Ako CPE nije dostupan, odgovor XMPP poslužitelj mora da sadrži *service-unavailable* grešku kao što se vidi na slici 3.4.

```
<iq from="L@D/R" to="[ACS=identity]" id="cr001" type="error">
<error type="cancel">
<service-unavailable xmlns="urn:ietf:params:xml:ns:xmpp-stanzas" />
</error>
</iq>
```

Slika 3.4 Odgovor na neuspješan zahtjev

U slučaju da CPE ne podržava “*urn:broadband-forum-org:cwmp:xmppConnReq-1-0*” name space, odgovor CPE-a mora da sadrži service-unavailable grešku kao što je prikazano na slici 3.5.

```
<iq from="L@D/R" to="[ACS=identity]" id="cr001" type="error">
<connectionRequest xmlns="urn:broadband-forum-org:cwmp:xmppConnReq-1-0">
<username>username</username>
<password>password</password>
</connectionRequest>
<error type="cancel">
<not-authorized xmlns="urn:ietf:params:xml:ns:xmpp-stanzas" />
</error>
</iq>
```

Slika 3.5 Odgovor sa service-unavailable

Ako XMPP zahtjev za vezu ne može biti autentikovan, CPE mora da pošalje not-authorized grešku kao što se vidi na slici 3.5.

XMPP zahtjev za implementaciju poslužitelja

XMPP poslužitelj korišten za slanje XMPP poruke zahtjeva za vezu između ACS-a i CPE-a koji odgovaraju ovom aneksu moraju se držati zahtjeva definisanih u XMPP-u. Zatim, preporučljivo je da XMPP poslužitelj budu konfigurisani na sljedeći način:

- TLS je mandatory-to-negotiate za sve korisnik-poslužitelj komunikacije
- TLS je mandatory-to-negotiate za sve poslužilac-poslužitelj komunikacije

Preporuke za konfiguraciju su održavanje gdje XMPP poslužitelj niju u potpunosti pod kontrolom davaoca usluga (eng. *Service Provider*), što bi impliciralo da su pod potencijalnim rizikom prilikom napada na bezbjednost.

Razmatranje bezbjednosti

Sljedeća sigurnosna razmatranja definisana u aneksu su identifikovana:

- XMPP specifikacija opisuje nekoliko bezbjednosnih razmatranja.
- Da bi se XMPP zahtjev za vezu razmatrao kao bezbjedan, ACS mora da konfiguriše dozvoljenu listu od Jabber ID-a iz adrese (parametar *ManagementServer.ConnREqAllowedJabberIDs*) i XMPP potvrda autentifikacije zahtjeva za povezivanje (*ConnectionRequestUsername* i *ConnectionRequestPassword Parameters* na objektu *ManagementServer*) tokom SSL-TLS sigurne CWMP sesije.
- Zahtjeva se da korisnik i poslužitelj podržavaju jaku zaštitu, koja se odnosi na korištenje zaštite tehnologija koje pružaju uzajamnu provjeru identiteta i integracija, ali to zavisi od instalirane postavke kako bi se obezbijedilo da su ovi detalji zaštite omogućeni. Obostrana potvrda identiteta je od posebne važnosti za bezbjedan mehanizam za zahtjev XMPP konekcije, posebno za poslužitelj–poslužitelj komunikaciju.

4. Programsко rješenje

U okviru ovog dijela rada biće opisan način pristupa rješenju problema i prikazane osnovne izmjene koje su urađene na polaznom problemu.

U repozitorijumu **tr-181-short.xml** se nalazi opis parametara. Opis sadrži podatke kao što su:

