УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Милан Савић

Реализација подршке захтева за повезивањем заснованог на TR-069 протоколу

МАСТЕР РАД

Нови Сад, 2013

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

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:			
Идентификациони број, **ИБР**:			
Тип документације, **ТД**:		Монографска документација	
Тип записа, **ТЗ**:		Текстуални штампани материјал	
Врста рада, **ВР**:		Дипломски – мастер рад	
Аутор, **АУ**:		Милан Савић	
Ментор, **МН**:		Проф. Др Иштван Пап	
Наслов рада, **НР**:		Реализација подршке захтева за повезивањем заснованог на TP-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/страна/ цитата/табела/слика/графика/прилога)			
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кqучне речи, **ПО**:		Захтев за повезивањем, TP-069 протокол, TP-111 протокол	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		У овом раду је реализован захтев за повезивање који се ослања на TP-069 протокол. Захтев за повезивање представља механизам који омогућава послужиоцу да на захтев добије податке од крајњег уређаја. Захтев за конекцију представља аутентификациони процес који уколико је успешан резултује иницирањем размене података између конфигурационог послужиоца и крајњег уређаја. Уколико се крајњи уређај налази у истом адресном простору као и послужилац TP-069 захтев за повезивање се користи. Уколико се крајњи уређај налази иза усмеривача протокола користи се TP-111 захтев за повезивање.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	Проф. др Јелена Ковачевић	
	Члан:	Проф. др Милош Сланкаменац	Потпис ментора
	Члан, ментор:	Проф. др Иштван Пап	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master Thesis	
Author, **AU**:		Milan Savić	
Mentor, **MN**:		Ištvan Pap, PhD	
Title, **TI**:		Realization of connection request mechanism 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/appendixes)			
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		Connection request, TR-069 protocl, TR-111 protocol	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		In this paper an approach for managing device over a network that relies on TR-069 and TR-111 protocols has been implemented. This mechanism enables the configuration server to request data from end device at any time. This type of mechanism is called connection request. There are two types of connection requests, TR-069 and TR-111. If the device is in the same address space as configuration server, then TR-069 connection request is used. On the other hand if the device is behind a gateway then the TR-111 connection request is used.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	Jelena Kovačević, PhD	
	Member:	Miloš Slankamenac, PhD	Menthor's sign
	Member, Mentor:	Ištvan Pap, PhD	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Zahvalnost

Želim da se zahvalim mojoj devojci Ivani na podršci, brizi i strpljenju pri izradi ovog master rada.

Posebno se zahvaljujem mentoru Ištvanu Papu na savetima i smernicama pruženim tokom izrade rada.

Zahvaljujem se mojoj porodici na podršci.

Na kraju želim da se zahvalim svima koji su na bilo koji način doprineli izradi ovog master rada.

SADRŽAJ

SPISAK SLIKA

SPISAK TABELA

SKRAĆENICE

ACS	*- **A**uto **C**onfiguration **S**erver*
CPE	*- **C**ustomer **P**remises **E**quipment*
SNMP	*- **S**imple **N**etwork **M**anagement **P**rotocol*
NAT	*- **N**etwork **A**ddress **T**ranslation*
STUN	*- **S**ession **T**raversal **U**tilities for **N**AT*
HTTP	*- **H**yper**t**ext **T**ransfer **P**rotocol*
IP	*- **I**nternet **P**rotocol*
UDP	*- **U**ser **D**atagram **P**rotocol*
TCP	*- **T**ransmission **C**ontrol **P**rotocol*
API	*- **A**pplication **P**rogramming **I**nterface*

1. Uvod

U današnje vreme, izražena je potreba za nadgledanjem, upravljanjem i konfigurisanjem uređaja preko mreže. Sa razvojem interneta ta potreba je rasla i kao odgovor dizajniran je SNMP (eng. Simple Network Management Protocol) koji je imao brojne prednosti ali i nedostatke koji su se ogledali u nedovoljno standardizovanoj podršci za uređaje različitih proizvođača. Sa težnjom da se otklone ključni nedostaci nastao je TR-069 protokol koji je nezavistan od tipa i proizvođača uređaja. Jedna od ključnih karakteristika TR-069 protokola je da se sva interakcija između TR-069 klijenta i TR-069 poslužioca odvija na zahtev klijenta, koji je odgovoran za uspostavljanje i okončanje transfera podataka. To dovodi do problema da poslužilac dobija podatke samo onda kada mu klijent dozvoli. Kako bi omogućili poslužiocu da inicira razmenu podataka sa klijentom razvijeni su TR-069 i TR-111 zahtevi za konekciju (eng. Connection Request). Oni omogućavaju poslužiocu da u bilo kom momentu signalizira klijentu da započne razmenu podataka.

U ovom radu realizovan je zahtev za povezivanjem zasnovan na TR-069 i TR-111 protokolima. Akcenat je stavljen na što manju zavisnost od spoljnih biblioteka sa ciljem da se smanji veličina samog programskog rešenja i ostvari veća portabilnost. Pošto rešenje zahteva implementaciju i na strani klijenta i na strani poslužioca, realizacija je izvedena u programskom jeziku C i programskom jeziku Java. Rešenje na klijentskoj strani je implementirano na operativnom sistemu Linux i preneseno na Android i STLinux operativne sisteme. Rešenje na strani poslužioca je implementirano u okviru ACS (eng. Auto Configuration Server) poslužioca. Realizacija je izvedena u više modula radi lakšeg razvoja i održavanja programske podrške. TR-111 i TR-069 zahtevi za povezivanje integrisani su i testirani u okviru postojećeg ACS poslužioca.

Ovaj rad je sačinjen od sedam poglavlja.

Drugo poglavlje izlaže teorijske osnove, pre svega TR-069 i TR-111 zahteva za povezivanje, kao i STUN protokola.

U trećem poglavlju dat je koncept rešenja za implementaciju TR-069 i TR-111 zahteva za povezivanje u vidu klijenata i poslužilaca.

Četvrto poglavlje daje detaljan opis modula unutar programskog rešenja i načine njihovog sprezanja.

U petom poglavlju su dati načini ispitivanja i testiranja TR-069 i TR-111 zahteva za povezivanje.

Šesto poglavlje sadrži moguće dalje pravce razvoja i pregled postojećeg programskog rešenja.

U sedmom poglavlju navedena je korišćena literatura.

Osmo poglavlje predstavlja dodatak sa detaljno definisanom programskom spregom rešenja

2. Teorijske osnove

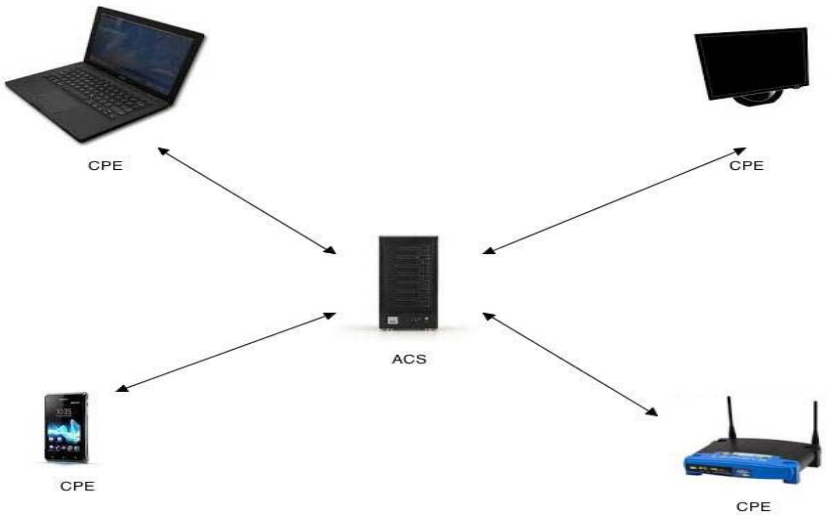
U ovom poglavlju date su teorijske osnove neophodne za implementaciju programskog rešenja zahteva za povezivanje zasnovanog na TR-111 i TR-069 protokolima. Zahtev za povezivanje predstavlja spregu između programske podrške krajnjih uređaja i poslužioca za automatsku konfiguraciju.

2.1 TR-069 protokol

TR-069 (eng. Technical Report 069) definiše protokol za rukovanje krajnjim uređajima [1]. Korišćenjem ovog protokola veza se uspostavlja između krajnjeg uređaja (eng. CPE – Customer Premise Equipment) i ACS-a. Protokolom je opisan samo način prenosa podataka, dok su definicije podataka koji se prenose date modelima podataka pridruženim TR-069 protokolu. U ovom radu korišćeni su TR-106 i TR-135 modeli podataka. TR-106 model podataka [2] sadrži osnovne parametre uređaja, među kojima su i parametri neophodni za uspostavu veze između TR-069 klijenta i poslužioca. TR-135 model podataka [3] je namenjen STB uređajima i sadrži parametre karakteristične za set-top box uređaje. Neke od osnovnih funkcionalnosti koje pruža TR-069 standard su:

- Nadgledanje statusa i performansi
- Rukovanje programskom podrškom krajnjeg uređaja
- Automatska konfiguracija

Komunikacija između krajnjeg uređaja i poslužioca se odvija samo na zahtev uređaja. Na slici 2.1 se može videti kako izgleda TR-069 okruženje. Razmena podataka između uređaja i poslužioca se odvija u periodičnim vremenskim intervalima. Problem takvog vida komunikacije ogleda se u tome da na taj način poslužilac nema načina da momentalno dobije podatka sa uređaja. Kako bi se to omogućilo definisan je TR-069 zahtev za povezivanje.



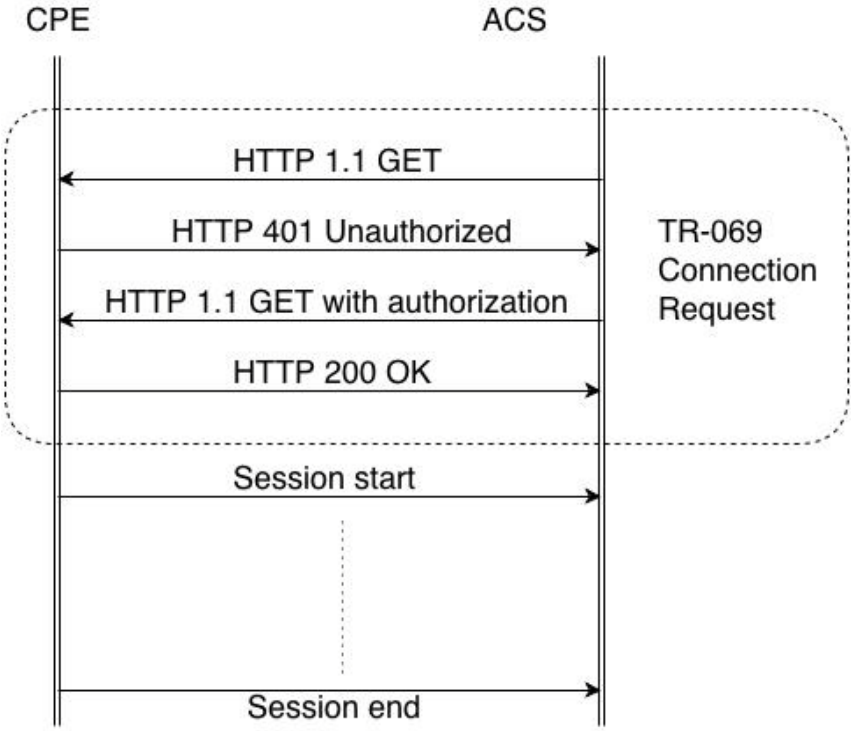
Slika 2. 1 Primer okruženja u TR-069 okruženju

Zahtev za povezivanje predstavlja mehanizam koji omogućava poslužiocu da signalizira krajnjem uređaju da započne razmenu podataka. U slučaju zahteva za povezivanje krajnji uređaj ima ulogu poslužioca, dok ACS poslužilac ima ulogu klijenta. Krajnji uređaj se u tom slučaju ponaša kao HTTP poslužilac. Mehanizam predstavlja proces autentifikacije. Nakon što je ona uspešno izvršena, krajnji uređaj započinje razmenu podataka (sesiju) sa poslužiocem. Ukoliko krajnji uređaj ima IP adresu dostupnu sa poslužioca (eng. routable IP address) koristi se TR-069 zahtev za konekciju.

Mehanizam za autentifikaciju zasniva se na HTTP *digest* protokolu [4]. TR-069 zahtev za povezivanje se sastoji od dve komponente, poslužioca i klijenta. Krajnji uređaj dobija ulogu HTTP poslužioca, dok ACS preuzima ulogu klijenta. Svi podaci neophodni za autentifikaciju se prenose preko zaglavlja HTTP poruka i kroz TR-106 model podataka. Na slici 2.2 se može videti da uspešna autentifikacija predstavlja okidač za početak sesije između krajnjeg uređaja i poslužioca.

