



УНИВЕРЗИТЕТ У НОВОМ САДУ ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



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

ЗАВРШНИ (BACHELOR) РАД

Кандидат: Норберт Немет
Број индекса: e13019

Тема рада: Реализација комуникационог слоја за конфигурациони послужилац заснован на TR-069 протоколу

Ментор рада: др Иштван Пап

Нови Сад, јун, 2013



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:	Монографска документација		
Тип записа, ТЗ:	Текстуални штампани материјал		
Врста рада, ВР:	Завршни (Bachelor) рад		
Аутор, АУ:	Норберт Немет		
Ментор, МН:	Др Иштван Пап		
Наслов рада, НР:	Реализација комуникационог слоја за конфигурациони послужилац заснован на TR-069 протоколу		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2013		
Издавач, ИЗ:	Ауторски репринт		
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6		
Физички опис рада, ФО: (поглавља/страница/цитата/табела/слика/графика/прилога)	7/37/7/5/20/0/0		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника		
Предметна одредница/Кључне речи, ПО:	Конфигурациони послужилац, комуникациони слој		
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	У овом раду је реализован комуникациони слој за конфигурациони послужилац заснован на TR-069 протоколу намењен управљању мрежом уређаја потрошачке електронике.		
Датум прихваташа теме, ДП:			
Датум одбране, ДО:			
Чланови комисије, КО:	Председник:	др Никола Теслић	
	Члан:	др Милан Ђелица	Потпис ментора
	Члан, ментор:	др Иштван Пап	



KEY WORDS DOCUMENTATION

Accession number, ANO:			
Identification number, INO:			
Document type, DT:	Monographic publication		
Type of record, TR:	Textual printed material		
Contents code, CC:	Bachelor Thesis		
Author, AU:	Norbert Nemet		
Mentor, MN:	Ištván Pap, PhD		
Title, TI:	Implementation of the communication layer for a configuration server based on TR-069 protocol		
Language of text, LT:	Serbian		
Language of abstract, LA:	Serbian		
Country of publication, CP:	Republic of Serbia		
Locality of publication, LP:	Vojvodina		
Publication year, PY:	2013		
Publisher, PB:	Author's reprint		
Publication place, PP:	Novi Sad, Dositeja Obradovica sq. 6		
Physical description, PD: (chapters/pages/ref./tables/pictures/graphs/appendices)	7/37/7/5/20/0/0		
Scientific field, SF:	Electrical Engineering		
Scientific discipline, SD:	Computer Engineering, Engineering of Computer Based Systems		
Subject/Key words, S/KW:			
UC			
Holding data, HD:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia		
Note, N:			
Abstract, AB:	In this paper the communication layer for a TR-069 based auto-configuration server is implemented. The solution is designed for network management of consumer electronics devices.		
Accepted by the Scientific Board on, ASB:			
Defended on, DE:			
Defended Board, DB:	President:	Nikola Teslić, PhD	
	Member:	Milan Bjelica, PhD	Menthor's sign
	Member, Mentor:	Ištván Pap, PhD	

Zahvalnost

Zahvaljujem se Milanu Bjelici, Dejanu Stefanoviću i Ištvanu Papu na stručnoj pomoći, savetima i utrošenom vremenu.

Posebno se zahvaljujem rukovodstvu Instituta RT-RK na ukazanoj prilici da se bolje upoznam sa načinom rada u inženjerskom okruženju i budem uključen u proces razvoja novih programskih rešenja.

Zahvaljujem se roditeljima i svima onima koji su me podržali kako mentalno tako i finansijski.

Na kraju se zahvaljujem svima onima koji su na bilo koji način doprineli izradi ovog završnog rada.



УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



SADRŽAJ

1.	Uvod	1
2.	Teorijske osnove.....	3
3.	Koncept rešenja	6
3.1	ACSServlet	9
3.2	TR069SOAPParse.....	9
3.3	TR069MessageHandler.....	11
3.4	TR069CommunicationAPI	14
4.	Programsko rešenje.....	16
4.1	Funkcionalni moduli komunikacionog sloja.....	17
4.2	Klase koje predstavljaju podatke.	23
5.	Rezultati.....	24
6.	Zaključak	28
7.	Literatura	29

SPISAK SLIKA

Slika 1. Hjjerarhija protokola.....	3
Slika 2. Arhitektura visokog nivoa sistema baziranom na TR-069 protokolu.....	4
Slika 3. Arhitektura autokonfiguracionog poslužioca.	6
Slika 4. Arhitektura visokog nivoa i tok razmene poruka	7
Slika 5. Veze između modula komunikacionog sloja.....	8
Slika 6. Format XML podataka.	9
Slika 7: Struktura SOAP poruke.	10
Slika 8. SOAP poruke sa HTTP standardom.	10
Slika 9. Tok obrade u JAXB komponenti.	11
Slika 10. Dijagram prelaza stanja jedne sesije.....	13
Slika 11. Osnovna sesija.	14
Slika 12. Sesija sa greškom.....	14
Slika 13. Sesija za ažuriranje programske podrške na CPE uređaju.	15
Slika 14. Jedna sesija između konfiguracionog poslužioca i krajnjeg uređaja.	24
Slika 15. Inform poruka u SoapUI alatu.	25
Slika 16. GetParameterValues poruka u SoapUI alatu.	25
Slika 17. Prazna poruka u SoapUI alatu.	26
Slika 18. Merenje vremena odziva u lokalnoj mreži.	26
Slika 19. Merenje vremena odziva preko interneta	27
Slika 20. Grafička korisnička sprega konfiguracionog poslužioca.....	27

SPISAK TABELA

Tabela 1. Poruke u TR-069 standardu.	5
Tabela 2. Funkcionalni moduli u komunikacionom sloju.	16
Tabela 3. Klase za čuvanje podataka.	17
Tabela 4. Lista metoda za obradu TR-069 poruka.....	19
Tabela 5. Spisak metoda za generisanje poruka.	22

SKRAĆENICE

ACS	- <i>Auto Configuration Server</i> - Poslužilac za automatsku konfiguraciju
CPE	- <i>Customer Premises Equipment</i> - Krajnji korisnički uređaj
STB	- <i>Set Top Box</i> - Set-top-boks uređaj
EJB	- <i>Enterprise JavaBeans</i> - Komponente Java standarda
FIFO	- <i>First In First Out</i> - Tip memorijске strukture
XML	- <i>Extensible Markup Language</i> – Proširiv opisni jezik
XSD	- <i>XML Schema Definition</i> – Definicija XML formata
SOAP	- <i>Simple Object Access Protocol</i> – Protokol za pristup objektima
JAXB	- <i>Java Architecture for XML Binding</i> – Standard za povezivanje Java objekata sa XML podacima

1. Uvod

Uredaji potrošačke elektronike doživljavaju ekspanziju, kako kvantitativnu u vidu broja korisnika, tako i kvalitativnu. Kvalitativna ekspanzija se ogleda u sve boljim performansama uređaja, uslugama koje njihova programska podrška može pružiti krajnjim korisnicima. U takvim uslovima, pružaoci usluga (eng. Provider) se susreću sa zahtevima ispunjavanja svojih obaveza prema krajnjim korisnicima. Takvi zahtevi nameću rešenje praćenja i upravljanja mrežom uređaja preko konfiguracionih poslužioca. Konfiguracioni poslužioci (eng. ACS, Auto-Configuration Server) su gradivni element protokola za daljinsko upravljanje uređajima. Zadatak konfiguracionih poslužioca je da obezbedi operacije i napredne funkcije na krajnjem uređaju sa ili bez znanja njegovog vlasnika. Mogu obavljati određene akcije automatski, na zadati vremenski period, u određenom slučaju, ili mogu ih odraditi na komandu operatera koji je u većini slučajeva u operatorskom centru pružaoca usluga. Konfiguracioni poslužioci zahtevaju puno resursa za svoj rad, kako procesorske moći i memorijskih kapaciteta, tako i komunikacionih resursa. Moraju biti u mogućnosti da simultano obrađuju komunikaciju i usluže veliki broj krajnjih uređaja. Njihovim korišćenjem pružaoci usluga smanjuju troškove i automatizuju proces korisničke podrške. Problematika razvoja konfiguracionih poslužioca je usko vezana za razvoj komunikacionih protokola koji ovakve specifične zahteve omogućuju. Na ovim zahtevima je i nastao TR-069 protokol o kome će kasnije biti reči. Poslužioci moraju omogućiti skalabilnu i distribuiranu obradu nezavisnu od tipa krajnjih uređaja, koji se po svojim funkcionalnostima mogu razlikovati.

U okviru ovog rada realizovan je komunikacioni sloj poslužioca za automatsku konfiguraciju (Auto Configuration Server – ACS) zasnovanog na TR-069 protokolu [1], namenjenog za nadzor i konfiguraciju uređaja potrošačke elektronike. Programsko rešenje je realizovano modularno radi što efikasnijeg iskorišćenja resursa i što lakšeg održavanja u

eksploataciji. Komunikacioni sloj je zadužen za direktnu komunikaciju sa krajnjim uređajima, isporuku poruka, njihov prijem i prosleđivanje na obradu.