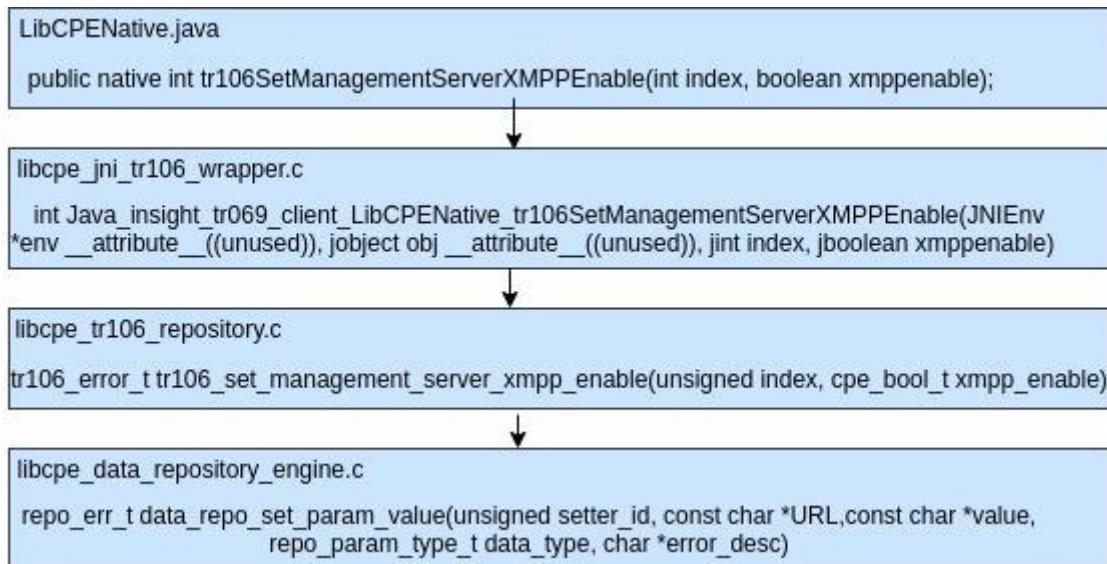
- da li može da se mijenja vrijednost parametra.
- kog tipa je parametar.
- kom objektu pripada dati parametar.