Prvo klijent, u ovom slučaju ACS, šalje HTTP 1.1 Get zahtev i dobija odgovor (eng. response) u vidu neovlašćenog pristupa (eng. unauthorized request). Odgovor sadrži informacije koje su ključne za uspešnu autentifikaciju. U kombinaciji sa informacijama koje su dostupne kroz TR-106 model podataka klijent je ispunio sve potrebne preduslove za uspešno prijavljivanje poslužiocu. Ponovo se pristupa slanju HTTP GET zahteva ali sada taj zahtev sadrži dodatne informacije koje su neophodne za uspešnu autentifikaciju. Kao odgovor na ovakav zahtev klijent prima HTTP 200 OK odgovor, pod uslovom da su podaci poslati kroz HTTP GET zahtev ispravni. Ukoliko podaci koji se prosleđuju kroz zaglavlje HTTP poruke nisu ispravni, poslužilac kojeg u ovom slučaju predstavlja krajnji uređaj će poslati odgovor u vidu neovlašćenog pristupa. Uspešna autentifikacija služi kao signal krajnjem uređaju da otpočne razmenu podataka sa ACS

poslužiocem. Iako u potpunosti ima kontrolu nad započinjanjem razmene podataka između krajnjeg uređaja i poslužioca, TR-069 zahtev za povezivanje nema nikakav uticaj na tok razmene podataka niti na okončanje sesije.



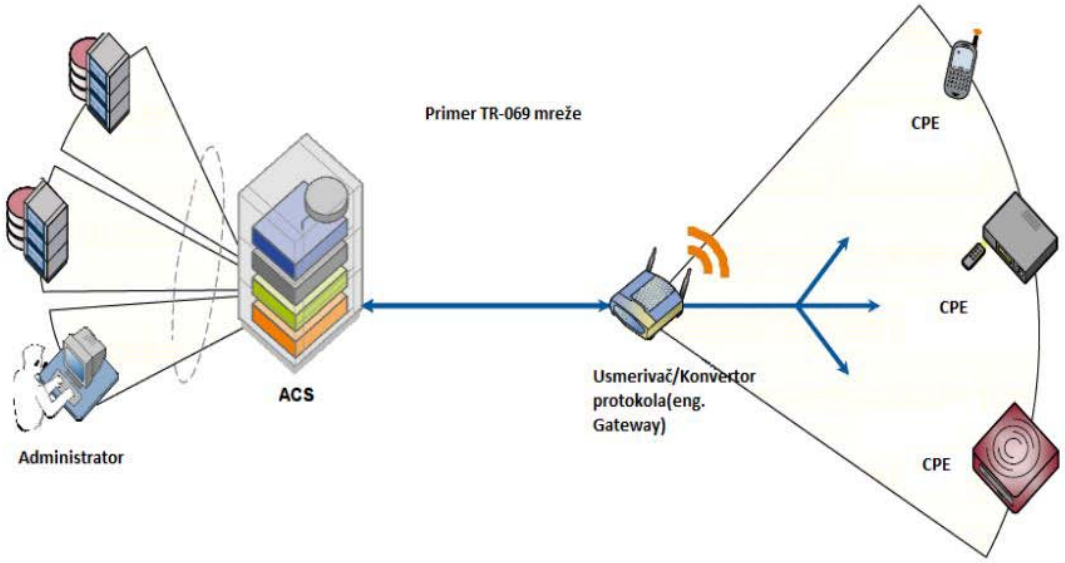
Slika 2. 2 Razmena podataka između krajnjeg uređaja i poslužioca na primeru TR-069 zahteva za konekciju

2.2 TR-111 protokol

Ukoliko krajnji uređaj nema adresu dostupnu sa poslužioca koristi se TR-111 zahtev za konekciju [5]. On omogućava ACS-u da komunicira sa uređajem iako se on nalazi iza usmerivača protokola. Na slici 2.3 može se videti izgled TR-111 okruženja. TR-111 zahtev za povezivanje sastoji se od dve celine:

- STUN klijent i poslužilac
- TR-111 klijent i poslužilac

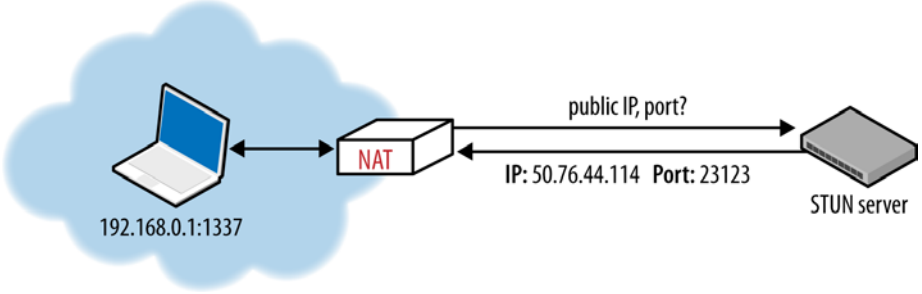
STUN mehanizam [6] predstavlja metod za određivanje i održavanje IP adrese na konvertoru protokola preko koje je moguće pristupiti krajnjem uređaju. TR-111 klijent i poslužilac su zaduženi za iniciranje konekcije između krajnjeg uređaja i ACS-a, kao u slučaju TR-069 zahteva za povezivanje. Razlika u odnosu na TR-069 zahtev za povezivanje ogleda se u korišćenju UDP poruka za komunikaciju između klijenta i poslužioca. Nakon uspešno izvršenog zahteva za konekciju transfer podataka između CPE-a i ACS-a se obavlja na isti način kao i u slučaju TR-069 zahteva za konekciju.



Slika 2. 3 Izgled TR-111 okruženja

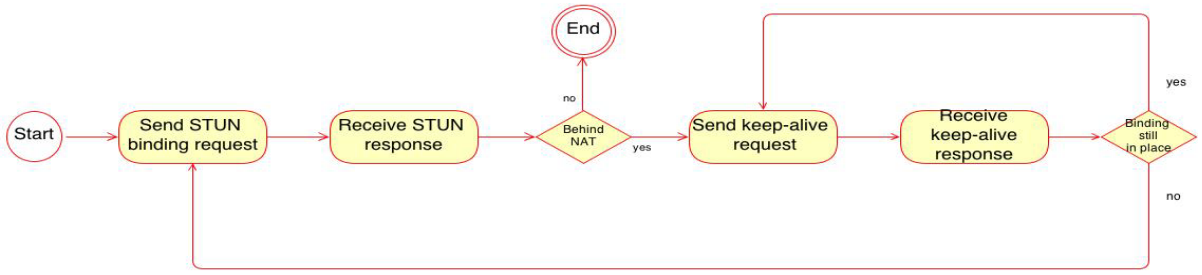
2.2.1 STUN protokol

Uloga STUN poslužioca i klijenta ogleda se u otkrivanju adrese i porta preko kojih je moguće vršiti interakciju sa uređajem. Pored otkrivanja adrese, STUN mehanizam ima ulogu da održava vezu na toj adresi i portu i da ne dozvoli da se ta povezanost (eng. binding) raskine. STUN poslužilac može a i ne mora biti lociran na istom poslužiocu koji je zadužen za pokretanje ACS-a, dok je klijent lociran na krajnjem uređaju. Sam mehanizam otkrivanja translacije adrese i porta je relativno jednostavan. Klijent šalje zahtev za vezivanje STUN poslužiocu, odgovor koji poslužilac šalje klijentu u sebi sadrži adresu na koju je taj odgovor poslat. Na taj način klijent dobija saznanje o postojanju usmerivača poredeći svoju adresu sa adresom u poruci. Ukoliko se port ili adresa razlikuju jasno je da se između klijenta i poslužioca nalazi usmerivač. Ukoliko jeste slučaj da se klijent nalazi iza usmerivača, pristupa se mehanizmu održavanja veze (eng. keep-alive). Tada klijent šalje periodične poruke sa ciljem da se vezivanje ne promeni tj. da dodeljena adresa i port koji su otkriveni prilikom zahteva za vezivanje ostanu nepromenjeni. Na slici 2.4 je prikazana STUN klijent/poslužilac komunikacija.



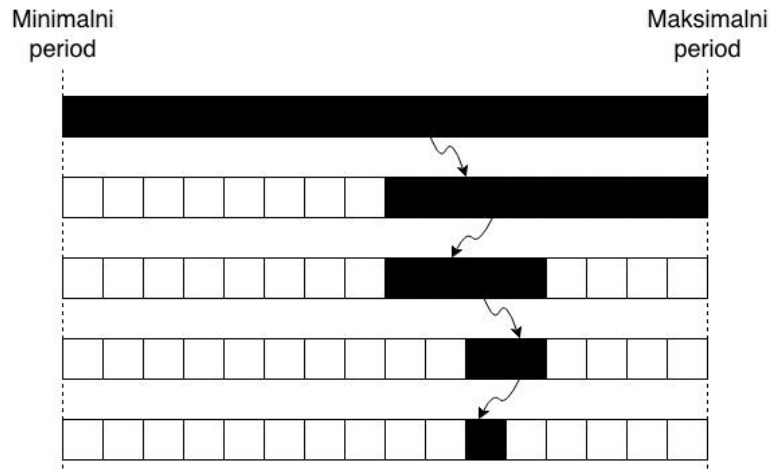
Slika 2. 4 STUN klijent-poslužilac komunikacija

Mehanizam održavanja veze se zasniva na slanju periodičnih poruka STUN poslužiocu i na taj način se veza (eng. binding) održava. STUN klijent šalje zahtev poslužiocu sa sekundarnog porta. Unutar zahteva se nalazi adresa na koju je potrebno poslati odgovor. Ukoliko odgovor stigne na adresu iz zahteva, koja je u ovom slučaju primarni port, dolazi se do zaključka da je veza održana. Ukoliko odgovor ne stigne klijentu ili stigne na sekundarni port sa kog je zahtev poslat, to je pokazatelj da veza nije održana i potrebno je opet pristupiti zahtevu za vezivanje. Da bi se osiguralo dostavljanje zahteva STUN poslužiocu, klijent pristupa mehanizmu ponovnog slanja (eng. retransmission). On se zasniva na slanju zahteva u tačno definisanim vremenskim razmacima [5]. Slanje se obustavlja kada se dobije odgovor od poslužioca ili kada istekne period retransmisije tj. ponovnog slanja. Važna stvar koja se mora uzeti u obzir leži u činjenici da se STUN poslužilac ne sme nalaziti iza usmerivača protokola [6]. To znači da STUN mehanizam neće funkcionisati ukoliko se STUN poslužilac nalazi unutar privatnog adresnog prostora. Na slici 2.5 se može videti postupak slanja zahteva za vezivanje i postupak održavanja veze.



Slika 2. 5 Postupak uspostavljanja i održavanja veze

Važan deo STUN mehanizma je period održavanja veze (eng. keep alive period). On je u TR-111 protokolu definisan pomoću dva parametra koji definišu maksimalan i minimalan period održavanja veze [5]. Ukoliko ta dva parametra nemaju istu vrednost, mora se pristupiti eksperimentalnom određivanju tačne vrednosti perioda održavanja veze. Metoda binarne pretrage se koristi za određivanje tačne vrednosti. Proces binarne pretrage se zasniva na metodi polovljenja intervala čije su granice parametri minimalnog i maksimalnog perioda održavanja veze, jer slanje zahteva koji služi za osvežavanje veze ne sme biti ređe od vrednosti koju nosi parametar za maksimalni period održanja veze i ne sme biti češći od parametra minimalnog perioda održavanja veze. Proces binarne pretrage je završen kada se pronađe vrednost za koju veza ostaje održana i koja se nalazi na granici tako da svaka vrednost koja je veća od utvrđene rezultuje prekidom veze između klijenta i poslužioca. Na slici 2.6 je prikazan postupak određivanja tačne vrednosti nad intervalom pomoću binarne pretrage.



Slika 2. 6 Metoda binarne pretrage

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

TR-111 protokol definiše svoj način korišćenja STUN protokola koji se razlikuje od preporučenog. To se posebno odnosi na način otkrivanja adrese i porta preko kojih se može pristupiti uređaju. STUN protokol [6] sugeriše da se otkrije kakav tip translacije adrese i porta je u pitanju prilikom interakcije između STUN klijenta i poslužioca. TR-111 protokol ne sledi takav metod, već se samo određuje adresa i port preko kojih se može pristupiti uređaju, dok je tip translacije u potpunosti zanemaren i ne igra nikakvu ulogu u TR-111 zahtevu za povezivanje. Iz tih razloga STUN klijent nema potrebu da koristi sledeće atribute definisane u STUN protokolu:

- CHANGE-REQUEST
- CHANGED-ADDRESS
- SOURCE-ADDRESS
- REFLECTED-FROM

Takođe, STUN klijent ne podržava razmenu zajedničke tajne (eng. Shared Secret). Sa druge strane TR-111 protokol predviđa unošenje dva opciona atributa u zahtev za vezivanje, CONNECTION-REQUEST-BINDING i BINDING-CHANGE. Pošto predstavljaju opcione atribute koji se odbacuju ukoliko ih poslužilac ne podržava, njihovo ubacivanje u zahtev za

vezivanje neće uticati na kompatibilnost sa komercijalno dostupnim STUN poslužiocima. Iako ne utiče na kompatibilnost ovakva specifična definicija STUN klijenta onemogućava korišćenje komercijalno dostupnih klijenata u TR-111 protokolu.