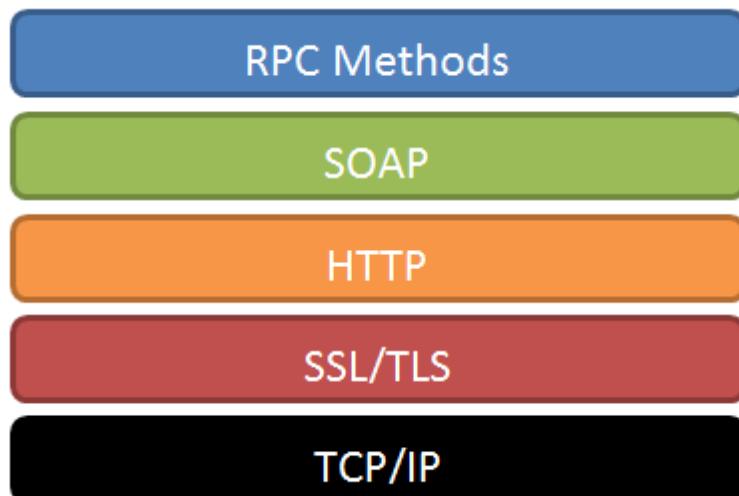
U radu je prikazana realizacija komunikacionog sloja kao prilagodnog sloja za kompletan poslužilac (eng. server). Komunikacioni sloj omogućava rad sa proizvoljnim podacima korišćenjem koncepta iz TR-069 komunikacionog protokola. Rešenje je realizovano u programskom jeziku Java.

Rad se sastoji od sledećih celina:

1. Teorijske osnove - pokrivaju osnove komunikacionog protokola.
2. Koncept rešenja - objašnjavaju veze među modulima i njihovu namenu.
3. Programsко rešenje - opis svih metoda u rešenju sa ulaznim parametrima i povratnim vrednostima.
4. Rezultati - prikazani su rezultati testiranja.
5. Zaključak - pokriva ispunjenost zadatka i budući rad.

2. Teorijske osnove

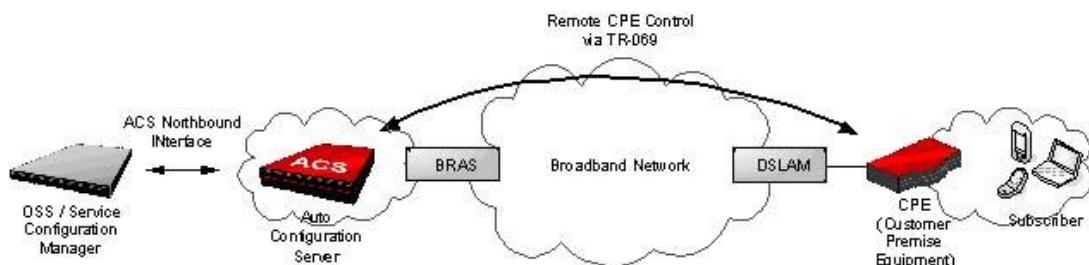
TR-069 je tehnička specifikacija protokola koji omogućuje daljinsko upravljanje krajnjih korisničkih uređaja i konfiguraciju krajnjih korisničkih uređaja potrošačke elektronike. Kao bidirekcioni protokol, omogućava komunikaciju između entiteta u standardu. U standardu su propisani CPE (eng. Customer Premises Equipment) i ACS (eng. Auto-Configuration Server) uređaji. Prvi su krajnji uređaji kojima se pristupa preko centralne tačke – ACS servera. Za transport poruka između tačaka koriste se standardizovani načini i standardi. Protokoli po nivoima su prikazani na Slici 1.



Slika 1. Hijerarhija protokola.

Standard je zbog svoje fleksibilnosti postao najzastupljeniji standard ovog tipa. TR-069, poznat i kao CWMP (eng. CPE Wan Management Protocol), omogućuje prilagođavanje komunikacije prirodi krajnjih uređaja. Preko koncepta modela podataka moguće je prilagoditi komunikaciju i upravljanje parametrima mreže i krajnjih uređaja. Upravo ovaj koncept je doprineo definisanju standardizovanih modela, od kojih je jedan specifičan za set-top-boks

uređaje (TR-135). Ovakva koncepcija standarda je dovela do toga da nadležne institucije za razvijanje standarda u oblastima digitalne televizije i mrežnih tehnologija predlože TR-069 kao primarno rešenje za upravljanje i nadgledanje krajnjih uređaja. Arhitektura visokog nivoa sistema zasnovanog na TR-069 protokolu je prikazana na Slici 2.



Slika 2. Arhitektura visokog nivoa sistema baziranom na TR-069 protokolu.

Komunikacija izmedju servera i krajnjih uređaja se odvija u sesijama (eng. session). Sadržaj poruka, njihova sintaksa i semantika su prethodno definisane. U komunikaciji postoje poruke koje su obavezne i one koje su mandatorne (neobavezne). Načelni pregled poruka u standardu je prikazan u Tabeli 1.

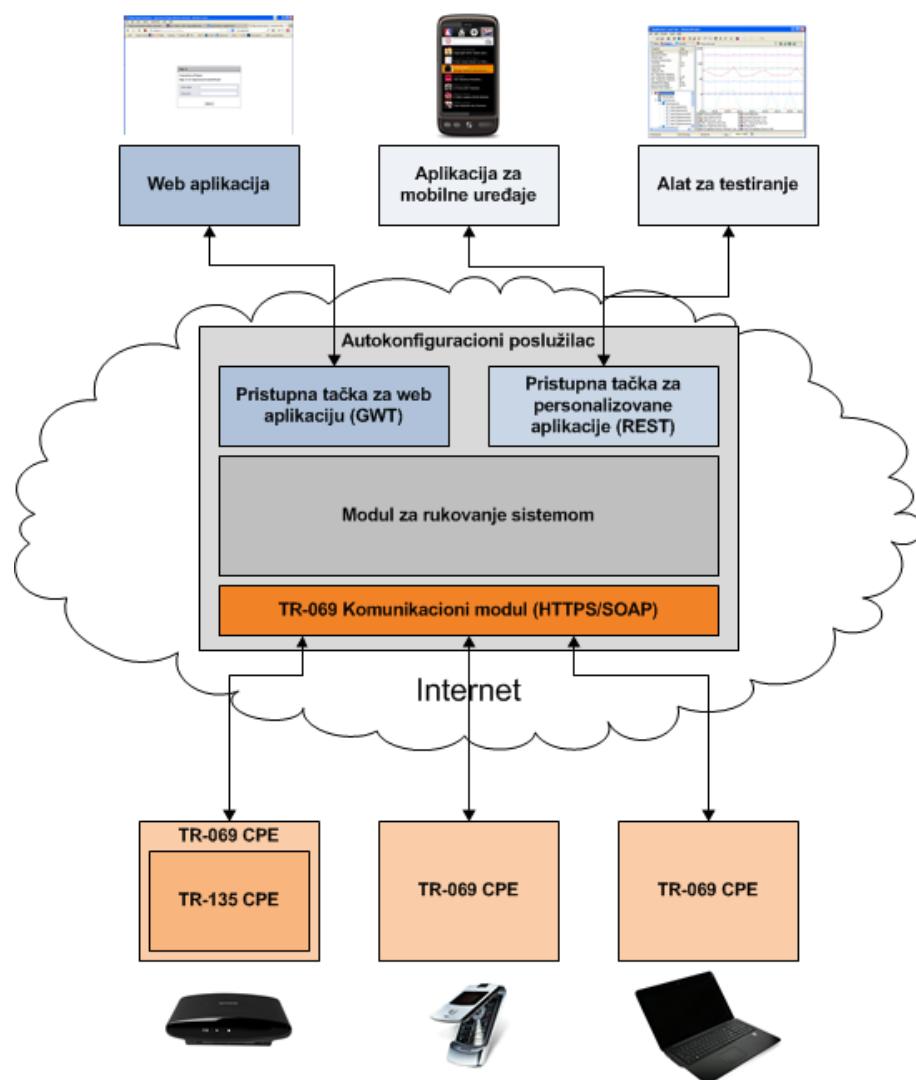
Sigurnost u komunikaciji i konzistentnost razmene poruka obezbeđena je korišćenjem HTTPS komunikacionog protokola na ISO OSI nivou sesije. Kriptovanje komunikacije nije obavezno jer je dopušteno i korišćenje HTTP standarda, međutim u ozbiljnijim aplikacijama je neophodno obezbediti autentikaciju klijenta naročito ako takve aplikacije opslužuju na hiljade krajnjih uređaja. Operacije visokog nivoa koje TR-069 svojim korisnicima dopušta su npr. inicijalna konfiguracija uređaja, praćenje statusa uređaja, postavljanje nove programske podrške na krajnjim uređajima, praćenje vrednosti parametra, postavljanje vrednosti nekih parametara itd. Kao bitna stavka svakog modela podataka (eng. Data Model) je i definisanje ne samo relevantnih podataka već i njihovih osobina. Svaki parametar može biti samo za čitanje, samo za upisivanje ili oba. Na ovaj način se poštuje specifičnost krajnjih uređaja.

CPE metode	Odgovor	Poziv
GetRPCMethods	Obavezан	Opcionalan
SetParameterValues	Obavezан	Obavezан
GetParameterValues	Obavezан	Obavezан
GetParameterNames	Obavezан	Obavezан
SetParameterAttributes	Obavezан	Opcionalan
GetParameterAttributes	Obavezан	Opcionalan
AddObject	Obavezан	Opcionalan
DeleteObject	Obavezан	Opcionalan
Reboot	Obavezан	Opcionalan
Download	Obavezан	Obavezан
Upload	Opcionalan	Opcionalan
FactoryReset	Opcionalan	Opcionalan
GetQueuedTransfers	Opcionalan	Opcionalan
ScheduleInform	Opcionalan	Opcionalan
SetVouchers	Opcionalan	Opcionalan
GetOptions	Opcionalan	Opcionalan
ACS Metode	Poziv	Odgovor
GetRPCMethods	Opcionalan	Obavezan
Inform	Obavezan	Obavezan
TransferComplete	Obavezan	Obavezan
AutonomousTransferComplete	Opcionalan	Obavezan
RequestDownload	Opcionalan	Opcionalan

Tabela 1. Poruke u TR-069 standardu.