Na osnovu ovih informacija formira se repozitorijum u obliku XML datoteke za čuvanje podataka o parametrima iz odgovarajućeg modela podataka.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2<Device>
3  <Services/>
4<DeviceInfo>
5  <Manufacturer>
6    <ParamValue></ParamValue>
7    <Notification>0</Notification>
8    <AccessList/>
9  </Manufacturer>
10 <ManufacturerOUI>
11   <ParamValue></ParamValue>
12   <Notification>0</Notification>
13   <AccessList/>
14 </ManufacturerOUI>
15 <ModelName>
16   <ParamValue>IPSetTopBox</ParamValue>
17   <Notification>0</Notification>
18   <AccessList/>
19 </ModelName>
20 <Description>
21   <ParamValue>IPSetTopBox-AOSP</ParamValue>
22   <Notification>0</Notification>
23   <AccessList/>
24 </Description>
25 <ProductClass>
26   <ParamValue>IPSetTopBox</ParamValue>
27   <Notification>0</Notification>
28   <AccessList/>
29 </ProductClass>
```

Slika 4.1 repozitorijum za čuvanje podataka o parametrima

Funkcije za postavljanje parametara su identične, razlikuju se samo po tipu podataka, pravu čitanja/pisanja.



Slika 4.2 dijagram postavljanja vrijednosti parametara

libcpe_tr033_repository.c

tr106_error_t tr106_set_management_server_xmpp_jid(unsigned index, const char *xmpp_jid)

- funkcija za mapiranje parametara u stablo repozitorijuma i postavljanje vrijednosti parametara pozivom funkcije ***data_repo_set_param_value***.
- parametri:
 - indeks višestrukog čvora.
 - vrijednost 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)

- funkcija postavlja vrijednost parametra u repozitoriju

- prarametri su :

- Setter_id – identifikator entiteta koji postavlja vrijednost parametara.

-
- URL – putanja do parametara u stablu modela podataka.
 - Value – nova vrijednost parametara.
 - Data_type – tip podataka za novu vrijednost parametara.
 - Error_desc – detaljni opis greške ukoliko dođe do nje tokom postavljanja vrijednosti parametara.

JNI model sadrži deklaraciju i definiciju nativnih funkcija koje se pozivaju iz TR-069 agentskog servisa i koje pozivaju funkciju iz libCPE biblioteke. Metode implementirane u JNI modelu su:

- Incijalizacija korisnika.
- Deincijalizacija korisnika.
- Pokretanje korisnika.
- Zaustavljanje korisnika.
- Brisanje višestrukog objekta.
- Registrovanje povratnog poziva.
- Odjavljivanje povratnog poziva.

Ove metode su standardne, ne zavise od modela podataka (od parametara čije vrijednosti mijenjaju ili dobavljaju i od višestrukih objekata koji treba da se dodaju ili brišu). Sve nativne funkcije imaju isti početak imena:

*Java_(ime paketa)_(ime klase)_(ime funkcije)(JNIEnv *env, jobject obj, ...)*

ime paketa – paket u kome se nalazi klasa

ime klase – klasa u kojoj se nalazi funkcija

ime funkcije – naziv funkcije u Java dijelu

env – referenca na Java okruženje za izvršavanje nativnih funkcija

obj – Java objekat čija je nativna funkcija članica klase

libcpe_jni_tr106_wrapper.c

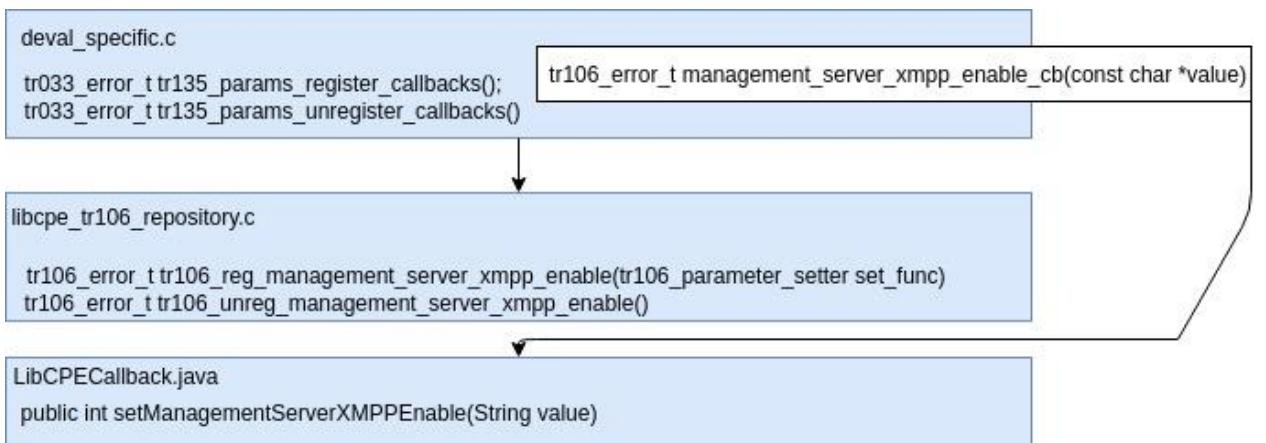
```
int Java_insight_tr069_client_LibCPENative_tr106SetManagementServerXMPPJid(JNIEnv
*env, jobject obj __attribute__((unused)), jint index, jstring xmppjid)
```

- JNI funkcija za postavljanje vrijednosti parametara u kojoj se sa identičnim parametrima poziva nativna funkcija - **tr033_set_management_server_XMPP**.

LibCPENative.java