2.2.2 TR-111 klijent i poslužilac

Drugi deo TR-111 zahteva za povezivanje čine TR-111 klijent i poslužilac. Poslužilac je lociran na krajnjem uređaju, dok ACS dobija ulogu klijenta. Sama postavka sistema je skoro identična TR-069 zahtevu za povezivanje sa jednom razlikom, umesto HTTP poslužioca koristi se UDP poslužilac. Komunikacija između klijenta i poslužioca se odvija isključivo jednosmerno. To znači da klijent ne dobija nikakvu povratnu informaciju od poslužioca. Klijent šalje zahtev za autentifikaciju i ukoliko je ona uspešna krajnji uređaj započinje razmenu podataka. Svi podaci neophodni za uspešnu autentifikaciju se nalaze u modelu podataka. TR-106 model podataka je proširen sa parametrima vezanim za TR-111 zahtevom za konekciju koji su prikazani u tabeli 2.1.

Naziv	Tip	Opis
UDPConnectionRequestAddress	string	Adresa na koju se šalju TR-111 zahtevi
UDPConnectionRequestAddress notification-limit	unsigned	Minimalno vreme u sekundama koje mora da protekne između 2 javljanja ACS poslužiocu da se UDP adresa promenila
STUNEnable	boolean	Parametar koji signalizira da li je STUN poslužilac operativan
STUNServerAddress	string	Adresa STUN poslužioca
STUNServerPort	unsigned	Port STUN poslužioca
STUNUsername	string	Korisničko ime za STUN poslužioca
STUNPassword	string	Šifra za STUN poslužioca
STUNMaximumKeepAlivePeriod	int	Maksimalni dozvoljeni period između dva zahteva za održavanje veze
STUNMinimumKeepAlivePeriod	unsigned	Minimalni dozvoljeni period između dva zahteva za održavanje veze
NATDetected	bool	Parametar koji signalizira da li se uređaj nalazi iza konvertora protokola

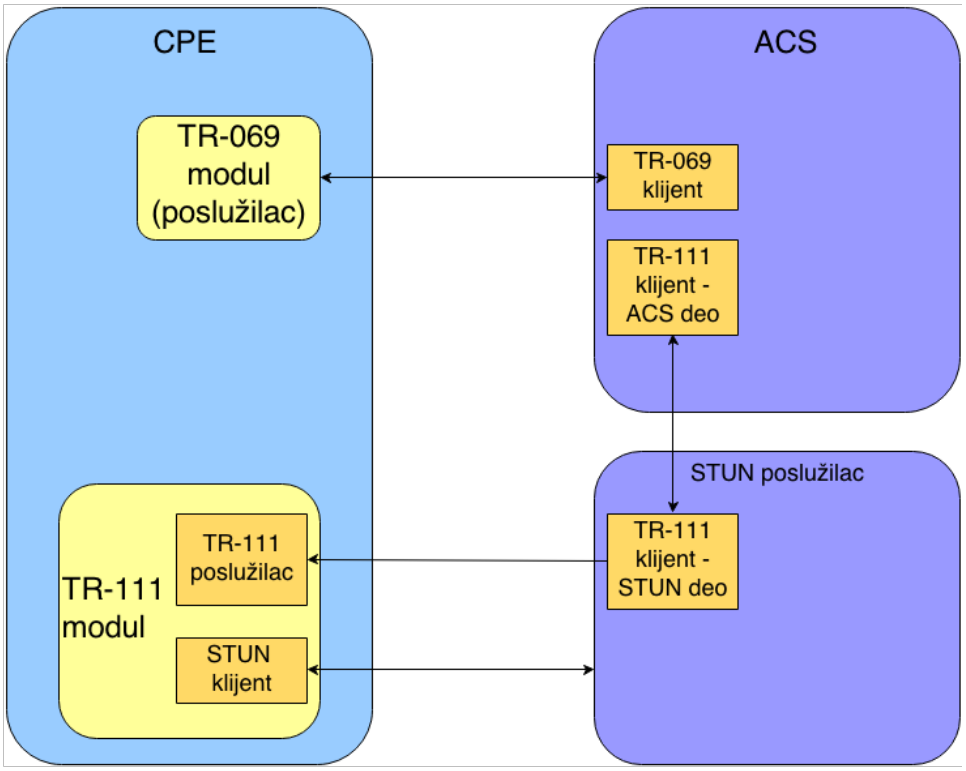
Tabela 2. 1 Spisak TR-111 parametara

TR-111 zahtev za povezivanje bazira se na HTTP 1.1 GET zahtevu [7], iako se sam HTTP 1.1 protokol ne koristi pri autentifikaciji. TR-111 zahtev za konekciju mora zadovoljiti sledeće uslove:

- Mora biti validan HTTP 1.1 GET zahtev.
- Zahtev ne sme imati telo poruke (eng. message body).
- Ukoliko se u zahtevu sadrži zaglavlje sa poljem dužina poruke (eng. content length), njena vrednost mora biti nula.
- Metoda u poruci mora biti GET.
- URI (eng. Uniform Resource Identifier) se formira na sledeći način:
    - URI mora započeti sa „http“ ili „HTTP“.
    - Authority deo URI-a mora biti formiran kao u rfc3986 [8].
    - URI putanja mora biti prazna
    - Upit deo URI-a sadrži string koji mora sadržati sledeće ime-vrednost parove:
        - ts – Vremenska oznaka (eng. timestamp)
        - id – Identifikacioni broj poruke
        - un – Korisničko ime
        - cn – Slučajno generisani string od strane ACS-a
        - sig – Potpis koji predstavlja HMAC-SHA1 sumu [9]

3. Koncept rešenja

TR-069 i TR-111 zahtevi za povezivanje su implementirani i na strani krajnjeg uređaja i na strani ACS poslužioca u vidu zasebnih modula. Na strani krajnjeg uređaja zahtevi za povezivanje su implementirani kao dva modula u sklopu TR-069 biblioteke (u daljem tekstu libcpe) koja implementira funkcionalnosti TR-069 klijenta [10]. Na ACS strani TR-069 i TR-111 klijenti su deo poslužioca dok je STUN poslužilac realizovan kao zasebna Java aplikacija, kao što je prikazano na slici 3.1.



Slika 3.1 Koncept rešenja TR-069 i TR-111 zahteva za konekciju

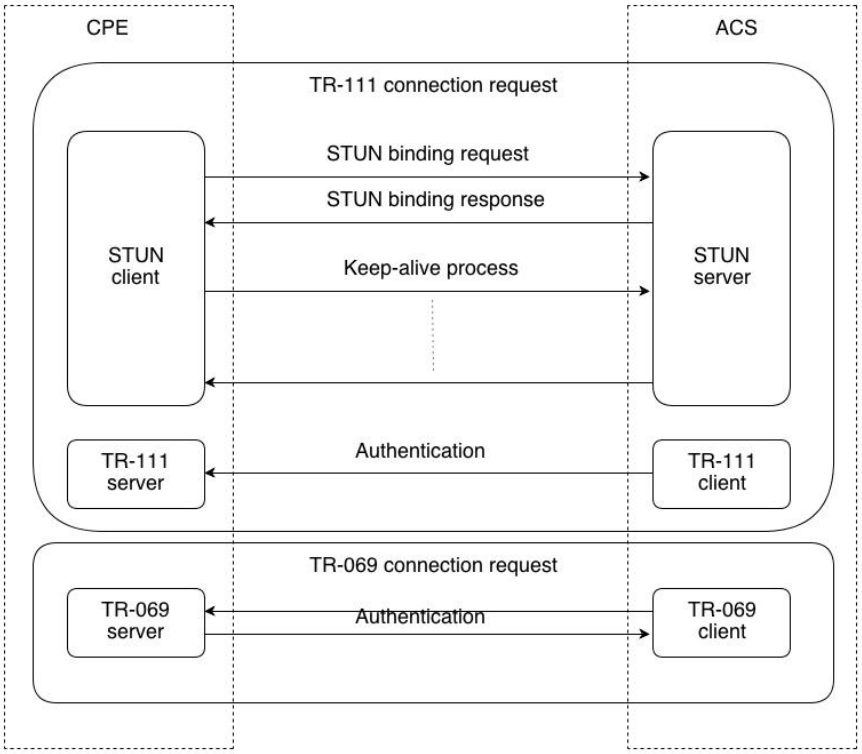
Na slici 3.1 se mogu jasno izdvojiti pet celina koje zajedno čine TR-069 i TR-111 zahteve za konekciju:

- TR-069 poslužilac
- TR-111 modul
- STUN poslužilac
- TR-069 klijent
- TR-111 klijent

TR-069 poslužilac i TR-069 klijent predstavljaju TR-069 zahtev za povezivanje. Sa druge strane STUN poslužilac, TR-111 klijent i TR-111 modul zaokružuju celinu TR-111 zahteva za povezivanje. Svi moduli na strani krajnjeg uređaja poseduju jasno definisan API koji omogućava laku integraciju sa libcpe. Prateći programsku spregu libcpe rešenja, svi moduli sadrže sledeće funkcionalnosti:

- Inicijalizacija
- Startovanje
- Zaustavljanje
- Deinicijalizacija

Samo funkcionisanje modula ni na koji način ne ometa funkcionisanje cele biblioteke. Tome doprinosi i sama svrha zahteva za povezivanje koji ni na koji način ne utiče na samu razmenu podataka već se njegova uloga ogleda u iniciranju razmene podataka. Na slici 3.2 se može videti princip funkcionisanja TR-069 i TR-111 zahteva za povezivanje.

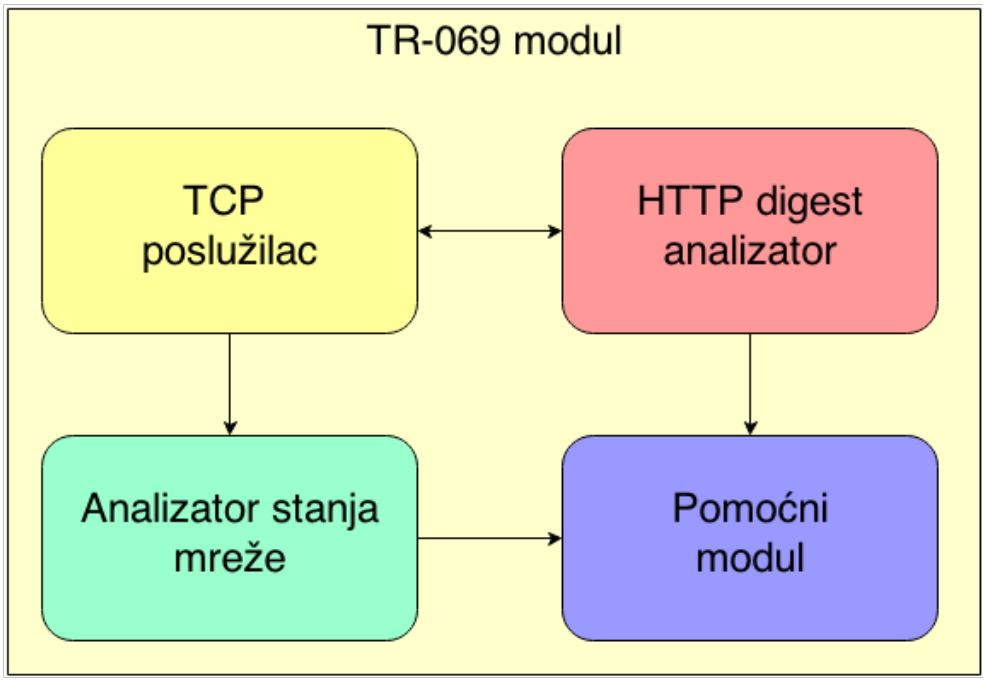


Slika 3. 2 Princip funkcionisanja TR-69 i TR-111 modula

3.1 TR-069 modul na strani krajnjeg uređaja

TR-069 modul predstavlja poslužioca koji vrši HTTP *digest* autentifikaciju klijenata. Veza između poslužioca i klijenta se zasniva na TCP protokolu. Ukoliko je autentifikacija uspešna krajnji uređaj započinje razmenu podataka sa ACS poslužiocem. Organizacija celina u okviru ovog modula prikazana je na slici 3.3. TR-069 modul na strani krajnjeg uređaja se sastoji od 4 komponente:

- TCP poslužioca
- HTTP *digest* analizatora
- Analizatora stanja mreže
- Pomoćnog modula



Slika 3. 3 Komponente TR-069 modula

TCP poslužilac predstavlja celinu koja je zadužena za rukovanje vezom između klijenta i poslužioca. Uloga ovog modula je da osluškuje da li će na određeni port stići poruka, prosledi je na autentifikaciju i pošalje odgovor klijentu. Modul sadrži sledeće funkcionalnosti:

- Početno podešavanje parametara TCP poslužioca
- Startovanje TCP poslužioca
- Prijem HTTP *digest* zahteva i slanje HTTP *digest* odgovora klijentu
- Zaustavljanje TCP poslužioca
- Deinicijalizacija TCP poslužioca

Analizator HTTP *digest* poruka je zadužen za analizu HTTP zahteva i za generisanje HTTP odgovora. Ovaj modul prima HTTP zahtev, analizira ga i kao rezultat analize generiše adekvatan odgovor.