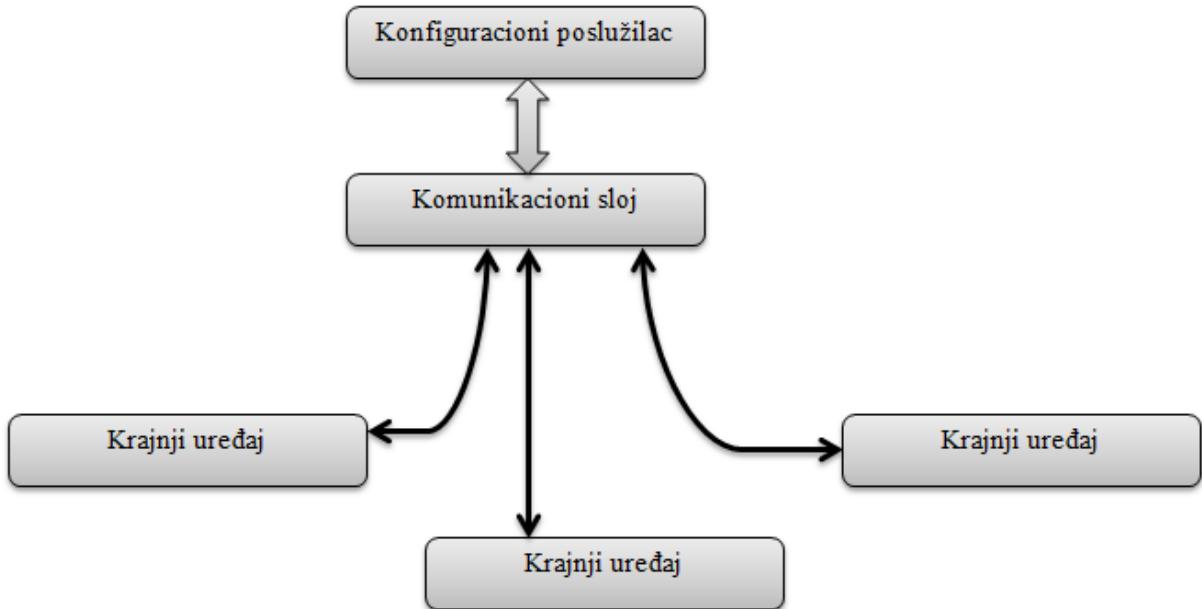
3. Koncept rešenja

Komunikacioni sloj ovog rešenja predstavlja prvi i takozvani spoljni deo kompletног konfiguracionog poslužiоca kao što je prikazano na Slici 3.



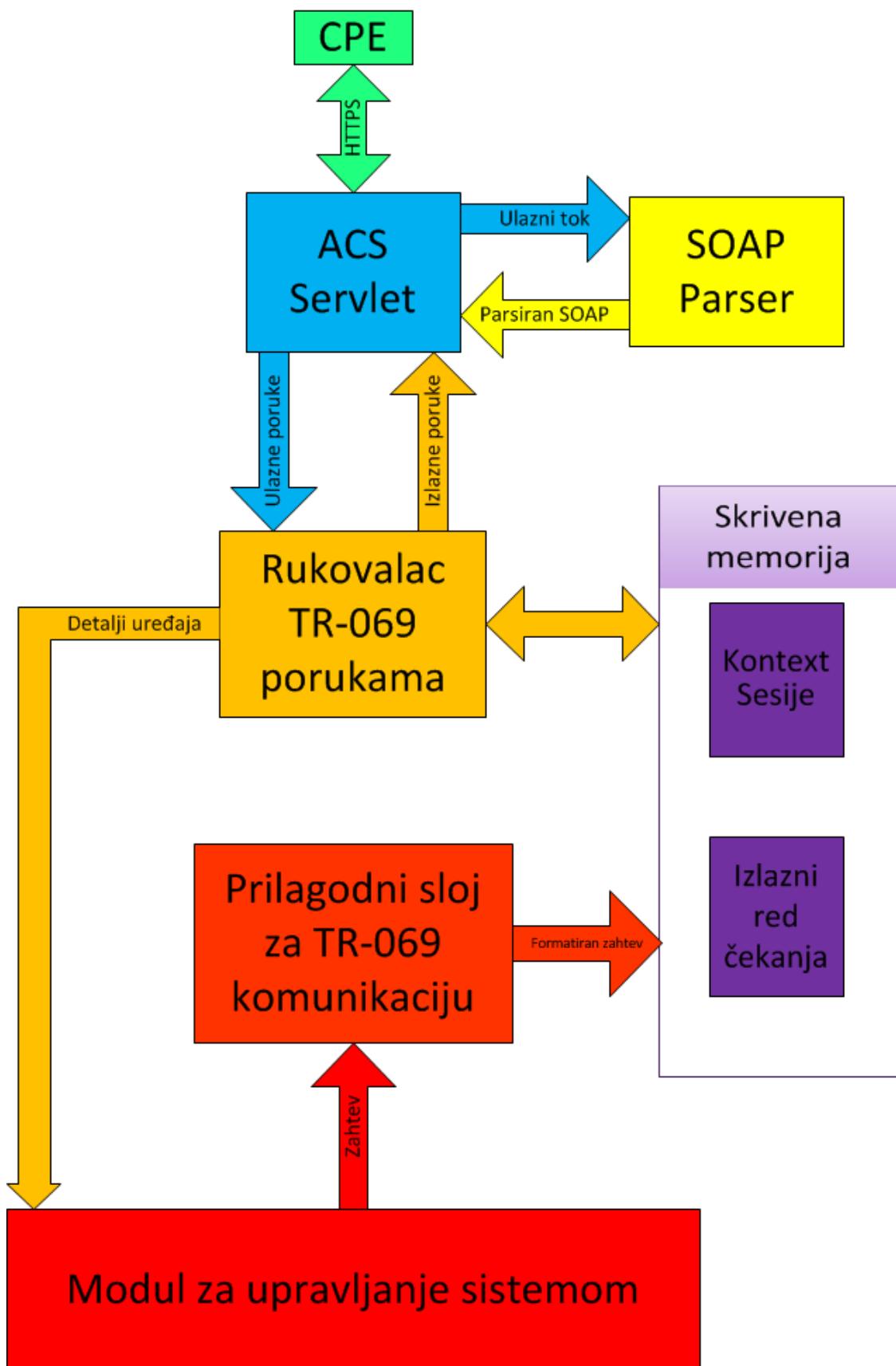
Slika 3. Arhitektura autokonfiguracionog poslužiоca.

Cilj ovakvog komunikacionog sloja je da omogući brzu razmenu informacija između unutrašnjih modula poslužioca i krajnjih CPE uređaja, zadržavajući pri tom fleksibilnost i modularnost [5]. Arhitektura visokog nivoa i tok razmene poruka je prikazan na Slici 4.



Slika 4. Arhitektura visokog nivoa i tok razmene poruka

Komunikacioni sloj je osmišljen kao modularan i fleksibilan segment rešenja. Takav koncept omogućava adaptibilnost i lako održavanje programskog rešenja. Komunikacioni moduli koriste standardizovane načine obrade i komunikacije. Moduli od kojih je komunikacioni sloj sastavljen su prikazani na Slici 5.



Slika 5. Veze između modula komunikacionog sloja.

3.1 ACSServlet

ACSServlet je pristupna tačka za CPE uređaje. Ovaj modul iz HTTP zahteva izdvoji ulazni tok podataka i prosledi *TR069SOAPParse*r modulu koji pretvori tok podataka u Java objekat. Tako dobijen objekat *ACSServlet* prosleđuje *TR069MessageHandler* modulu koji vrši dalju obradu. *ACSServlet* je ulazna tačka celog sistema. Koncipiran je kao Java Servlet. Java Servlet je tehnologija proistekla iz Java standarda koja se koristi za web bazirane projekte. Servleti služe za automatski odgovor na određene zahteve a po definiciji predstavljaju proširenje mogućnosti servera. Po svojim osobinama, servleti omogućavaju procesiranje i skladištenje podataka dobijenih u HTML formi, pružanje dinamičkih sadržaja iz baza podataka ali i informacije o stanjima koja ne postoje u HTTP protokolu.