```
Public native int tr106SetManagementServerXMPPJid(int index, String xmppjid)
```

- Deklaracija JNI funkcije za postavljanje vrijednosti parametra.



Slika 4.3 Dijagram postavljanja vrijednosti parametara

Read/Write – za razliku od read parametara vrši se implementacija četiri funkcije na najnižem nivou, za postavljenje i dobijanje vrijednosti i funkcije za registraciju i poništavanje registracije povratne funkcije o promjeni vrijednosti parametara na strani poslužitelja.

tr106_error_t tr106_reg_management_server_enable(tr106_parameter_setter set_fun)

- registracija povratne funkcije o promjeni vrijednosti parametra na strani poslužitelja

tr106_error_t tr106_unreg_management_server_xmpp_enable()

- deregistracija povratne funkcije o promjeni vrijednosti parametra na strani poslužitelja

tr106_error_t tr106_set_management_server_xmpp_enable(unsigned index, cpe_bool_t xmpp_enable)

- funkcija za mapiranje parametara u stablu repozitorijuma i postavljanje vrijednosti pozivom funkcije ***data_repo_set_param_value***.

tr106_error_t tr106_get_management_server_xmpp_enable(unsigned index, cpe_bool_t *xmpp_enable)

- funkcija za mapiranje parmetara u stablu repozitorijuma i dobavljanje vrijednosti pozivom funkcije ***data_repo_get_param_value***.

Funkcija ***data_repo_get_param_value*** definisana je u ***libcpe_data_repository_engine.c***.

deval_specific.c

tr106_error_t tr106_params_register_callbacks()

tr106_error_t tr_106_params_unregister_callbacks()

- funkcije DEVAL sloja u kom se vrši registracija i deregistracija povratnih funkcija o promjeni vrijednosti parametara pozivom opisnih funkcija iz ***libcpe_tr106_repository.c***

ManagementServer.java

Funkcije o promjeni vrijednosti nekog od XMPP parametara:

- ***public void setXMPPEnable(boolean xmppenable)***
- ***public void setXMPPServerAddress(String xmppserverAddress)***
- ***public void setXMPPServerPort(int xmppserverPort)***
- ***public void setXMPPUsername(String xmppusername)***
- ***public void setXMPPPPassword(String xmpppassword)***
- ***public void setXMPPJid(String xmppjid)***

TR181Monitor.java

- ***public void restartXMPPClient()***

Funkcija za ponovo pokretanje XMPP klijenta. Poziva se na svaku promjenu nekog od XMPP parametara.

XMPPClient.java

XMPPClient klasa implementira **ConnectionListener** kako bi omogućila slušanje stanja veze.

Postavlja status veze, i osim *override* funkcija ima i konstruktor koji preuzima *jid* i lozinku iz **Shared Preferences**.

Metod *connect()* rukuje sa povezivanjem na poslužitelja.

XMPPTCPConnectionConfiguration pomaže da se podaci o vezi umotaju u jedan objekat. On čuva korisničko ime, lozinku, domen poslužitelja i resurs XMPP korisnika.

Zatim se vrši inicijalizacija **XMPPTCPConnection**, koji prolazi u konfiguraciji.

Funkcija **initConnection()** poziva metodu **connect()**.

Unutar **start()** metode se poziva funkcija **initConnection()**.