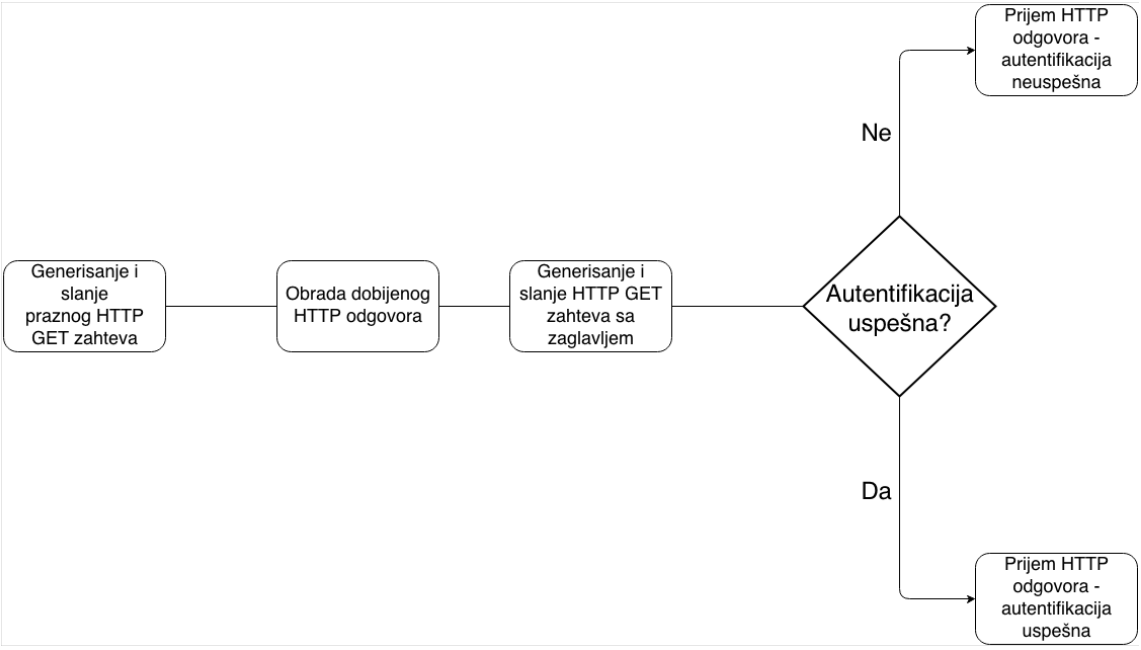
Analizator stanja veze je zadužen da prati status mreže , tj. da li postoji internet konekcija ili ne. Ukoliko dođe do toga da u trenutku internet konekcija nestane, TR-069 modul se zaustavlja i kada se internet konekcija opet uspostavi TR-069 modul se ponovo pokreće.

Pomoćni modul sadrži pomoćne funkcije koje olakšavaju rad TR-069 modula.

3.2 TR-069 klijent na strani ACS poslužioca

TR-069 klijent na strani poslužioca predstavlja modul u ACS-u koji je razvijen u Java programskom jeziku. On predstavlja HTTP *digest* klijenta koji vrši autentifikaciju ACS-a na strani krajnjeg uređaja. Autentifikacija se odvija iz dva koraka koja ukoliko su pravilno izvršena

vode ka iniciranju razmene podataka između ACS poslužioca i krajnjeg uređaja. Na slici 3.4. je prikazan princip rada TR-069 klijenta i tok autentifikacije. Prvi korak ka uspešnoj autentifikaciji je slanje praznog HTTP GET zahteva ka TR-069 poslužiocu. Na takvu poruku poslužilac šalje odgovor u kome se klijent obaveštava da mu pristup nije odobren, ali pored toga šalje sve potrebne informacije koje u kombinaciji sa podacima dostupnim kroz TR-106 model podataka čine sav neophodan materijal za uspešnu autentifikaciju. Nakon što klijent ima sve potrebne informacije, on ponovo šalje zahtev TR-069 poslužiocu koji u svom zaglavlju sadrži sve potrebne informacije za autentifikaciju. Poslužilac na takav zahtev šalje odgovor u kome se signalizira da je autentifikacija uspešno izvršena. Uspešna autentifikacija rezultuje započinjanjem razmene podataka između ACS-a i krajnjeg uređaja.

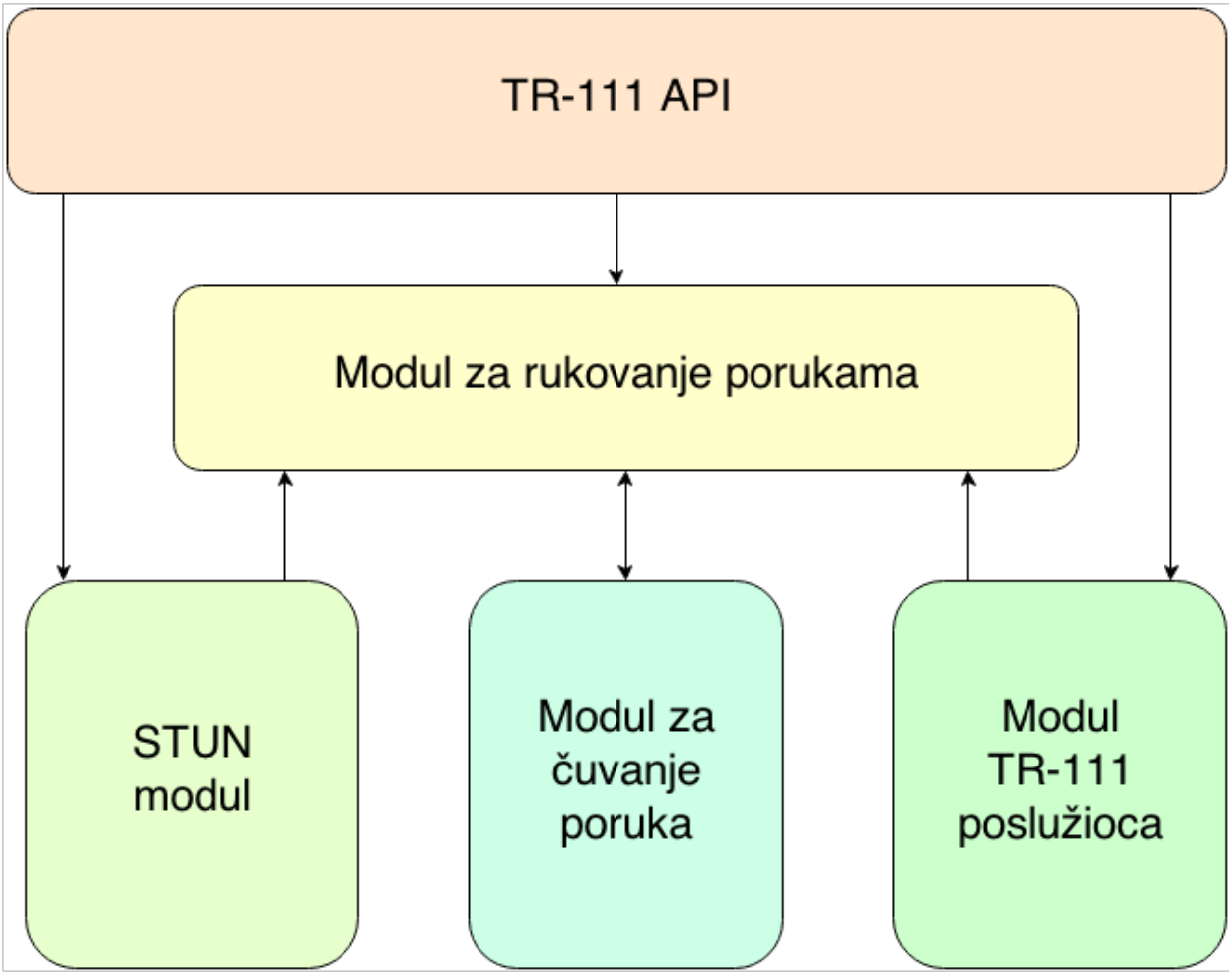


Slika 3. 4 Princip rada TR-069 klijenta

3.3 TR-111 modul na strani krajnjeg uređaja

Zahtev za povezivanje je baziran na TR-111 protokolu i na strani krajnjeg uređaja sastoji se od dve funkcionalne celine, TR-111 poslužioca i STUN klijenta. STUN klijent ima ulogu otkrivanja translacije adrese. Uloga TR-111 poslužioca se ogleda u obradi TR-111 zahteva. Funkcionalnost TR-111 poslužioca je gotovo identična TR-069 poslužiocu uz par izuzetaka. U pitanju je UDP poslužilac koji za razliku od poslužioca u TR-069 zahtevu za povezivanje ne generiše nikakav odgovor klijentu, tj. ne postoji povratna informacija prilikom autentifikacije. Na slici 3.5 su prikazani ključni elementi TR-111 zahteva za povezivanje. Koncept rešenja TR-111 zahteva za povezivanje se može podeliti u 5 celina:

- TR-111 API modul
- STUN klijent modul
- Modul TR-111 poslužioca
- Modul za rukovanje porukama
- Modul za čuvanje poruka



Slika 3. 5 Koncept rešenja TR-111 modula

3.3.1 TR-111 API modul

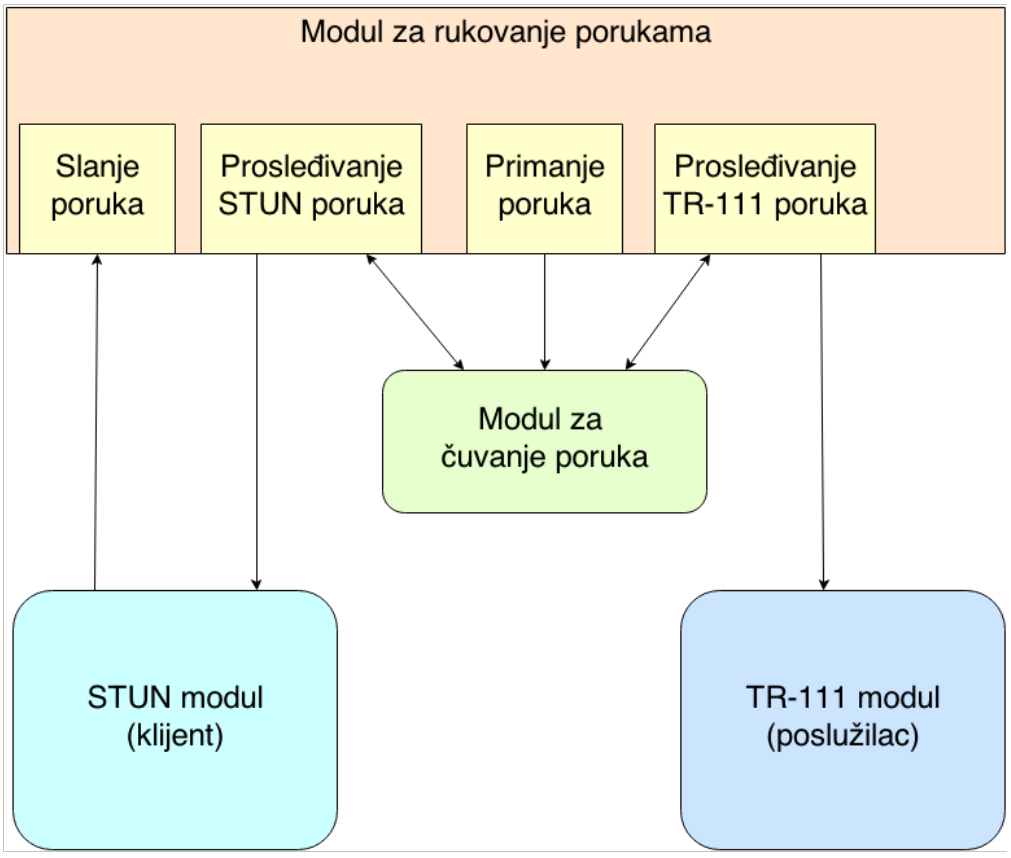
TR-111 API objedinjuje funkcionalnosti svih modula u okviru TR-111 zahteva za povezivanje. U suštini ova celina predstavlja upravljački modul celog TR-111 sistema. Ovaj modul omogućava sledeće funkcionalnosti nad TR-111 sistemom:

- Inicijalizaciju
- Deinicijalizaciju
- Pokretanje
- Zaustavljanje

API koji pruža ovaj modul se u potpunosti poklapa sa programskom spregom libcpe biblioteke. Time je olakšana integracija, a uvođenjem TR-111 API modula, količina izmena u libcpe biblioteci koji je neophodan za integraciju je sveden na minimum.