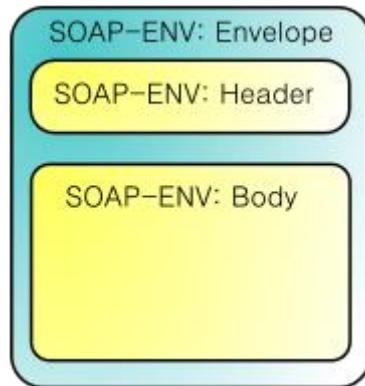
3.2 TR069SOAPParse

Modul prihvata ulazni tok podataka i u njemu sadržan SOAP (eng. Simple Object Access Protocol) XML (eng. Extensible Markup Language) objekat pretvori u Java objekat. SOAP je W3C (eng. World Wide Web Consortium) standard koji obezbeđuje osnovu za razvoj web servisa i drugih protokola. Jedna SOAP poruka se sastoji od poglavlja (eng. Header) i tela (eng. Body) koji su enkapsulirani u SOAP koverat (eng. Envelope) kao što se vidi na Slici 7. Poglavlje SOAP poruke služi za identifikaciju a telo sadrži sve ostale podatke. SOAP standard se najčešće oslanja na XML standard za format podatak, čiji je format prikazan na Slici 6 [4] , dok za prenos poruka i pregovoranje oko ispostave veze koristi HTTP ili SMTP (eng. Simple Mail Transfer Protocol) protokol kako je prikazano na Slici 8.

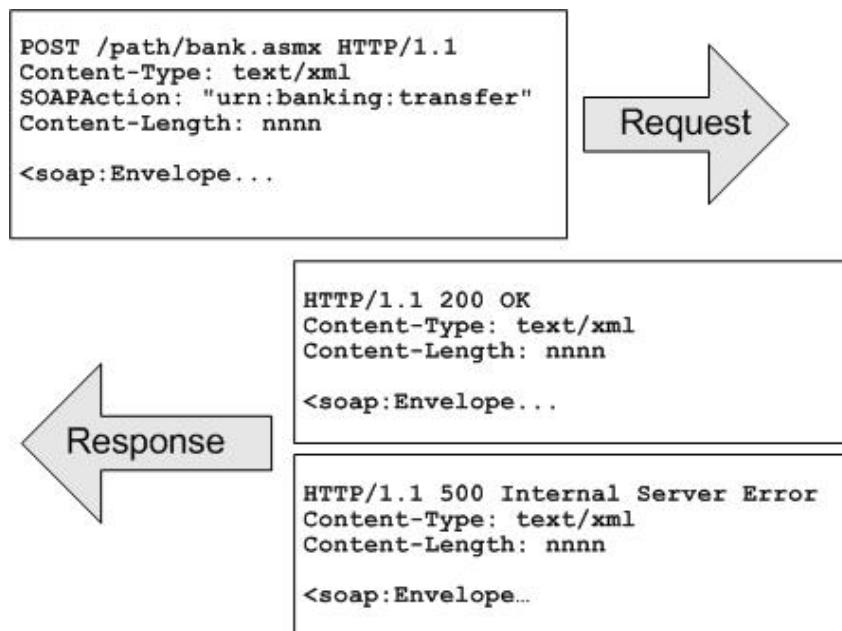
```
<?xml version="1.0" encoding="utf-8" ?>
<cricketers>
    <category type="Bowlers">
        <cricketer>Wasim Akram</cricketer>
        <cricketer>Michael Holding</cricketer>
        <cricketer>Shane Warne</cricketer>
        <cricketer>Muthaiah Muralidaran</cricketer>
    </category>
    <category type="Batsmen">
        <cricketer>Geoff Boycott</cricketer>
        <cricketer>Sunny Gavaskar</cricketer>
        <cricketer>Sachin Tendulkar</cricketer>
        <cricketer>Vivian Richards</cricketer>
    </category>
    <category type="Allrounders">
        <cricketer>Kapil Dev</cricketer>
        <cricketer>Ian Botham</cricketer>
    </category>
    <category type="WicketKeeper">
        <cricketer>Adam Gilchrist</cricketer>
    </category>
</cricketers>
```

Slika 6. Format XML podataka.

Najčešće ove poruke su naslonjene na aplikativne slojeve programskih podrški koje ih koriste. SOAP je pogodan format i za dodatnu bezbednost u komunikaciji, jer se može zaštititi dodatnim kodovanjem informacija, na primer, WS-S (eng. Web Service-Security) formatom.



Slika 7: Struktura SOAP poruke.

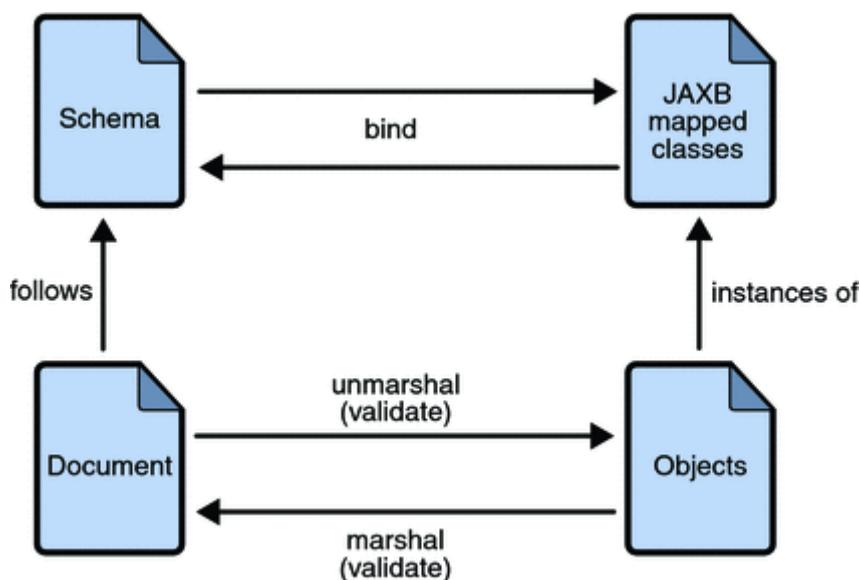


Slika 8. SOAP poruke sa HTTP standardom.

Pretvaranje se vrši pomoću JAXB (eng. Java Architecture for XML Binding) biblioteke koji je sastavni deo Java razvojnih alata. Ulazni tok podataka se enkapsulira u XML čitaču ulaznog toka i tako se zajedno sa odredištem prosleđuje *Unmarshaller* modulu. Da bi *Unmarshaller* znao da pretvori ulazni XML objekat u Java objekat, odredište mora da ima odgovarajuće osobine. Ovo se može ostvariti dodavanjem Java anotacija ručno ili generisanjem odredišnih klasa iz XSD (eng. XML Schema Descriptor) datoteke koja opisuje strukturu ulaznog XML objekta. JAXB je namensko rešenje za web bazirane aplikacije jer omogućava direktno mapiranje Java klasa na XML objekte. Pruža dve osnovne usluge:

1. Transformaciju Java objekata u XML objekte,
2. Transformaciju XML objekata u Java objekte.

Korišćenje JAXB generatora je korisno kod kompleksne programske podrške koja se često menja. U takvim slučajevima, redovna promena takozvane XML šeme podataka omogućava uredno sinhronizovanje sa Java definicijama tih objekata. JAXB je integriran u Java SE (eng. Java Platform, Standard Edition) platformu i kao takav je deo Java EE standarda (eng. Java Platform, Enterprise Edition). Šema toka obrade u JAXB komponenti je prikazana na Slici 9.



Slika 9. Tok obrade u JAXB komponenti.