```
ChatManager chatManager = ChatManager.getInstanceFor(mConnection) ;
```

daje *ChatManager* koji se koristi da se dobije znakovni objekat (eng. *Char*) koji šalje poruku.

ChatManager se nalazi u Smack biblioteci.

ChatManager.getInstanceFor(mConnection).addIncomingListener(new IncomingChatMessageListener()) – funkcija za slušanje pristiglih poruka

5. Ispitivanje i verifikacija

U okviru ovog rada obavljena su ispitivanja realizovanih modula.

I. OPENFIRE

Openfire je poslužitelj za IM i grupni chat koji je baziran na XMPP poslužitelju napisanom u Java programskom jeziku i licenciran sa Apache License 2.0 licencom. Napisan je kao proizvod open-source zajednice programera i običnih korisnika koja se naziva Ignite Raltime [9].

Openfire ima sljedeće funkcionalnosti:

- Web - bazirana sprega za administraciju i konfiguraciju
- Instalacija datoteka
- Posjeduje prilagodljivo okruženje
- Podržava implementaciju SSL/TLS enkripcije
- Prilično jednostavnu instalaciju sa prostim spregama
- Nezavisnu platformu, u potpunosti Java
- Kompletna integracija i kompatibilnost sa Spark XMPP klijentom
- Podržava više od 50000 istovremeno prisutnih korisnika

Pokretanje Openfire i upotreba web pregledača inicira administratorsku konzolu. Podrazumijevani port za web administraciju i konfiguraciju je 9090 za nezaštićenu konekciju, odnosno 9091 za zaštićenu konekciju (SSL/TLS).

Okruženje Openfire prikazano je na slici 5.1.

The screenshot shows the Openfire Admin Console interface. The left sidebar includes links for Server Manager, Server Settings, TLS/SSL Certificates, Media Services, and Pub/Sub. The main content area has tabs for Server Information, System Properties, Language and Time, Clustering, Cache Summary, Database, Logs, Email Settings, SMS Settings, and Security Audit Viewer. The Server Information tab is active, displaying details like Server Uptime (11 minutes), Version (Openfire 4.3.2), and Java Version (1.8.0_191). The Server Ports tab lists various ports (e.g., 5222, 5223, 7070, 7443, 5269, 5270, 5275, 5276, 5262, 5263, 9090, 9091, 7777) with their descriptions. A right-hand panel titled 'Ignite Realtime News' displays recent plugin releases.

Slika 5.1 Okruženje Openfire

II. Ispitivanje vrijednosti parametara

Ispitivanje parametara je vršeno ručno upoređivanjem parametara na uređaju i poslužitelju.

Prilikom testiranja korišten je javni GenieACS poslužitelj otvorenog koda [6]. GenieACS je izabran za kompatibilnost sa standardom i cilnjim poslužiteljem operatora. Okruženje GenieACS-a je prikazano na slici 5.2.