3.3.2 Modul za rukovanje porukama

Modul za rukovanje porukama predstavlja komunikacioni modul koji omogućava prijem i slanje UDP poruka. Poruke se primaju na određenom portu i adresi koji je definisan u TR-106 modelu podataka, nakon što se poruka primi ona se razvrstava na STUN ili TR-111 poruku. Nakon toga poruka se smešta u modul za čuvanje poruka i signalizira se da je poruka primljena. Pošto dva modula koriste ovaj komunikacioni sloj, definisane su i dve funkcije koje preuzimaju poruku iz modula za čuvanje poruka. Jedna funkcija je zadužena za preuzimanje STUN poruka, dok je druga zadužena za preuzimanje TR-111 poruka. Druga funkcija modula za rukovanje porukama je slanje STUN zahteva generisanih od strane STUN modula. Slanje poruka se može obaviti sa primarnog ili sekundarnog porta, u zavisnosti od režima rada STUN klijenta. Na slici 3.6 je prikazan princip rada modula za rukovanje porukama i njegova interakcija sa STUN i TR-111 modulima.



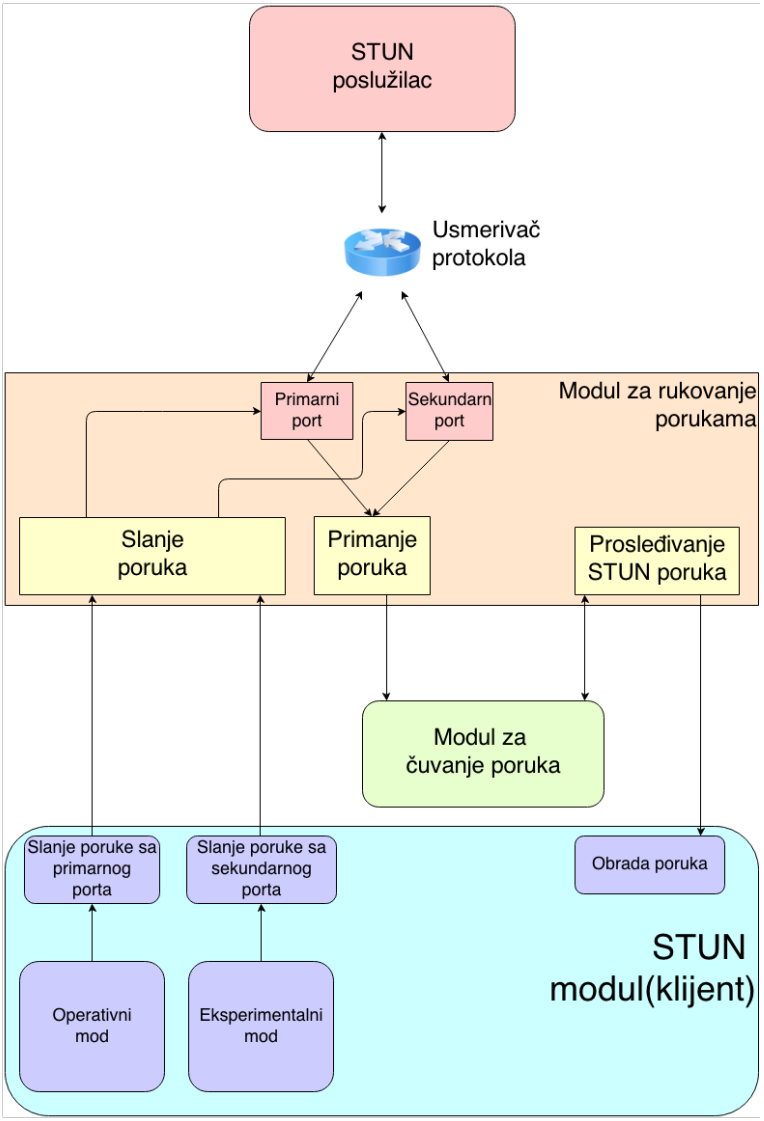
Slika 3. 6 Primanje i slanje modula u TR-111 okruženju

3.3.3 STUN modul

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

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

Upravljački deo je zadužen za pokretanje i zaustavljanje klijenta kao i za upravljanje klijentom tokom rada. Deo zadužen za rukovanje STUN porukama je zadužen za generisanje STUN zahteva i obradu STUN odgovora. Princip rada STUN klijenta je prikazan na slici 3.7. Pošto sam STUN klijent ne može funkcionisati kao samostalna celina, on funkcioniše zajedno u sprezi sa modulom za rukovanje porukama i modulom za čuvanje poruka.



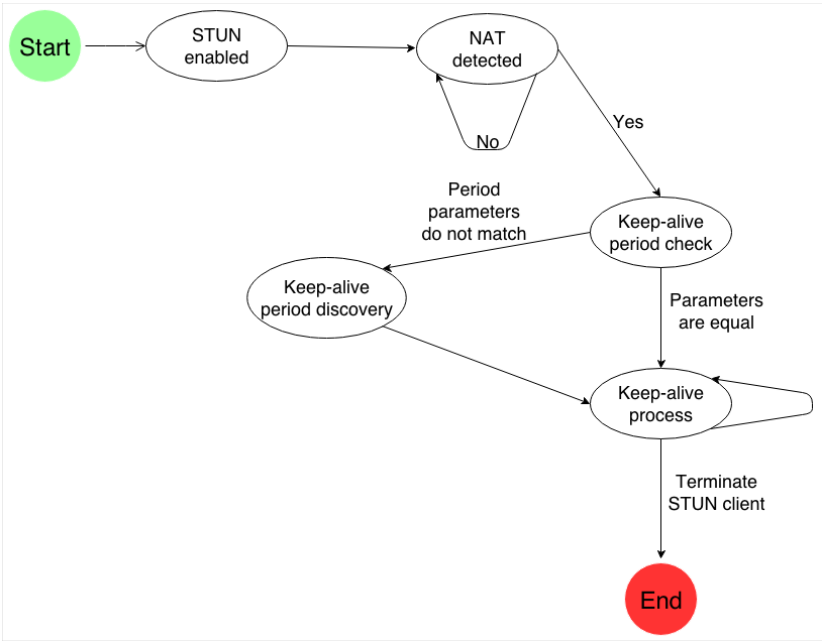
Slika 3. 7 Princip rada STUN modula u sprezi sa modulom za rukovanje i modulom za čuvanje poruka

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

Drugu fazu čini održavanje veze. STUN klijent sa primarnog porta periodično šalje poruke koje za cilj imaju da osveže vezu koju je nastala otkrivanjem translacije adrese i porta. Učestalost slanja tih zahteva je definisana parametrima minimalnog i maksimalnog vremena perioda održavanja veze, koji su definisani u TR-106 modelu podataka. Ukoliko su vrednosti ovih parametara različiti STUN klijent mora pristupiti eksperimentalnom otkrivanju tačne vrednosti perioda održavanja veze. Eksperimentalno otkrivanje parametra održavanja veze se odvija slanjem poruka sa sekundarnog porta. Sam princip izračunavanja je objašnjen u teorijskim osnovama rada.

Tok rada STUN klijenta je definisan pomoću mašine stanja koja je prikazana na slici 3.8 i koja sadrži sledeća stanja:

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



Slika 3. 8 Automat stanja STUN klijenta

Inicijalno stanje vrši proveru da li je inicijalizacija STUN klijent modula uspešno izvršena.

Stanje prisustva STUN poslužioca vrši proveru parametra u TR-106 modelu podataka koji sadrži informaciju o postojanju i operativnosti STUN poslužioca na strani ACS-a.

Pod analizom postavke mreže se podrazumeva provera postojanja usmerivača protokola između ACS poslužioca i krajnjeg uređaja.

Analiza parametara perioda održavanja veze podrazumeva proveru minimalnog i maksimalnog perioda održavanja veze.

Ukoliko se vrednosti razlikuju pristupa se eksperimentalnom određivanju tačne vrednosti parametra perioda održanja veze.

Održavanje veze predstavlja proces periodičnog slanja zahteva STUN poslužiocu sa ciljem da se veza održi. STUN klijent se nalazi u ovom stanju tokom celog trajanja rada libcpe biblioteke.

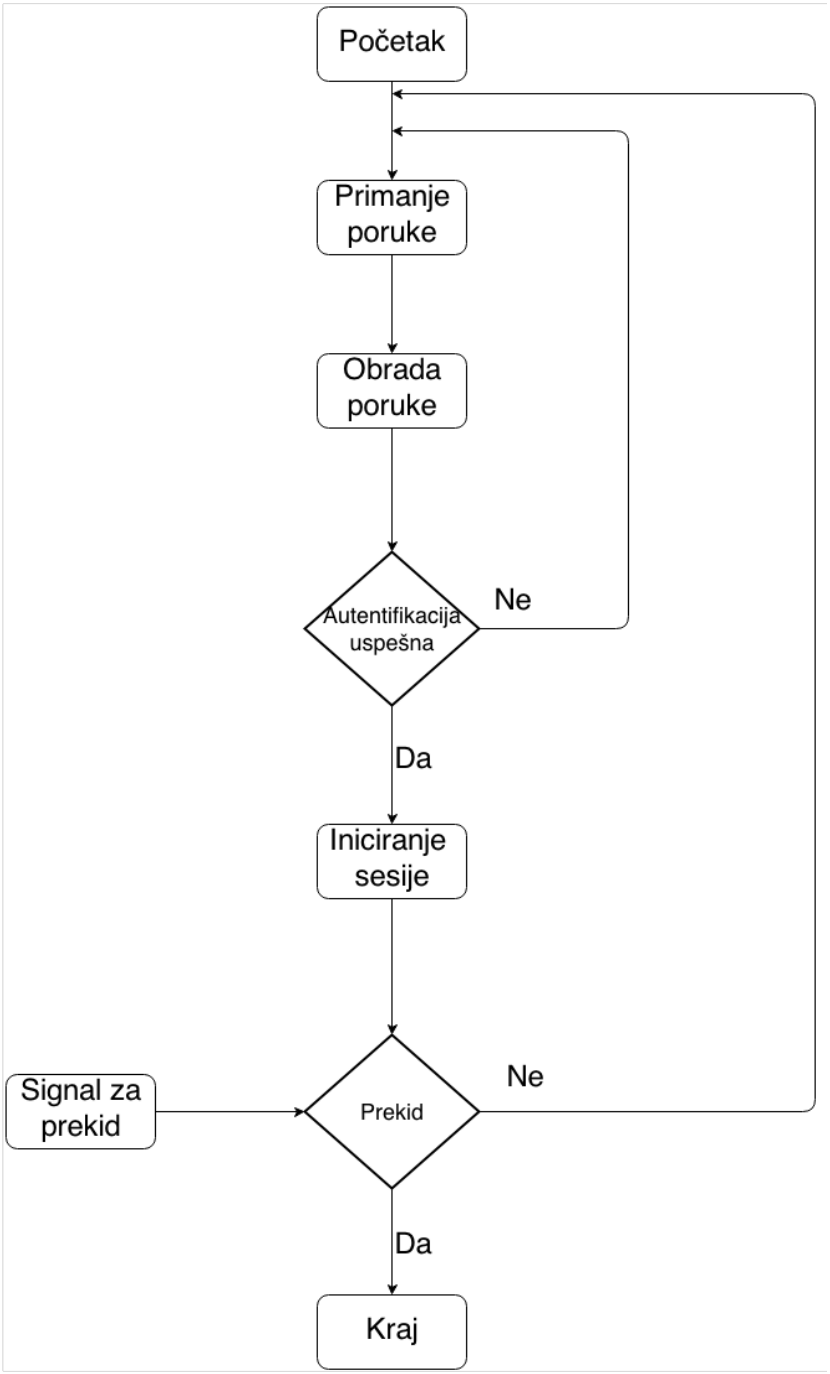
3.3.4 Modul za čuvanje poruka

Modul za čuvanje poruka predstavlja strukturu u formi reda. Njegova uloga je da skladišti poruke namenjene za TR-111 zahtev za povezivanje koje su pristigle na krajnji uređaj. Ovaj modul sadrži sledeće funkcionalnosti:

- Dodavanje poruke u red
- Brisanje poruke iz reda
- Preuzimanje poruke iz reda
- Brisanje sadržaja celog reda
- Brojanje elemenata reda