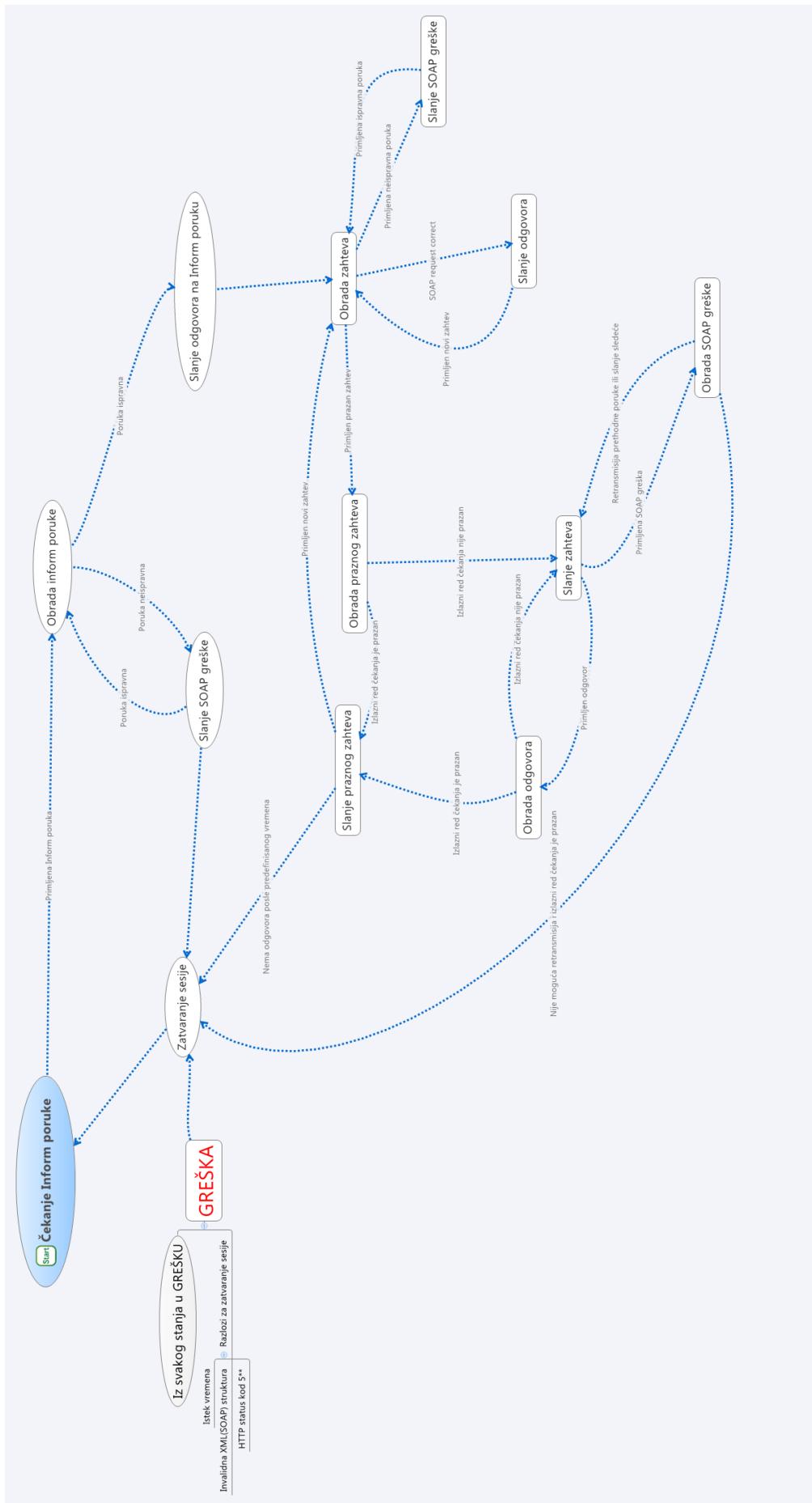
3.3 TR069MessageHandler

Ova komponenta ima za zadatku obradu poruka koje pristižu od *ACSServleta* kao što se može videti sa Slike 5. Komunikacija sa *ACSServlet* modulom je dvosmerna, što podrazumeva da *TR069MessageHandler* modul odgovara na posleđene poruke. Tako primljene poruke *TR069MessageHandler* obrađuje mehanizmom koji izdvoji podatke i pogodnoj formi ih prosledi sloju za upravljanje poslužiocem (eng. Management). Jedan od bitnih segmenata ovog koncepta je njegova mogućnost da koristi sistem događaja (eng. Event). U kompleksnim programskim podrškama ovakvi sistemi se koriste kako bi se određeni događaji objavili celom sistemu. U ovom slučaju oni su zaduženi da obaveste o specifičnim događajima koji nastaju iz sadržaja poruke.

Sesija (eng. Session) je nedeljiva celina komunikacije između krajnjih uređaja i poslužioca. U okviru jedne sesije se razmenjuju jedna ili više poruka propisanih standardom. Svaka sesija se osim svojom identifikacijom, učesnicima odlikuje i stanjem. Dijagram prelaza

stanja je prikazan na Slici 10. *TR069MessageHandler* modul je zadužen za ažuriranje stanja sesije.

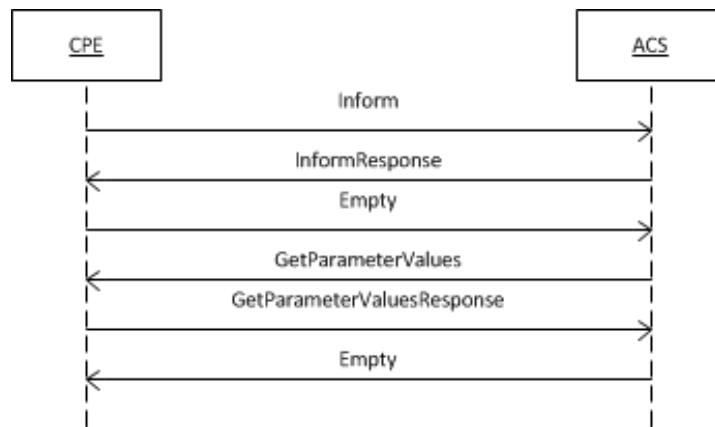
Izlazne poruke su one poruke koje se šalju krajnjim uređajima. *TR069MessageHandler* te poruke formira čitanjem iz skrivene memorije formirane pomoću koncepta objektnog mapiranja (ORM, eng. Object-Related Mapping).



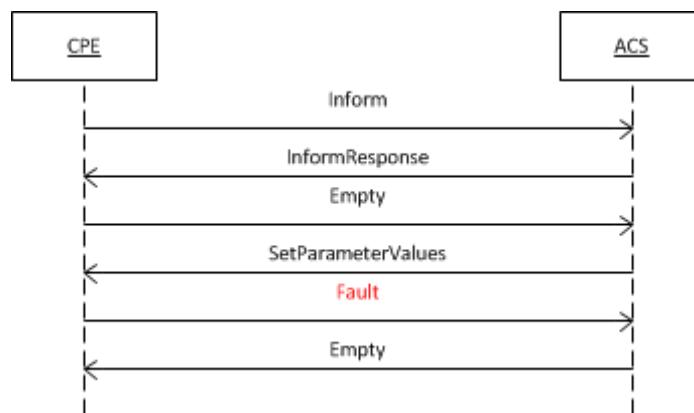
Slika 10. Dijagram prelaza stanja jedne sesije.

3.4 TR069CommunicationAPI

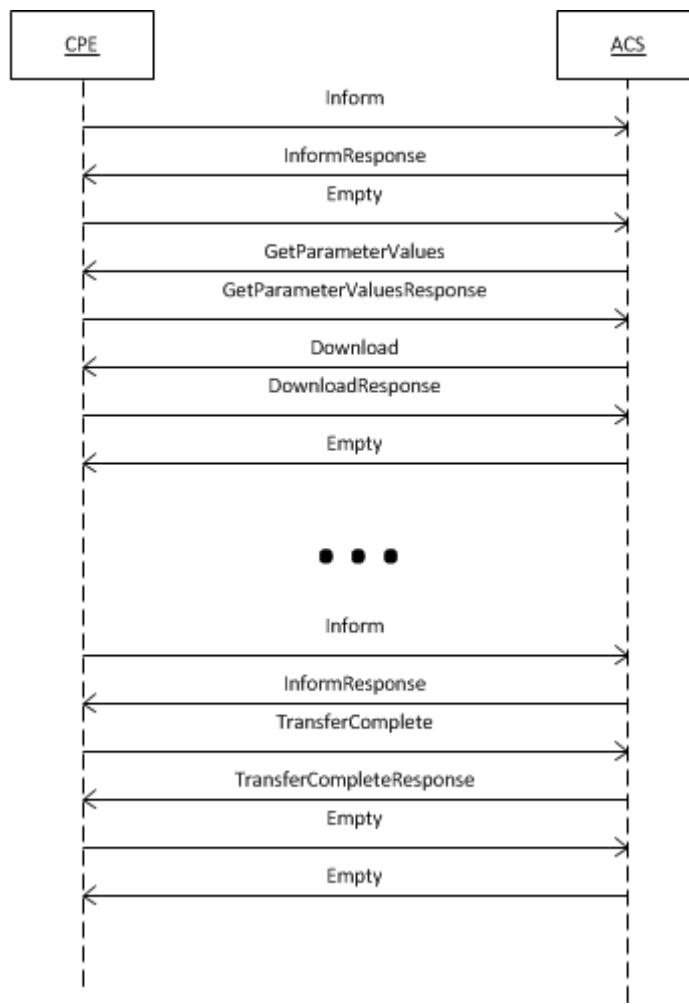
U ovom modulu je realizovan prilagodni sloj između sloja za upravljanje poslužiocem i direktnе komunikacije sa krajnjim uređajima. Ovaj modul može da obavi sledeće funkcionalnosti primanje zahteva od sloja za upravljanje, formiranje zahteva u formi SOAP poruke koja poštaje TR069 standard kao i upis poruke u FIFO (eng. First In First Out) strukturu odgovarajućeg CPE uređaja. Ovaj sloj služi kao uslužni sloj za modulima za upravljanje poslužiocem. Primeri sesija koje ovaj modul u komunikaciji sa CPE uređajima obezbeđuje su prikazani na slikama ispod. Redosled poruka u sesijama, njihovu formu i funkciju definiše standard TR-069. Pored osnovne sesije prikazane na Slici 11, na Slici 12 je prikazana sesija u kojoj se pojavljuje greška u komunikaciji. Na Slici 13 je prikazana sesija za ažuriranje programske podrške na CPE uređajima.



Slika 11. Osnovna sesija.



Slika 12. Sesija sa greškom.



Slika 13. Sesija za ažuriranje programske podrške na CPE uređaju.

4. Programsко rešenje

U ovom poglavlju je objašnjeno programsko rešenje modula za komunikaciju sa krajnjim uređajima zasnovano na TR-069 protokolu. Programsko rešenje je realizovano korišćenjem Enterprise Java Beans (EJB) mehanizma [2] definisanog u okviru Java EE standarda [3] koji omogućava kreiranje skalabilnih i pouzdanih programskih rešenja sa velikim kapacitetom obrade. Kratak opis modula TR-069 komunikacionog sloja u okviru ACS servera je dat u Tabeli 2 a detaljnije će biti opisani u daljem tekstu.

Ime	Opis	Tip
ACSServlet	Pristupna tačka za CPE uređaje.	HTTP Servlet
TR069SOAPPParser	Vrši pretvaranje ulaznog toka podataka u Java objekte.	Java Class
TR069MessageHandler	Interpretira pristigle poruke, ažurira stanje sistema i vraća odgovor.	Singleton Session Bean
TR069CommunicationAPI	Prilagodni sloj prema menedžment modulu servera.	Stateless Session Bean

Tabela 2. Funkcionalni moduli u komunikacionom sloju.