The screenshot shows the GenieACS web interface. The top navigation bar includes Home, Devices, Faults, Presets, Objects, Provisions, Virtual Parameters, and Files. The main content area shows a device configuration for 'Device: D0052A-IPSetTopBox-SCY5Y351NB3I'. It displays 'Device parameters' such as 'xmpp', 'Device.XMPP.Connection.1.Enable true', and 'Device.XMPP.Connection.1.Username admin'. Below this is a 'Task queue' table with columns Task, Time, Fault code, Fault message, Fault detail, and Retries, which is currently empty. At the bottom, there are buttons for Reboot, Factory reset, Print file, Add Firmware, and Delete.

Slika 5.2 GenieACS poslužitelj

Test	Rezultat
Ručno testiranje parametara	✓
Testiranje korištenjem GenieACS poslužitelja	✓

Tabela 5.1 Spisak testova ispitivanja parametara

III. Spark

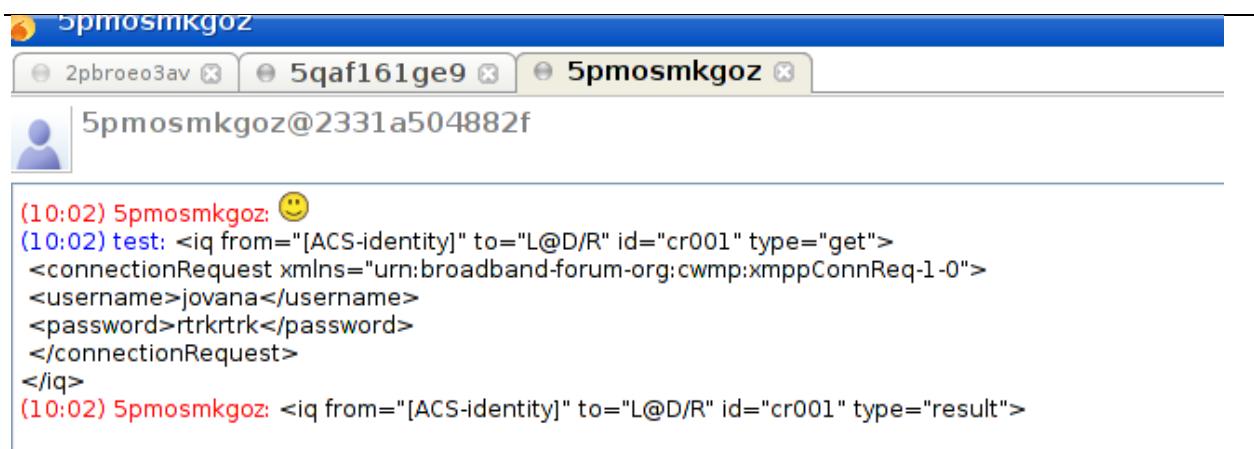
Spark je open-source program za razmjenu poruka (zasnovan na XMPP protokolu) koji omogućava korisnicima da komuniciraju putem teksta u realnom vremenu [8]. Može se integrisati sa Openfire poslužiteljem da bi se obezbijedile dodatne funkcije, kao što je kontrolisanje različitih djelova Spark funkcionalnosti iz centralne upravljačke konzole, ili integracija sa podrškom poslužitelja, omogućujući Spark korisnicima da se prijavljuju u redovima, prihvataju i prosljeđuju zahtjeve za podršku, koriste konzervisane odgovore. Budući da se koriste kao platforma, može se pokrenuti na različitim sistemima.

Korištenjem Spark programa izvršena su testiranja.

Test	Rezultat
Primljena poruka sa ispravnim podacima	✓
Prijmljena poruka sa neispravnim podacima	✓
Primljena poruka neispravnog formata	✓

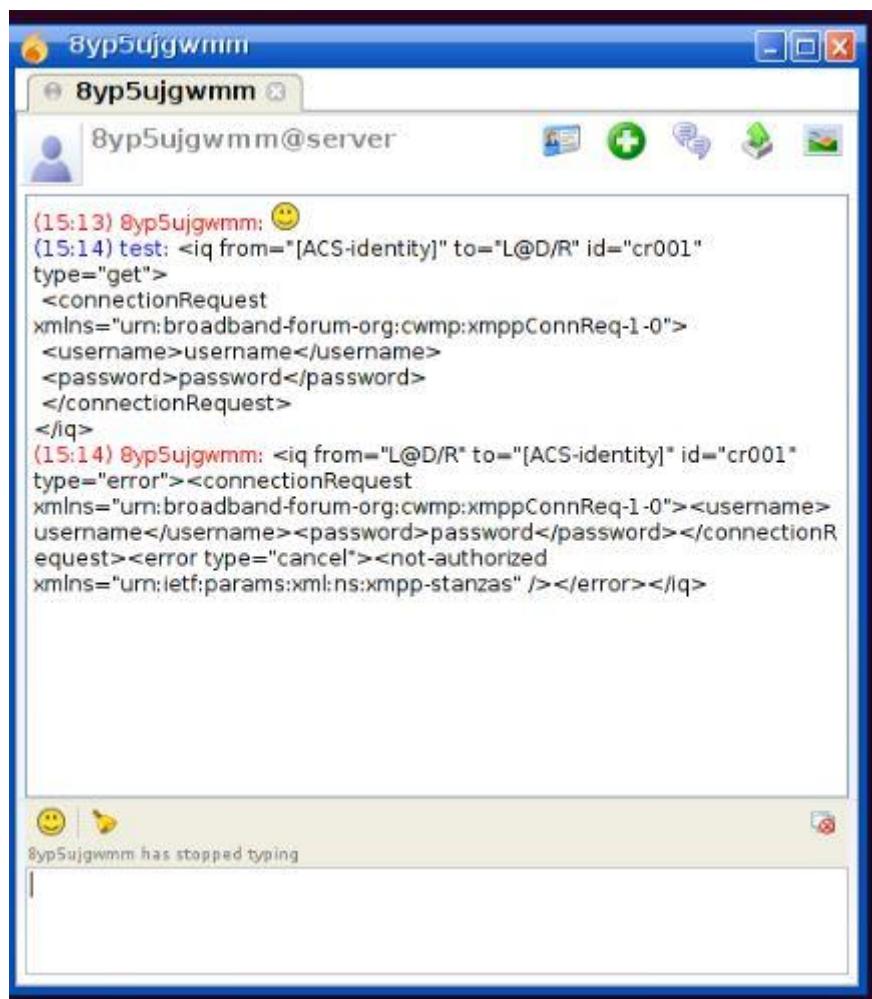
Tabela 5.2 Spisak testova primljenih poruka

Primjeri odgovora na poruke prikazani su na slikama ispod.



Slika 5.3 Odgovor na uspješan zahtjev

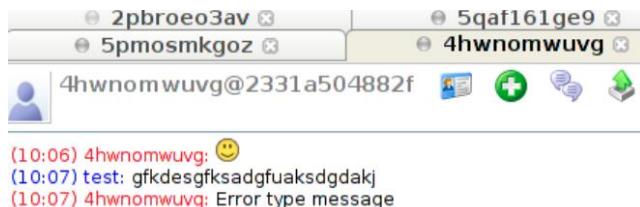
Na slici 5.3 prikazana je poruka za konekciju sa potvrđnim odgovorom. Poruka je odgovarajućeg formata sa ispravno unesenim korisnikom i šifrom.



Slika 5.4 Odgovr na neuspješan zahtjev

Negativan odgovor na zahtjev za konekciju šalje se kada korisnik nepravilno unese ime korisnika ili šifru.

Poruka neodgovarajućeg formata se isporučuje posle pristizanja poruka bez imena korisnika i šifre.



Slika 5.5 Poruka neodgovarajućeg formata

Test	Rezultat
Kreiranje novog korisnika na poslužitelju Openfire	✓
Slanje zahtjeva za komunikaciju sa ACS serverom	✓
Komunikacija sa ACS serverom	✓

Tabela 5.3 Spisak testova za komunikaciju sa ACS-om

Komponente platforme na kojoj je razvijano rješenje 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 5.6 Ploča na kojoj je razvijano rješenje

6.Zaključak

Osnovni zadatak rada bila je realizacija TR-069 klijenta. Rješenje je realizovano upotrebom programskih jezika C i Java. Pri implementaciji korištena je direktna komunikacija sa poslužiteljem Openfire.