3.3.5 TR-111 modul (poslužilac)

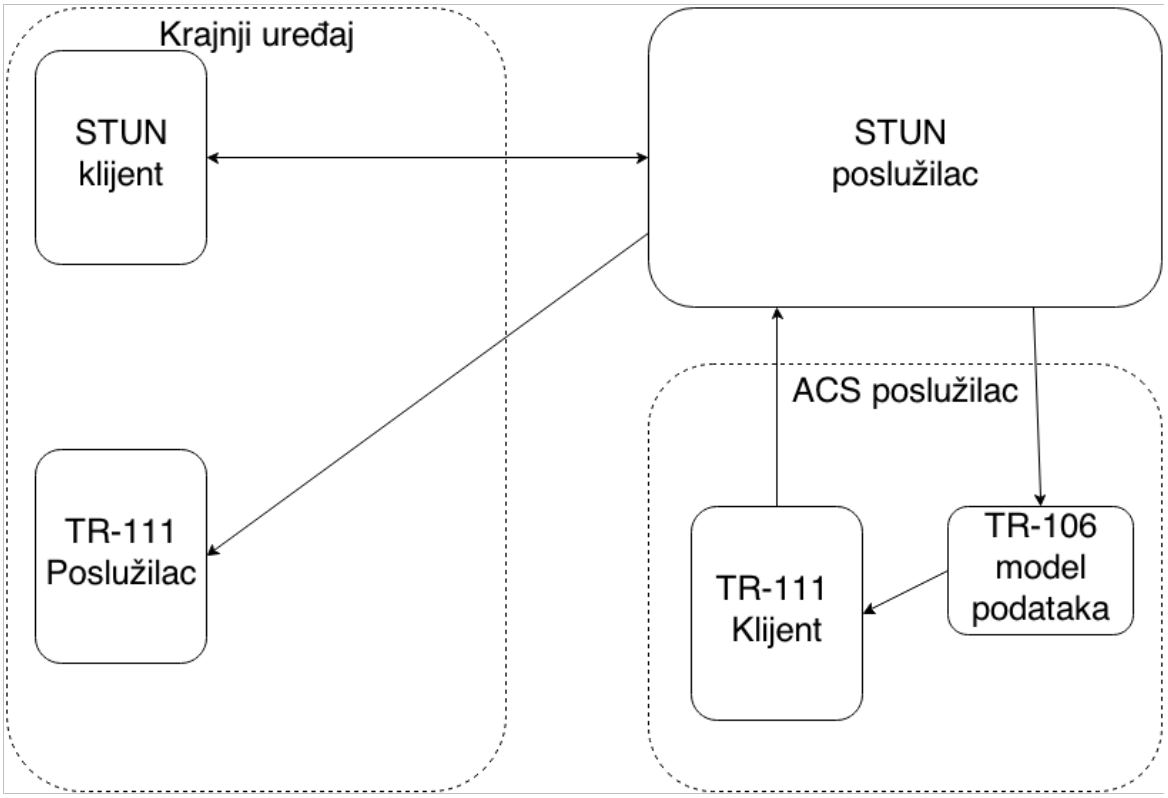
TR-111 modul predstavlja poslužioca koji je zadužen za obradu TR-111 zahteva. UDP zahtevi se preuzimaju iz modula za čuvanje poruka i obrađuju. Vrši se autentifikacija zahteva [5] i ukoliko je ona uspešna, inicira se razmena podataka između krajnjeg uređaja i ACS poslužioca. Na slici 3.9 je prikazan princip rada TR-111 poslužioca.



Slika 3. 9 Princip rada TR-111 poslužioca

3.4 TR-111 klijent i STUN poslužilac na strani ACS-a

TR-111 klijent na strani ACS poslužioca funkcioniše u sprezi sa STUN poslužiocem. Razlog za ovakvim rešenjem leži u činjenici da se STUN poruke i TR-111 poruke primaju i šalju sa istog porta. Otuda činjenica da STUN poslužilac i TR-111 klijent moraju funkcionisati zajedno. STUN poslužilac je u mogućnosti da funkcioniše nezavisno i obavlja svoje funkcije, ali kod TR-111 klijenta to nije slučaj i on ne čini samostalnu celinu. Na slici 3.10 je prikazan princip rada TR-111 klijenta i STUN servera. Svi podaci koji su potrebni za autentifikaciju pomoću TR-111 zahteva za konekciju se na strani ACS-a nalaze se u TR-106 modelu.



Slika 3. 10 Princip rada TR-111 zahteva za konekciju i tok poruka prilikom komunikacije ACS poslužioca i krajnjeg uređaja

STUN poslužilac ima dvojaku ulogu. Primarna uloga poslužioca je da prihvata zahteve od STUN klijenata, obrađuje ih i generiše odgovore. Ova funkcionalnost STUN poslužioca zasniva se na *jstun* programskom rešenju. Pored ove, STUN poslužilac ima još jednu ulogu i ona se ogleda u generisanju TR-111 zahteva na osnovu informacija koje mu dostavi ACS poslužilac. Komunikacija između STUN i ACS poslužioca se odvija pomoću HTTP poruka koje sadrže sve informacije potrebne za generisanje TR-111 zahteva. Na strani STUN poslužioca se nalazi HTTP poslužilac, a na strani ACS poslužioca se nalazi HTTP klijent. Taj klijent zapravo predstavlja deo TR-111 klijenta na strani ACS-a. On prikuplja informacije iz TR-106 modela podataka i

šalje ih STUN poslužiocu koji na osnovu njih generiše TR-111 zahtev i šalje ga. Ovakav pomalo komplikovan način implementacije TR-111 klijenta proističe iz TR-111 standarda koji zahteva da se TR-111 poruke šalju sa iste adrese i porta na kojoj se nalazi STUN poslužilac. Ovakva implementacija deluje nepotrebno kada se STUN poslužilac nalazi na istom uređaju kao i ACS poslužilac, ali problem leži u činjenici da STUN poslužilac ne sme biti iza bilo kakve vrste usmerivača protokola, tj. STUN poslužilac se ne sme naći u privatnom adresnom prostoru. Iz tog razloga se može dogoditi da STUN poslužilac nije na istom uređaju kao i ACS poslužilac, tako da deljenje iste adrese i porta između ACS i STUN poslužioca je nemoguće i zato STUN poslužilac mora biti zadužen za slanje TR-111 zahteva.

4. Programsko rešenje

U ovom poglavlju je opisano programsko rešenje modula koji implementiraju funkcionalnosti zahteva za povezivanje zasnovanog na TR-069 i TR-111 protokolima. Rešenje je realizovano u programskom jeziku C i Java.

Rešenje koje se oslanja na C programski jezik je realizovano na Linux platformi. Programsko rešenje poseduje zavisnosti od biblioteka:

- *pthread*
- *openssl* [11]

Programsko rešenje TR-069 i TR-111 zahteva za konekciju se zasniva na radu 4 niti:

- Nit TR-069 poslužioca
- Komunikaciona nit u TR-111 modulu
- Nit TR-111 poslužioca
- Nit koja pokreće automat stanja STUN klijenta

Niti TR-111 poslužioca i STUN klijenta su u direktnoj zavisnosti od komunikacione niti, međutim ove dve niti ne poseduju nikakvu međusobnu zavisnost u radu. Komunikaciona nit čini srž TR-111 zahteva za povezivanje, bez njenog postojanja moduli STUN klijenta i TR-111 poslužioca bi u potpunosti bili neupotrebljivi. Nit TR-069 poslužioca je u potpunosti nezavisna.

Pored već pomenutih biblioteka, važno je napomenuti da su u TR-069 modulu zahteva za povezivanje na strani krajnjeg uređaja implementirane sledeće dve biblioteke:

- Biblioteka koja rukuje trenutnim stanjem mreže
- Biblioteka koja je odgovorna za generisanje MD5 *hash* suma

Programsko rešenje koje je implementirano na strani ACS poslužioca se oslanja na Java programski jezik i na korišćenje Enterprise JavaBeans (EJB) mehanizma definisanog u okviru

Java EE standarda [12]. Sama celina implementacije programskog rešenja na strani ACS poslužioca se deli na dve celine:

- Moduli koji su u sastavu ACS poslužioca
- STUN poslužilac sa pripadajućim modulima TR-111 poslužioca

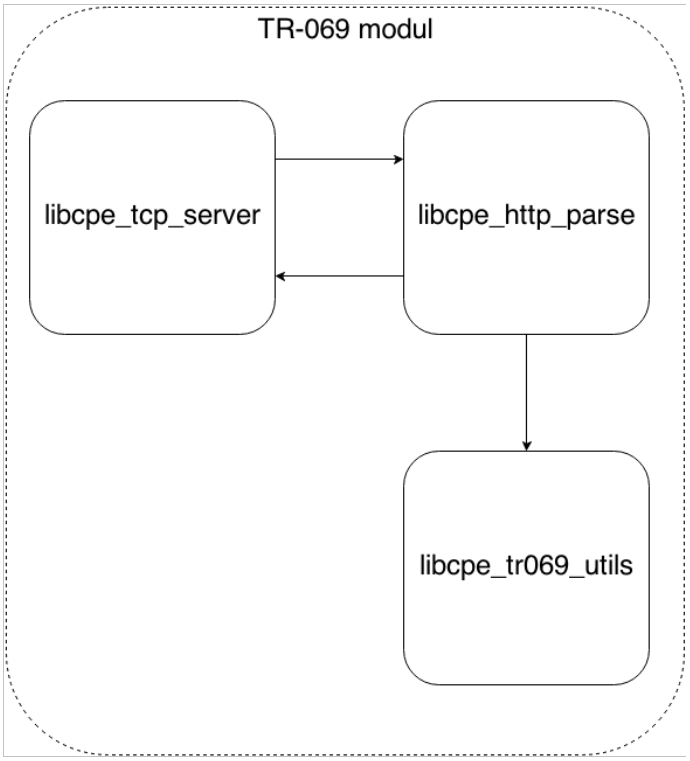
Moduli u sastavu ACS poslužioca su implementirani pomoću Enterprise JavaBeans mehanizma, dok su STUN poslužilac i moduli vezani za TR-111 koji su stacionirani na njemu, implementirani kao zasebna Java aplikacija. STUN poslužilac je realizovan uz pomoć *jstun* biblioteke.

U poglavlju osam dat je detaljan pregled programske sprege TR-069 i TR-111 zahteva za povezivanje.

4.1 TR-069 modul na strani krajnjeg uređaja

TR-069 modul na strani krajnjeg uređaja predstavlja poslužioca čija je uloga da izvrši autentifikaciju klijenata. Na slici 4.1 je prikazan TR-069 modul koji je sačinjen od 3 celine:

- libcpe_tcp_server
- libcpe_http_parse
- libcpe_tr069_utils



Slika 4. 3 TR-069 modul na strani krajnjeg uređaja

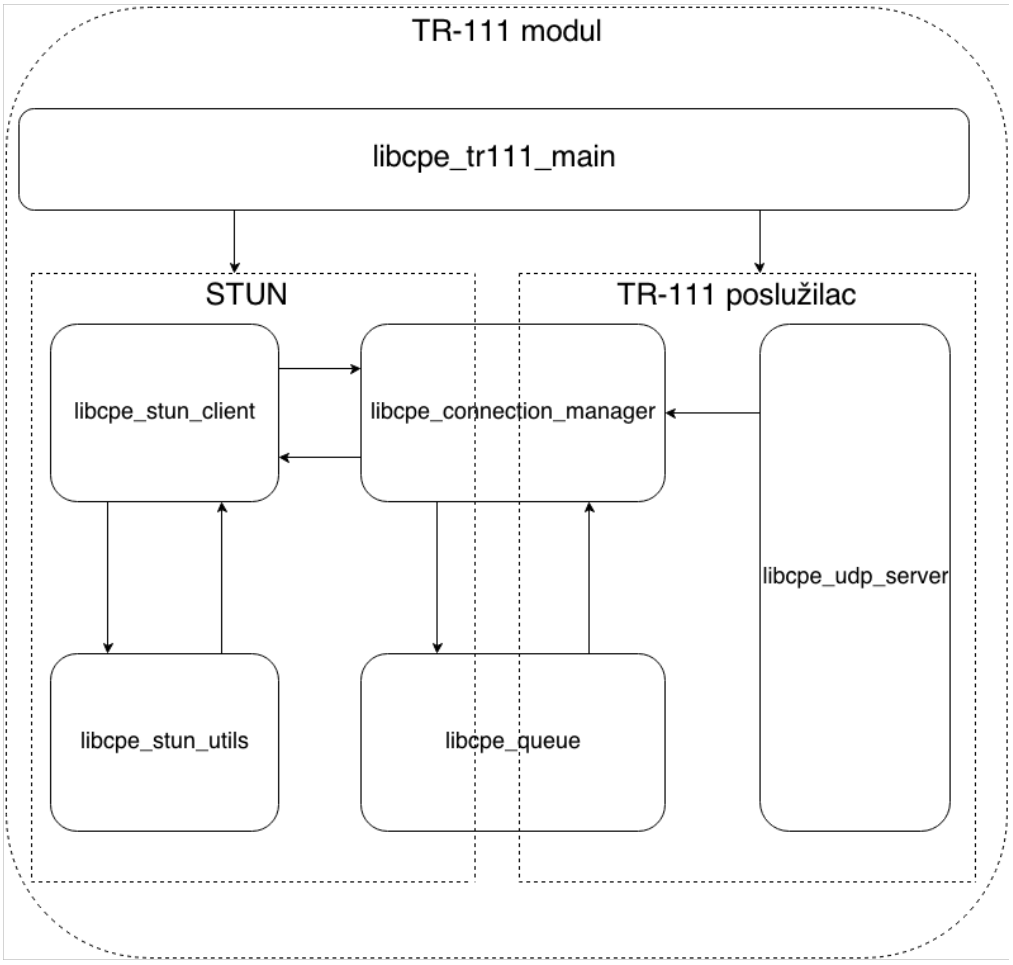
TCP poslužilac predstavlja ključnu celinu modula. Njegova uloga je da prima poruke, prosleđuje ih na obradu i šalje generisane odgovore. Po prijemu poruke ona se prosleđuje

modulu libcpe_http_parse koji vrši obradu poruke i generisanje odgovora. Kao pomoć pri autentifikaciji koriste se funkcije iz modula libcpe_tr_069 utils. Kao celina TR-069 modul je u okviru libcpe biblioteke predstavljen kao statička biblioteka.