Pored funkcionalnih modula koje vrše obradu podataka postoje i klase koje predstavljaju podatke. Ove podatke o trenutnim sesijama i izlaznim redovima čekanja se čuvaju u brzoj skrivenoj memoriji koji je deo izvršnog okruženja (u ovom slučaju Jboss aplikacioni server verzije 7.1). Kratak opis je dat u Tabeli 3.

Ime	Opis
TR069SessionContext	Čuva sve bitne informacije o jednoj sesiji.
TR069Request	Sadrži jedan zahtev za CPE uređaj u redu čekanja.
TR069Event	Predstavlja događaj koji se desi u rukovaocu porukama. Na ovaj način se sloj za rukovanje poslužiocem o bitnim događajima u komunikacionom sloju i u CPE uređajima.

Tabela 3. Klase za čuvanje podataka.

4.1 Funkcionalni moduli komunikacionog sloja

4.1.1 ACSServlet

4.1.1.1 Metode

- **doPost(HttpServletRequest request, HttpServletResponse response)**

Ova metoda je nasleđena od *HTTPServlet* superklase. Njena funkcija je da ulazni tok podataka prosledi modulu za parsiranje i na osnovu dobijenih podataka pozove odgovarajuću metodu modula za rukovanje porukama.

Ulagani parametri: **request** – HTTP zahtev od CPE uređaja, **response** – povratni kanal prema CPE uređaju.

Povratna vrednost: Nema.

- **getSessionID(HttpServletRequest request)**

Metoda pretražuje kolačiće (eng. cookies) u HTTP zahtevu i izdvaja identifikator sesije.

Ulagani parametri: **request** – HTTP zahtev od CPE uređaja

Povratna vrednost: identifikator sesije.

4.1.2 TR069SOAPParser

4.1.2.1 Polja

- **SOAPEnvelope**

Polje koje sadrži generisan SOAP koverat.

- **operationName**

Iz generisanog SOAP Envelope objekta se isčita ime TR-069 poruke i smešta se u ovo polje. Isčitavanje bi mogao i ACSServlet da izvrši, ali radi što efikasnijeg rada sistema ova operacija je postala deo parsiranja ulaznog toka podataka.

4.1.2.2 Metode

- **parseInputStream(InputStream inputStream)**

Metoda služi za pretvaranje XML-a iz ulaznog toka podataka u SOAP Envelope objekat.

Ulagni parametri: **inputStream** – ulazni tok podataka iz HTTP zahteva.

Povratna vrednost: Nema.

- **writeSOAPEnvelopeToStream(Envelope env, OutputStream os)**

Pomoću ove metode se izlazna SOAP poruka pretvara u XML format i upisuje se u izlazni tok podataka HTTP odgovora.

Ulagni parametri: **env** – izlazna SOAP poruka, **os** – izlazni tok podataka.

Povratna vrednost: Nema.

4.1.3 TR069MessageHandler

4.1.3.1 Polja

- **events**

Instanca klase javax.enterprise.event.Event koja omogućava priljavljivanje događaja u sistemu. Za ove notifikacije može svako da se pretplati u sistemu, i da reaguje na odgovarajući način.

- **cpeDB**

Predstavlja vezu sa bazom podataka pomoću koje je moguće upisati veću količinu podataka direktno u bazu podataka. Ova mogućnost je podtrebna zato što slanje velike količine podataka preko prethodno navedenih notifikacija znatno usporava rad sistema.

- **cache**

Sprega ka Infinispan skrivenoj memoriji koja omogućava brz pristup sladištenim podacima. Modul za rukovajne porukama u ovoj memoriji čuva informacije o trenutno aktivnim sesijama kao i izlazne redove čekanja za pojedinačne CPE uređaje.

4.1.3.2 Metode za rukovanje porukama

Sve metode za rukovanje porukama imaju istu strukturu u vidu pojedinačnih koraka u obradi. Svaka obradjuje jedan tip TR-069 poruke a razlikuju se samo u notifikacijama koje emituju i u skladištenju podataka. Koraci kod obrade jedne TR-069 poruke su:

- Provera validnosti poruke

- Interpretiranje poruke
- Obaveštavanje višeg sloja o promenama i događajima
- Formiranje odgovora

Lista sa imenima metoda koje obradjuju istoimene TR-069 poruke je data u Tabeli 4.

public Envelope inform (String sessionID, Envelope envelope)
public Envelope empty (String sessionID)
public Envelope getParameterValuesResponse (String sessionID, Envelope envelope)
public Envelope setParameterValuesResponse (String sessionID, Envelope envelope)
public Envelope downloadResponse (String sessionID, Envelope envelope)
public Envelope transferComplete (String sessionID, Envelope envelope)
public Envelope fault (String sessionID, Envelope envelope)
public Envelope getRPCMethodsResponse (String sessionID, Envelope envelope)
public Envelope getParameterNamesResponse (String sessionID, Envelope envelope)
public Envelope getParameterAttributesResponse (String sessionID, Envelope envelope)
public Envelope setParameterAttributesResponse (String sessionID, Envelope envelope)
public Envelope addObjectResponse (String sessionID, Envelope envelope)
public Envelope deleteObjectResponse (String sessionID, Envelope envelope)
public Envelope rebootResponse (String sessionID, Envelope envelope)
public Envelope uploadResponse (String sessionID, Envelope envelope)
public Envelope scheduleInformResponse (String sessionID, Envelope envelope)
public Envelope factoryResetResponse (String sessionID, Envelope envelope)

Tabela 4. Lista metoda za obradu TR-069 poruka.

Ulagani parametri u svaku metodu su SOAP koverat sa TR-069 porukom i identifikator sesije.

4.1.3.3 Pomoćne metode

Ove metode generalizuju zajedničke operacije prethodno navedenih metoda za obradu TR-069 poruka.

- **updateState(String sessionID, State newState)**

Ažurira stanje zadate sesije.

Ulagni parametri: **sessionID** – identifikator sesije, **newState** – novo stanje sesije.

Povratna vrednost: Nema.

- **getSessionContext(String sessionID)**

Ova metoda služi za dobavljanje konteksta TR-069 sesije iz brze skrivene memorije. Na osnovu zadatog identifikatora sesije prvo proverava da li postoji sesija u memoriji, ako ima onda vraća referencu natraženi objekat.

Ulagni parametri: **sessionID** – identifikator sesije.

Povratna vrednost: Traženi kontekst sesije tipa TR069SessionContext.

- **getRequestList(DeviceIdStruct deviceID)**

Kod ove metode se koristi isti mehanizam kao u prethodnoj sa razlikom da se ovde dobavlja red čekanja izlaznih poruka na osnovu identifikatora CPE uređaja.

Ulagni parametri: **deviceID** – identifikator CPE uređaja.

Povratna vrednost: Lista izlaznih poruka.

- **getPendingSOAP(TR069SessionContext sessionContext)**

Metoda proverava da li postoje izlazne poruke za dat uređaj i vraća prvu poruku iz reda čekanja ako ih ima.Za dobavljanje reda čekanje koristi identifikator uređaja koji je dostupan iz objekta koji opisuje kontekst sesije.Takođe, poslatu poruku zapisuje i u kontekst sesije za slučaj da CPE uređaj zahteva retransmisiju poruke.

Ulagni parametri: **sessionContext** – kontekst sesije.

Povratna vrednost: Sledeća poruka iz reda čekanja ili **null**ako nema više poruka.

- **generateFault(String faultSource, String faultCode, String faultString)**

Ako se prilikom provere ispravnosti poruke uoči neka greška, pomoću ove metode možemo izgenerisati izveštaj o grešci i poslati nazad CPE uređaju.CPE uređaj će na osnovu njegove logike pokušati da ispravlja grešku ili da završi sesiju.

Ulagni parametri: **faultSource** – opisuje koja strana je izazvala grešku, **faultCode** – šifra greške (šifre su definisane u TR-069 standardu), **faultString** – opis greške.

Povratna vrednost: Formatirana SOAP poruka sa izveštajem o grešci.

4.1.4 TR069CommunicationAPI

Ovaj modul predstavlja prilagodni sloj za TR-069 komunikaciju višim slojevima sistema. Pozivanjem metoda modula sloj za rukovanje sistemom može dodavati razne zahteve određenim CPE uređajima ili grupi uređaja. Kao kod modula za rukovanje porukama i ovde postoje dve vrste metoda. Metode za generisanje poruka i takozvane pomoćne metode.

4.1.4.1 Metode za generisanje poruka

Metode ovog tipa omogućuju formatiranje poruka i smeštanje istih u red čekanja zadatog uređaja ili grupi uređaja. Podržane su sve obavezne poruke TR-069 protokola i još neke važnije ali neobavezne. Spisak ovih metoda je dat u Tabeli 5.

```
public void getParameterValues(String requestID,
                               DeviceIdStruct deviceID,
                               ArrayList<String> paramNames)

public void setParameterValues(String requestID,
                               DeviceIdStruct deviceID,
                               String commandKey,
                               List<ParameterValueStruct> parameters)

public void updateFirmware(ArrayList<DeviceIdStruct> deviceList,
                           String commandKey,
                           String fileType,
                           String url,
                           long fileSize,
                           long delaySeconds,
                           String targetFileName)

public void getRPCMethods(String requestID,
                         DeviceIdStruct deviceID)

public void getParameterNames(String requestID,
                             DeviceIdStruct deviceID,
                             String path,
                             boolean nextLevel)

public void getParameterAttributes(String requestID,
                                  DeviceIdStruct deviceID,
```

```

        ArrayList<String> paramNames)

public void setParameterAttributes(String requestID,
                                  DeviceIdStruct deviceID,
                                  List<SetParameterAttributesStruct> parameters)

public void addObject(String requestID,
                      DeviceIdStruct deviceID,
                      String objectName,
                      String parameterKey)

public void deleteObject(String requestID,
                        DeviceIdStruct deviceID,
                        String objectName,
                        String parameterKey)

public void reboot(String requestID,
                   DeviceIdStruct deviceID,
                   String commandKey)

public void upload(String requestID,
                   DeviceIdStruct deviceID,
                   String commandKey,
                   String fileType,
                   String url,
                   String username,
                   String password,
                   int delaySeconds)

public void scheduleInform(String requestID,
                           DeviceIdStruct deviceID,
                           String commandKey,
                           int delaySeconds)

public void factoryReset(String requestID,
                        DeviceIdStruct deviceID)

```

Tabela 5. Spisak metoda za generisanje poruka.