Korištenje XMPP protokola i direktna komunikacija sa poslužiteljem omogućilo je uspostavljanje dvosmjerne asinhronne veze između klijenta i poslužitelja. Takva veza je pogodna za postizanje potrebne funkcionalnosti koje su uspješno obavljene. Rješenje ne narušava koncept već postojećeg rješenja klijenta za krajnji uređaj i ACS poslužitelja.

Kao poslužitelj je korišten Spark program. Dalji razvoj bi bila implementacija našeg poslužitelja, što bi dalo još veću mogućnost za poboljšanje postojećeg rješenja.

7. Literatura

- [1] Quality Of Video and Audio for Digital Television Services (Quovadis) project, Final report, European Commision Cordis..
- [2] Professional Android Application Development, Autor: Reto Meier ,ISBN-10: 0470344717, 2008
- [3] Embedded Android: Porting, Extending, and Customizing, Autor: Karim Yaghmour, ISBN-13: 978-1-449-30829-2, 31. Mart 2013
- [4] Meet the Extensible Messaging and Presence Protocol (XMPP),
<https://developer.ibm.com/tutorials/x-xmppintro/>, jun 2019
- [5] TR-069 (CPE Wan Management Protocol), Broadband Forum, <https://cwmp-data-models.broadband-forum.org/>, jun 2019
- [6] An open standard for instant messaging: eXtensible Messaging and Presence Protocol (XMPP), Pin Nie, Helsinki University of Technology, 2006
- [7] GenieACS - Open Source solution for auto-configuration server, <https://genieacs.com/>, jun 2019
- [8] Spark, <https://igniterealtime.org/projects/spark/>, jun 2019
- [9] Openfire 4.3.2 documentation,
<https://www.igniterealtime.org/projects/openfire/documentation.jsp>, jun 2019

-
- [10] TR-106 Data Model Template for TR-069-Enabled Devices, Broadband Forum,
https://www.broadband-forum.org/download/TR-106_Amendment-6.pdf, jun 2019
 - [11] <https://en.wikipedia.org/wiki/XMPP>, jun 2019
 - [12] Mikko Laukkanen. Extensible Messaging and Presence Protocol (XMPP). At University of Helsinki, Department of Computer Science, 2004.
 - [13] Android Smack XMPP Introduction:Building a Simple Client, Blikon,
<https://www.blikoontech.com/tutorials/android-smack-xmpp-introductionbuilding-a-simple-client>, jun 2019
 - [14] Using XMPP for TR-069 Connection Requests, <https://www.qacafe.com/articles/2015-03-26-using-xmpp-for-tr-069-connection-requests/>, jun 2019
 - [15] RFC 6120, Extensible Messaging and Presence Protocol (XMPP): Core,
<https://www.ietf.org/rfc/rfc6120.txt>, jun 2019
 - [16] tr-157-1-8.xml, Component Objects for CWMP, Broadband Forum, <https://cwmp-data-models.broadband-forum.org/tr-157-1-8.xml>, jun 2019
 - [17] RFC 7395, An Extensible Messaging and Presence Protocol (XMPP) Subprotocol for WebSocket, <https://www.ietf.org/rfc/rfc7395.txt>, jun 2019
 - [18] <https://cwmp-data-models.broadband-forum.org/cwmp-xmppConnReq-1-0.xsd>, jun 2019