4.2 TR-111 modul na strani krajnjeg uređaja

TR-111 modul na strani krajnjeg uređaja predstavlja celinu sačinjenu od TR-111 poslužioca i STUN klijenta. Na slici 4.2 prikazan TR-111 modul koji je sačinjen od 6 celina:

- libcpe_tr1111_main
- libcpe_connection_manager
- libcpe_queue
- libcpe_stun_client
- libcpe_stun_utils
- libcpe_udp_server



Slika 4. 4 TR-111 modul na strani krajnjeg uređaja

Modul libcpe_tr111_main predstavlja programsku spregu celog TR-111 modula koja je isturena prema libcpe biblioteci. Omogućava pokretanje i zaustavljanje STUN klijenta i TR-111 poslužioca.

Kao celina TR-111 modul je u okviru libcpe biblioteke predstavljen kao statička biblioteka.

4.2.1 STUN klijent modul

STUN klijent predstavlja modul koji u sprezi sa STUN poslužiocem vrši otkrivanje translacije adrese i porta, kao i održavanje veze na usmerivaču protokola. Satoji se od 4 celine:

- libcpe_connection_manager
- libcpe_queue
- libcpe_stun_client
- libcpe_stun_utils

Libcpe_connection_manager predstavlja deljeni modul između STUN klijenta i TR-111 poslužioca. Njegova uloga je da:

- prima STUN i TR-111 poruke
- prosleđuje ih modulu za čuvanje poruka
- omogućava preuzimanje poruka iz modula za čuvanje poruka
- omogućava slanje STUN poruka

Libcpe_queue predstavlja modul za čuvanje poruka u formi reda. Njegova uloga je da čuva poruke prosleđene od strane komunikacionog modula sve dok ne budu preuzete i prosleđene TR-111 poslužiocu ili STUN klijentu.

Libcpe_stun_client je odgovoran za otkrivanje translacije adrese i porta. Pored toga ovaj modul vrši i održavanje veze na usmerivaču protokola, kao i eksperimentalno otkrivanje perioda održanja veze.

Libcpe_stun_utils vrši obradu STUN zahteva i generisanje odgovora na date STUN zahteve.

4.2.2 TR-111 poslužilac

TR-111 poslužilac ima ulogu obrade TR-111 zahteva i iniciranja sesija između krajnjeg uređaja i ACS poslužioca. Sastoji se od 3 celine:

- libcpe_connection_manager
- libcpe_queue
- libcpe_udp_server

Libcpe_connection_manager i libcpe_queue predstavljaju deljene module čija je uloga ista kao i u slučaju STUN klijent dela TR-111 modula.

Libcpe_udp_server predstavlja poslužioca čija je uloga da obrađuje TR-111 zahteve i inicira otpočinjanje sesije između krajnjeg uređaja i ACS poslužioca.

4.3 TR-069 modul na strani ACS poslužioca

TR-069 zahtev za povezivanje na strani ACS poslužioca predstavlja klijenta čija je uloga da autentifikacijom sa TR-069 poslužiocem koji se nalazi na krajnjem uređaju izazove otpočinjanje razmene podataka između uređaja i ACS poslužioca. Pored klijentske uloge ovaj modul vrši odlučivanje koji će zahtev za konekciju biti korišćen u zavisnosti od stanja parametara u TR-106 modelu podataka. Detaljan opis funkcionalnosti TR-069 klijenta je prikazan u poglavlju 8.

4.4 TR-111 modul na strani ACS poslužioca

TR-111 zahtev za povezivanje na strani ACS poslužioca predstavlja sistem od dve logičke celine, STUN poslužioca i TR-111 klijenta. STUN poslužilac u saradnji sa STUN klijentom radi na održavanju veze, dok je TR-111 klijent zadužen za autentifikaciju. STUN poslužilac predstavlja zasebnu Java aplikaciju. Implementacija ovog modula se prevashodno zasniva na korišćenju *jstun* biblioteke. To je biblioteka sa STUN poslužiocem koja je proširena tako da udovolji zahtevima TR-111 standarda. STUN aplikacija pored svoje osnovne namene poseduje još jednu. Usled zahteva TR-111 standarda da se sa iste adrese i porta šalji u primaju STUN poruke kao i da se šalju TR-111 zahtevi, STUN aplikacije u sebi sadrži i TR-111 klijenta koji generiše i šalje TR-111 zahteve. Generisanje zahteva se vrši uz pomoć podataka od ACS-a koji se dostavljaju pomoću HTTP poruka STUN aplikaciji.

5. Testiranje i rezultati

Implementacija zahteva za povezivanje je testirana kroz niz testova u kojima su pokrivena sva okruženja u kojima će se koristiti zahtev za povezivanje. Pored testova, izvršeno je i poređenje rezultata rada krajnjeg uređaja sa i bez zahteva za povezivanje.

Platforma korišćena tokom testiranja zahteva za povezivanje na strani krajnjeg uređaja je RK-1000 DBT-2 set-top box baziran na Linux operativnom sistemu. Testna platforma na strani ACS poslužioca je bazirana na i7 Intel procesoru sa 8 GB radne memorije. Na slici 5.1. je prikazan RK-1000 set-top box koji predstavlja testno okruženje u slučaju zahteva za povezivanje na strani krajnjeg uređaja.

Slika 5. 1 RK -1000 set-top box

U tabeli 5.1 su prikazana osnovne vrste testova i okruženje u kojima se testovi izvode. Pored testova, izvršeno je i međusobno poređenje zahteva za povezivanje, kao i poređenje uređaja sa i bez implementiranog zahteva za povezivanje. U okviru svakog testa vršena su testiranja funkcionalnosti implementiranih modula. Testno okruženje možemo podeliti u tri kategorije:

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

Prilikom testiranja korišćeni su različiti mehanizmi za iniciranje prenosa podataka između uređaja i poslužioca. Time je dobijeno međusobno poređenje implementiranih mehanizama iniciranja razmene podataka i već prisutnog metoda razmene podataka. TR-069 uređaji pružaju tri mehanizma za razmenu podataka:

- TR-069 zahtev za konekciju
- TR-111 zahtev za konekciju
- Prenos podataka preko periodičnih javljanja krajnjeg uređaja poslužiocu

Test	Opis
TR-069 zahtev za konekciju	Testiranje obuhvata slučaj kada se poslužilac i krajnji uređaj nalaze u okviru istog adresnog prostora
TR-111 zahtev za konekciju	Testiranje obuhvata slučaj kada se između poslužioca i krajnjeg uređaja nalazi usmerivač protokola
TR-069 i TR-111 zahtevi u prisustvu *proxy*-ja	Testiranje obuhvata slučaj kada je u mreži prisutno *proxy* podešavanje. U okviru ovog testa obuhvaćeni su TR-069 i TR-111 zahtevi za konekcijom

Tabela 5. 1 Testni slučajevi zahteva za povezivanje

5.1 TR-069 zahtev za povezivanje

Kada se uređaji nalaze u istom adresnom prostoru pribegava se korišćenju TR-069 zahteva za povezivanje. U tabeli 5.2 su prikazani rezultati testiranja TR-069 zahteva za konekciju na primeru promene kanala od strane poslužioca.

Naziv	Opis	Vrednost
Promena kanala	Vreme potrebno da se kanal na set-top box-u promeni	~2,8 s
Vreme autentifikacije	Vreme potrebno za izvršavanje TR-069 zahteva za konekciju	~50 ms
Otpočinjanje razmene podataka	Vreme potrebno za izvršavanje TR-069 zahteva za povezivanje i otpočinjanje razmene podataka između krajnjeg uređaja i poslužioca	~250ms

Tabela 5. 2 Rezultati TR-069 zahteva za konekciju

5.2 TR-111 zahtev za povezivanje

Kada se krajnji uređaj nalazi iza usmerivača protokola koristi se TR-111 zahtev za povezivanje. Iako je topologija mreže složenija od konfiguracije za testiranje TR-069 zahteva za konekciju, vreme odziva sistema je manje. Razlog za to je u jednostavnijem principu autentifikacije koja se zasniva na slanju TR-111 poruke i autentifikaciji iste na strani krajnjeg uređaja. Nikakva povratna informacija se ne šalje TR-111 klijentu o stanju autentifikacije. Takav vid jednosmerne komunikacije ostavlja ACS poslužioca bez informacije o uspešnosti zahteva za povezivanje. Druga stvar koja omogućava bolji odziv sistema je manje kompleksna implementacija TR-111 poslužioca u odnosu na TR-069 poslužioca, kao i tip konekcije koja se koristi pri autentifikaciji. U tabeli 5.3 su prikazani rezultati testiranja TR-111 zahteva za konekciju. Ovaj test je sproveden u okruženju kada se između uređaja i poslužioca nalazi jedan usmerivač protokola

Naziv	Opis	Vrednost
Promena kanala	Vreme potrebno da se kanal na set-top box-u promeni	~2.1
Vreme autentifikacije*	Vreme potrebno da se podaci sa ACS-a prenesu do STUN poslužioca koji na osnovu tih podataka generiše TR-111 zahtev	~18 ms
Otpočinjanje razmene podataka	Vreme proteklo od započinjanja TR-111 zahteva za povezivanje do otpočinjanja razmene podataka između krajnjeg uređaja	~150 ms

Tabela 5. 3 Rezultati TR-111 zahteva za konekciju

*Vreme verifikacije nije moguće u potpunosti izmeriti zbog činjenice da poslužilac ne generiše nikakav odgovor na TR-111 zahtev

5.3 Zahtevi za povezivanje kada je u mreži prisutan *proxy*

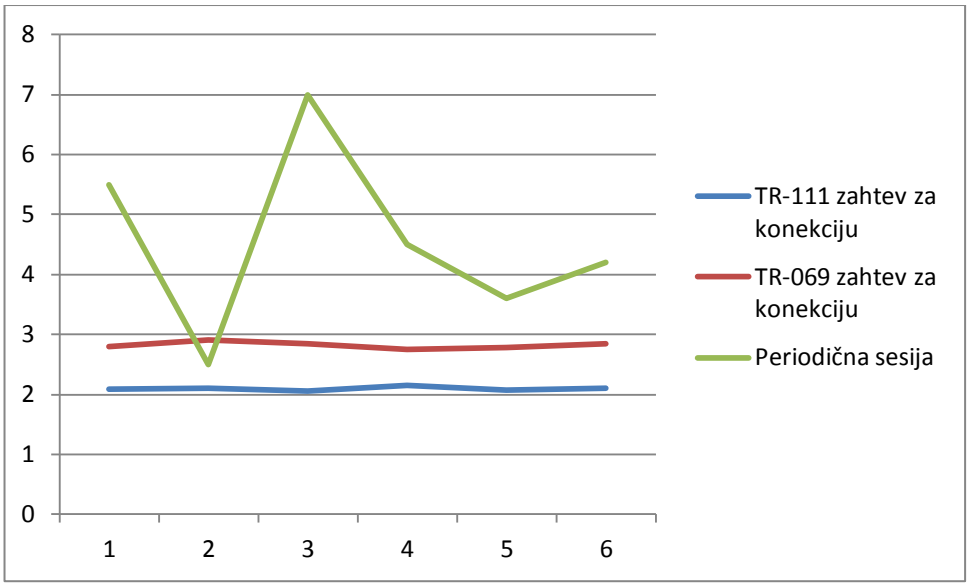
Ukoliko je u mreži prisutan *proxy*, prilikom izvršavanja zahteva za povezivanje nisu uočeni nikakvi problemi. TR-069 zahtevi su izvršeni bez ikakvih poteškoća. Situacija sa TR-111 zahtevom za konekciju je ista kao i u slučaju TR-069 zahteva za konekciju. STUN i TR-111 mehanizmi rade bez ikakvih problema. Testni slučajevi za okruženje u kom je prisutan *proxy* su identični kao i za slučaj kada *proxy* nije prisutan u mreži.

5.4 Pregled rezultata

Pored TR-069 i TR-111 zahteva za povezivanje krajnji uređaj prenosi podatke poslužiocu preko periodičnih sesija (eng. inform). U tabeli 5.4 i na slici 5.2 su prikazani rezultati merenja promene kanala kada se koriste zahtevi za konekciju i kada se koriste periodične sesije. Osim dužeg vremenskog odziva sistema, uređaji bez implementiranog zahteva za konekciju ispoljavaju nekonzistentnost u odzivu sistema. Vreme odziva zavisi od vremena proteklog od poslednjeg javljanja uređaja poslužiocu. Za potrebe testa periodičnih sesija, uzeto je da se uređaj javlja poslužiocu na svakih pet sekundi. Kao što se sa tabele 5.4 može videti, promena kanala putem periodičnih sesija izaziva velike oscilacije u radu sistema.