4.2 Klase koje predstavljaju podatke.

Klase ovog tipa imaju ulogu da grupišu povezane podatke u jedan objekat radi što efikasnijeg rada sistema.

4.2.1 TR069SessionContext

4.2.1.1 Polja

- **sessionState** – Polje za čuvanje stanja sesije.
- **deviceId** – Identifikator uređaja koji je započeo sesiju.
- **lastSentEnv** – Poslednje poslata poruka. Uloga ovog parametra je dikutovana kod getPendingSOAP metodeTR069MessageHandler modula.
- **lastActivity** – Vreme kada je sesija poslednji put bila aktivna. Ako sesija nije aktivna neko predefinisano vreme ona se progasi inaktivnom i ugasi se.

4.2.2 TR069Request

4.2.2.1 Polja

- **tr069RPC** – TR-069 poruka koja treba da se posalje.
- **requestID** – Identifikatorporuke na osnovu čega se obavesti sloj za rukovanje sistemom ako se primi odgovor na datu poruku

4.2.3 TR069Event

4.2.3.1 Polja

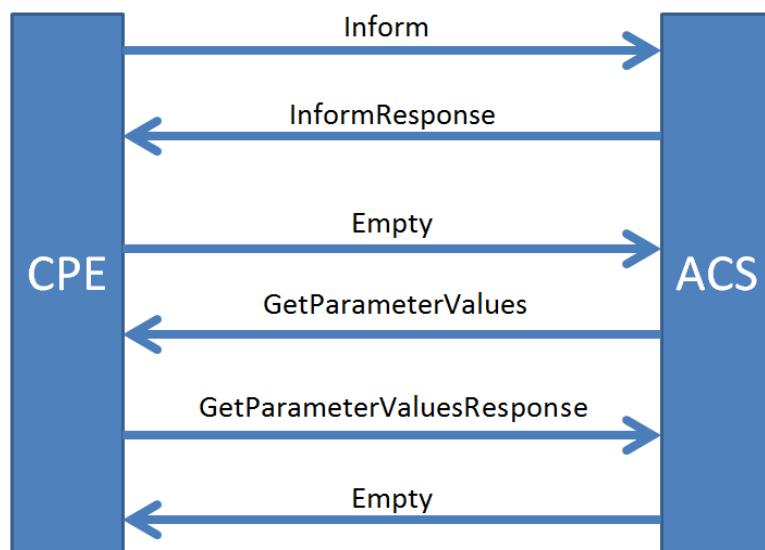
- **eventType** – Tip događaja na osnovu čega sloj za rukovanje sistemom zna kako da reaguje na događaj.
- **requestId** – IdentifikatorTR-069 poruke na koju je stigao odgovor.
- **deviceID** – Identifikator CPE uređaja koji je izazvao događaj.

5. Rezultati

Komunikacioni sloj je testiran kako u realnim uslovima, kao elementarni deo konfiguracionog poslužioca, tako i pomoću specijalizovanih alata za testiranje ovakvih rešenje. Za testiranje je korišćen alat SoapUI, koji predstavlja web baziranu aplikaciju za testiranje servisno orijentisanih arhitektura (SOA, eng. Service Oriented Architecture) [6] . Ovaj alat ima mogućnosti da pokrije dosta testnih slučajeva od komunikacije do testova opterećenja i kao takav predstavlja najbolje rešenje za testiranje komunikacionih slojeva poslužioca svih vrsta.

Kroz testiranje jedne kompletne sesije sa krajnjim uređajem razmenjene su standardizovane poruke kao što su *Inform* i *GetParameterValues*. U slučaju testiranja sa SoapUI alatom, on preuzima ulogu krajnjeg uređaja a korisnik u njemu kreira zahteve za slanje i definiše odgovore na komandne poruke.

Izgled testirane sesije je prikazan na Slici 14.



Slika 14. Jedna sesija između konfiguracionog poslužioca i krajnjeg uređaja.

Pojedinačne poruke koje su poslate komunikacionom sloju konfiguracionog poslužioca iz alata SoapUI su prikazane na Slici 15 za Inform, odnosno na slikama 16 za GetParameterValues i 17 za praznu poruku. Upoređivanjem odgovora dobijenog od poslužioca sa zahtevima u TR-069 standardu možemo zaključiti da je implementiran modul ispravan.

The screenshot shows the SoapUI interface with the title "RE ST Inform Request". The "Request" tab is selected, displaying an XML message. The XML content is as follows:

```

<?xml version="1.0"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope">
<soap-env:Header>
<cwmp:ID soap-env:mustUnderstand="1">17777789389</cwmp:ID>
<cwmp:HoldRequests soap-env:mustUnderstand="1">1</cwmp:HoldRequests>
</soap-env:Header>
<soap-env:Body>
<cwmp:Inform>
<DeviceId>
<Manufacturer>RTRK</Manufacturer>
<OUI>147A55</OUI>
<ProductClass>FooGate 2012</ProductClass>
<SerialNumber>022880000999</SerialNumber>
</DeviceId>
<Event soap-enc:arrayType="cwmp:EventStruct[2]">
<EventStruct>
<EventCode>1 BOOT</EventCode>
<CommandKey></CommandKey>
</EventStruct>
<EventStruct>
<EventCode>2 PERIODIC</EventCode>
<CommandKey></CommandKey>
</EventStruct>
</Event>
<MaxEnvelopes>1</MaxEnvelopes>
<CurrentTime></CurrentTime>
</cwmp:Inform>
</soap-env:Body>
</soap-env:Envelope>

```

The "Response" tab is also visible, showing the response XML. The status bar at the bottom indicates a response time of 1285ms (614 bytes).

Slika 15. Inform poruka u SoapUI alatu.

The screenshot shows the SoapUI interface with the title "RE ST GetParamValuesResp". The "Request" tab is selected, displaying an XML message. The XML content is as follows:

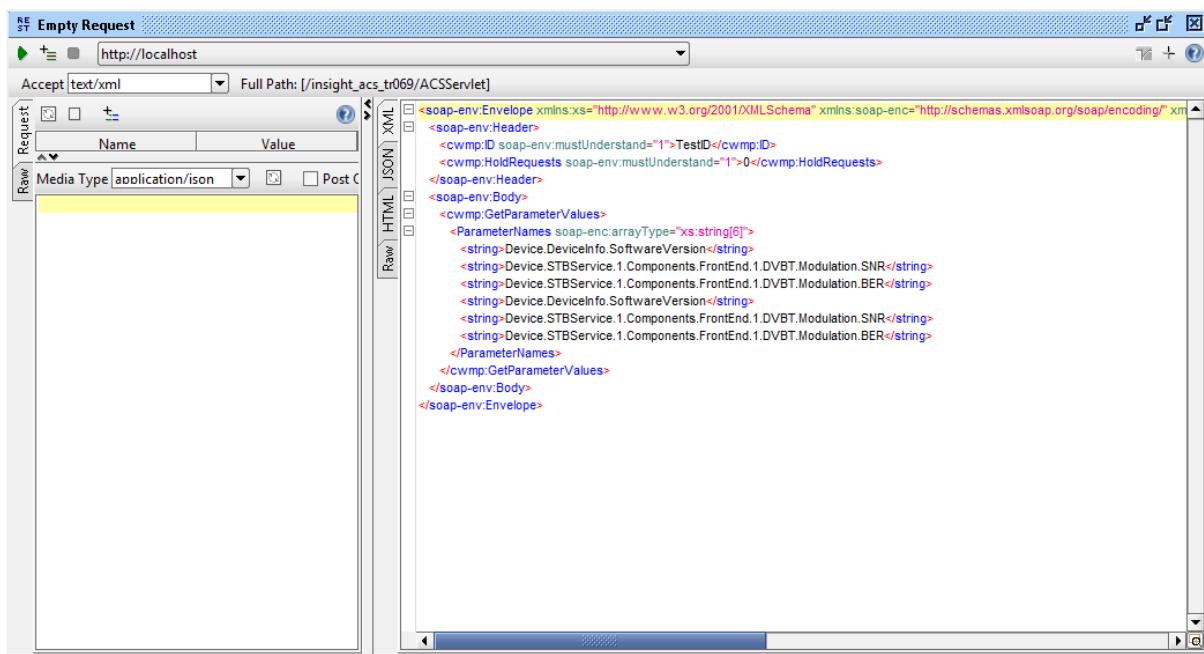
```

<?xml version="1.0"?>
<soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope">
<soap-env:Header>
<cwmp:ID soap-env:mustUnderstand="1">ID:intnrl.unset.id.GetParameterValues1361890617620.296132</cwmp:ID>
</soap-env:Header>
<soap-env:Body>
<cwmp:GetParameterValuesResponse>
<ParameterList soap-enc:arrayType="cwmp:ParameterValueStruct[3]">
<ParameterValueStruct>
<Name>Device.DeviceInfo.HardwareVersion</Name>
<Value xsi:type="xsd:string">test</Value>
</ParameterValueStruct>
<ParameterValueStruct>
<Name>Device.DeviceInfo.SoftwareVersion</Name>
<Value xsi:type="xsd:string">test</Value>
</ParameterValueStruct>
<ParameterValueStruct>
<Name>Device.STBService.1.Components.FrontEnd.1.DVBT.Modulation.Frequency</Name>
<Value xsi:type="xsd:unsignedInt">1313</Value>
</ParameterValueStruct>
</ParameterList>
</cwmp:GetParameterValuesResponse>
</soap-env:Body>
</soap-env:Envelope>

```

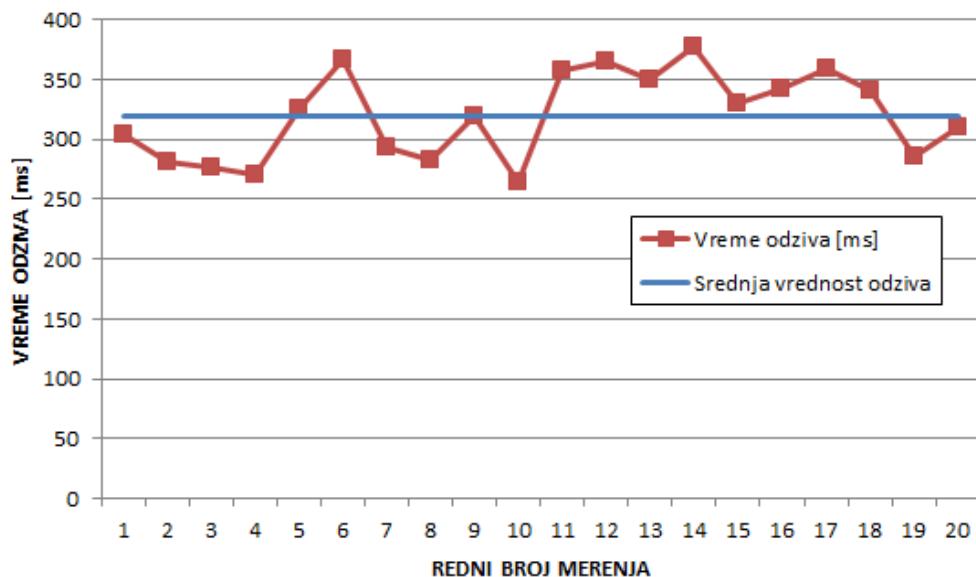
The status bar at the bottom indicates an HTTP/1.1 204 No Content response from Apache-Coyote/1.1 on Fri, 14 Jun 2013 08:19:55 GMT.

Slika 16. GetParameterValues poruka u SoapUI alatu.

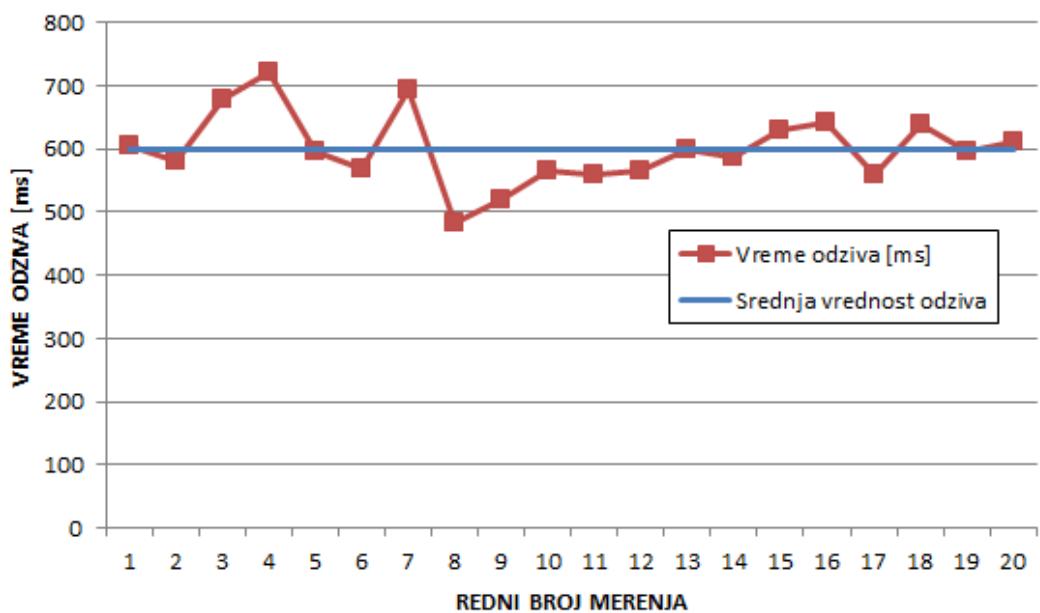


Slika 17. Prazna poruka u SoapUI alatu.

U sklopu funkcionalnog testiranja pomoću SoapUI alata, sprovedeno je i testiranje na vreme odziva. Test se sastoji od 20 testnih slučajeva od kojih svaki meri vreme odziva na *Inform* poruku. Testiranje je izvršeno dva puta, prvo u lokalnoj mreži pa zatim preko interneta. Rezultati su prikazani na Slikama 18 i 19. Srednje izmereno vreme iznosi 320,1 milisekundi u lokalnoj mreži i 603,7 milisekundi kod konekcije preko interneta bez obzira na opterećenje komunikacionog sloja brojem CPE uređaja što je uporedivo sa postojećim rešenjima [7].



Slika 18. Merenje vremena odziva u lokalnoj mreži.



Slika 19. Merenje vremena odziva preko interneta

Šem specijalizovanih testiranja pomoću već pomenutog alata, komunikacioni sloj je testiran u realnim uslovima sa pravim krajnjim uređajima i konfiguracionim poslužiocem. Na Slici 20 je prikazana korisnička korisnička sprega realizovana preko web bazirane aplikacije. Na slici se mogu primetiti registrovani uređaji koji sa programskim poslužiocem komuniciraju upravo preko komunikacionog sloja realizovanog u ovom rešenju.

The screenshot shows the 'Device List' section of the INSIGHT interface. On the left, there's a sidebar with 'Devices' (Device List, Firmware Upgrade), 'Stats', and 'Configuration'. The main area displays a table of devices:

	Name	Manufacturer	Last Seen
STB	Beograd	RT-RK	2013-06-14 10:23
STB	device_id_1	RT-RK	2013-06-13 18:51
STB	device_id_10	RT-RK	2013-06-11 18:30
STB	device_id_11	RT-RK	2013-06-11 18:30
STB	device_id_12	RT-RK	2013-06-11 18:30
STB	device_id_13	RT-RK	2013-06-11 18:30
STB	device_id_14	RT-RK	2013-06-11 18:30
STB	device_id_15	RT-RK	2013-06-11 18:30
STB	device_id_3	RT-RK	2013-06-11 14:01
STB	device_id_4	RT-RK	2013-06-12 20:56

To the right, a 'Brief Information' panel shows details for the 'Beograd' device:

- Serial Number: 915A7
- Name: Beograd
- Manufacturer: RT-RK
- OUI: 147A55
- Product Class: SABC4RS

Below that is a user profile section with fields for ID, Name, Provider, Address, and Phone.

Slika 20. Grafička korisnička sprega konfiguracionog poslužioca.

6. Zaključak

Zadatak ovog rada je bio da se realizuje komunikacioni sloj za konfiguracioni poslužilac koji je kompatibilan sa TR-069 standardom. Programsko rešenje je realizovano modularno kako bi se zadržala brzina obrade i slanja poruka između entiteta u komunikaciji. Sem toga komunikacioni sloj je prilagodljiv što ga čini lakis za održavanje. Kako je jedna od osobina ovog sloja i laka proširivost, budući rad na ovom rešenju će obuhvatiti i kreiranje komunikacionog sloja koji podržava dodatne standarde sem TR-069, kao što su SNMP (eng. Simple Network Management Protocol) ili specifično razvijeni protokol korisnika ovog sistema. To će ovakovom sistemu omogućiti punu fleksibilnost i nezavisnost od standarda komunikacije. Optimizacija po pitanju brzine i trošenja memorije je moguća i biće takođe predmet daljeg istraživanja i rada.

7. Literatura

- [1] TR-069 (CPE Wan Management Protocol), Broadband Forum,
<http://www.broadband-forum.org/cwmp>
- [2] „Enterprise JavaBeans 3.0 Fifth Edition“, Bill Burke, Richard Monson-Haefel, 2006.
- [3] „Building Java Enterprise Systems with J2EE“, Paul Perrone, S.R.R. Chaganti, 2000.
- [4] „Deeply embedded XML communication: towards an interoperable and seamless world“, Johannes Helander, 2005.
- [5] „A framework for self-configuring devices using TR-069“, Rachidi, H, 2011.
- [6] „Building test steps for SOA service orchestration in web service testing tools“, Hoijin Yoon, EunMi Ji, Byoungju Choi, 2008.
- [7] „Method for managing a communication between a server device and a customer device“, Liekens, Werner Mario, 2008