Prenos podataka	Test	Vrednost
TR-069 zahtev za povezivanje	Vreme potrebno da se kanal na set-top box-u promeni	~2.8 s
TR-111 zahtev za povezivanje	Vreme potrebno da se kanal na set-top box-u promeni	~2.1 s
Periodične sesije (period od 5 sekundi između javljanja)	Vreme potrebno da se kanal na set-top box-u promeni	~2.5s - ~7.5s

Tabela 5. 4 Poređenje rezultata prenosa podataka između uređaja i poslužioca



Slika 5. 2 Poređene rezultata odziva sistema

Pored standardnih testova vezanih za TR-111, izvršeni su i testovi otkrivanja translacije adrese i porta, i održavanja veze tokom perioda vremena od 48 sati. Testiranjem TR-111 zahteva ustanovljeno je da *proxy* nema uticaj na funkcionisanje STUN mehanizma. Sa druge strane, ukoliko zaštitni zid mreže blokira slanje UDP poruka, tada STUN mehanizam neće raditi. U tabeli 5.5 su prikazani rezultati STUN testova.

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

Tabela 5. 5Rezultati testiranja STUN modula

Pregled svih izvršenih testova za TR-069 i TR-111 zahteve za povezivanje su prikazani u tabeli 5.6. Ovi rezultati pokazuju da bez obzira na okruženje u kom se krajnji uređaj nalazi, on će uvek biti dostupan ACS poslužiocu.

Test	TR-069 zahtev za konekciju	TR-111 zahtev za konekciju
Zahtev kada su uređaj i poslužilac u istom adresnom prostoru	✓	×
Zahtev kada se uređaj nalazi iza usmerivača protokola	×	✓
Zahtev kada zaštitni zid blokira UDP poruke	NA	×
Zahtev kada su uređaj i poslužilac u istom adresnom prostoru - *proxy*	✓	×
Zahtev kada se uređaj nalazi iza usmerivača protokola - *proxy*	×	✓

Tabela 5. 6 Pregled urađenih testova za TR-069 i TR-111 protokole

6. Zaključak

U ovom radu je realizovana podrška zahtevu za povezivanjem zasnovanog na TR-069 protokolu. Zahtev je bio da se omogući momentalni odziv uređaja na pobudu sa poslužioca. Osim toga zahtev je bio da se omogući dostupnost uređaja bez obzira na konfiguraciju mreže u kojoj se krajnji uređaj nalazi. Implementacijom TR-106 i TR-111 zahteva za konekciju ispunjeni su svi zahtevi u vezi odziva i dostupnosti uređaja. Postojanjem zahteva za konekciju eliminisana je nepredvidljivost pri odzivu sistema koja je bila izražena pri prenosu podataka isključivo preko periodičnih razmena podataka koje su inicirane jedino od strane krajnjeg uređaja. Svi podaci koje krajnji uređaj poseduje su dostupni poslužiocu na zahtev uz značajno manje kašnjenje od perioda javljanja.

Rešenje je realizovano upotrebom programskih jezika C i Java. Rešenje poseduje malu zavisnost od eksternih biblioteka i iz tog razloga je portabilno i prilagodljivo svakoj Linux platformi. Rešenje ne narušava koncept već postojećeg rešenja klijenta za krajnji uređaj i ACS poslužioca.

Testiranjem su potvrđene sve funkcionalnosti TR-069 i TR-111 zahteva za konekciju. Opsežnim testiranjem je utvrđeno da bez obzira na stanje mreže tj. okruženje u kome se nalazi krajnji uređaj. Testiranje je podeljeno u dve celine, testiranje TR-069 zahteva za konekciju i testiranje TR-111 zahteva za konekciju. U okviru svake celine testiranja definisana su dva stanja mreže pod kojima se testira dati zahtev za konekciju. Uspešnim testiranjem za različita stanja i postavke mreže potvrđena je robusnost rešenja. Dostupnost uređaja prestaje jedino kada zaštitni zid onemogući slanje UDP poruka.

Pravci daljeg razvoja se ogledaju u zameni TR-069 zahteva u potpunosti sa TR-111 zahtevom za konekciju koji se po svim parametrima pokazao robusnijim i efikasnijim. Osim toga jedan od mogućih pravaca bi predstavljao razvoj sopstvene biblioteke STUN poslužioca čija bi isključiva namena bila funkcionisanje u kontekstu TR-111 protokola. Time bi se postigla veća

efikasnost STUN poslužioca jer bi se eliminisale STUN funkcionalnosti koje se ne koriste u TR-111 protokolu. Samim poboljšanjem performansi STUN mehanizma postigao bi se bolji odziv celog sistema.

7. Literatura

[1] TR-069 CPE WAN Management Protocol, http://www.broadband-forum.org/technical/download/TR-069_Amendment-4.pdf

[2] TR-106 Data Model Template for TR-069-Enabled Devices, http://www.broadband-forum.org/technical/download/TR-106_Amendment-3.pdf

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

[4] HTTP Authentication: Basic and Digest Access Authentication, http://www.ietf.org/rfc/rfc2617.txt

[5] TR-111 Applying TR-069 to Remote Management of Home Networking Devices, http://www.broadband-forum.org/technical/download/TR-111.pdf

[6] STUN – Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), http://www.ietf.org/rfc/rfc3489.txt

[7] Hypertext Transfer protocol – HTTP/1.1, http://www.ietf.org/rfc/rfc2616.txt

[8] Uniform Resource Identifier (URI), http://www.ietf.org/rfc/rfc3986.txt

[9] HMAC: Keyed-Hashing for Message Authentication, http://tools.ietf.org/html/rfc2104

[10] Nemanja Ignjatov, “Realizacija komunikacionog sloja TR-069 klijenta za STB uređaj”, http://www.rt-rk.uns.ac.rs/media/download/bsc_radovi/e13048_nemanja_ignjatov.pdf

[11] OpenSSL: The Open Source toolkit for SSL/TLS, http://www.openssl.org/

[12] “Building Java Enterprise Systems with J2EE”, Paul Perrone, S.R.R. Chaganati, 2000

8. Dodatak

8.1 Programska sprega TR-069 modula na strani krajnjeg uređaja

U okviru ovog modula realizovane su osnovne funkcije TR-069 poslužioca koji predstavlja deo TR-069 zahteva za konekciju na krajnjem uređaju.

8.1.1 libcpe_tcp_server.h

- **CR_ERR_T cr_tcp_create_server_ctx(void* ctx, CRCALLBACK user_callback, NCCALLBACK net_notify_callback);**
  - Opis: Inicijalizacija TR-069 poslužioca i pribavljanje svih podataka neophodnih za autentifikaciju korisnika.
  - Parametri:
    - ctx – Kontekst CPE klijenta u kome se nalazi struktura koja sadrži podatke o TR-069 poslužiocu, kao i strukture koje sadrže podatke iz TR-106 modela podataka.
    - user_callback – funkcija koja se poziva prilikom završene autentifikacije
    - net_notify_callback – funkcija koja se poziva pri promeni stanja mreže
  - Povratna vrednost: Indikacija uspešnosti inicijalizacije
    - CR_TRUE – poslužilac nije inicijalizovan usled grešaka prilikom inicijalizacije
    - CR_FALSE – poslužilac je uspešno inicijalizovan
- **CR_ERR_T cr_tcp_start_server(void *ctx);**

- Opis: Pokretanje TR-069 poslužioca i samim tim stavljanje u rad TR-069 zahteva za povezivanje
- Parametri: ctx – Kontekst CPE klijenta
- Povratna vrednost: Indikacija uspešnosti pokretanja TR-069 poslužioca
  - CR_TRUE – pokretanje TR-069 poslužioca obustavljeno usled grešaka
  - CR_FALSE – TR-069 poslužilac uspešno pokrenut

- **CR_ERR_T cr_tcp_stop_server(void *ctx);**
  - Opis: Funkcija koja je odgovorna za zaustavljanje TR-069 poslužioca
  - Parametri: ctx – Kontekst CPE klijenta
  - Povratna vrednost: Indikacija uspešnosti zaustavljanja TR-069 poslužioca
    - CR_TRUE – TR-069 poslužilac nije uspešno zaustavljen usled grešaka
    - CR_FALSE – TR-069 poslužilac je uspešno generisan
- **CR_ERR_T cr_tcp_delete_server(void *ctx);**
  - Opis: Funkcija koja vrši deinicijalizaciju TR-069 poslužioca
  - Parametri: ctx – Kontekst CPE klijenta
  - Povratna vrednost: Indikacija uspešnosti deinicijalizacije TR-069 poslužioca
    - CR_TRUE – Deinicijalizacija TR-069 poslužioca nije uspela usled grešaka
    - CR_FALSE – Deinicijalizacija TR-069 poslužioca uspešno izvršena
- **void cr_tcp_generate_hex_string(char *string, unsigned string_length);**
  - Opis: Funkcija koja vrši generisanje stringa proizvoljne dužine sa nasumično odabranim heksadecimalnim vrednostima
  - Parametri:
    - string – promenljiva u koju će generisani string heksadecimalnih vrednosti biti smešten
    - string_length – željena dužina stringa koji će se generisati

8.1.2 libcpe_http_parse.h

- **HTTP_ERR_T http_parse_request (void *ctx , char *http_req_message,char *http_resp_message);**
  - Opis: Funkcija vrši analiziranje HTTP *digest* zahteva i generisanje odgovora na dati zahtev

- Parametri:
  - ctx – Kontekst CPE klijenta koji u kome se između ostalog sadrže i podaci vezani za proces autentifikacije
  - http_req_message - HTTP zahtev koji je potrebno obraditi
  - http_resp_message – HTTP odgovor koji se generiše na osnovu analize HTTP zahteva
- Povratna vrednost: Indikacija uspešnosti izvršenog analiziranja
  - HTTP_TRUE – Prilikom analiziranja HTTP zahteva došlo je do grešaka
  - HTTP_FALSE – HTTP zahtev je uspešno obrađen

- **HTTP_ERR_T http_generate_ok_code (char *message);**
  - Opis: Funkcija koja generiše HTTP odgovor u kome se sadrži poruka da je autentifikacija uspešno izvršena
  - Parametri: message – Generisani HTTP odogovor
  - Povratna vrednost: Indikacija o uspešnosti generisanja odgovora
    - HTTP_TRUE – Odgovor nije bilo moguće generisati usled grešaka
    - HTTP_FALSE – Odgovor je uspešno generisan
- **HTTP_ERR_T http_generate_unauthorized_code (void *ctx, char *message);**
  - Opis: Funkcija koja generiše HTTP odgovor sa porukom da je korisniku pristup odbijen
  - Parametri:
    - ctx – Kontekst struktura CPE klijenta
    - message – Generisani HTTP odogovor
  - Povratna vrednost: Indikacija o uspešnosti generisanja odgovora
    - HTTP_TRUE – Odgovor nije bilo moguće generisati usled grešaka
    - HTTP_FALSE – Odgovor je uspešno generisan
- **HTTP_ERR_T http_generate_not_found_code (char *message);**
  - Opis: Funkcija koja generiše HTTP odgovor sa porukom da je URI deo HTTP zahteva neodgovarajući
  - Parametri: message – Generisani HTTP odgovor
  - Povratna vrednost: Indikacija o uspešnosti generisanja odgovora
    - HTTP_TRUE – Odogovor nije bilo moguće generisati usled grešaka
    - HTTP_FALSE – Odgovor je uspešno generisan
- **HTTP_ERR_T http_gen_serv_unavailable_code (char *message);**
  - Opis: Funkcija koja generiše HTTP odgovor sa porukom da je servis nedostupan.

- Parametri: message – Generisani HTTP odgovor
- Povratna vrednost: Indikacija o uspešnosti generisanja odgovora
  - HTTP_TRUE – Odgovor nije bilo moguće generisati usled grešaka
  - HTTP_FALSE – Odgovor je uspešno generisan

8.1.3 libcpe_tr069_utils.h

- **void net_notif_callback(void* ctx, int status);**
  - Opis: Funkcija koja se poziva od strane biblioteke za nadgledanje mreže ukoliko dođe do promene stanja mreže
  - Parametri:
    - ctx – Korisnički podaci koji se na ovaj način prilikom pozivanja ove funkcije prosleđuju biblioteci za nadgledanje stanja mreže. U slučaju TR-069 zahteva za konekciju ova struktura predstavlja skup podataka o TR-069 poslužiocu
    - status – Status mreže u trenutku pozivanja funkcije
- **void auth_callback(, void* ctx , int auth_valid);**
  - Opis: Funkcija koja se poziva prilikom završene autentifikacije
  - Parametri:
    - ctx – Kontekst CPE klijenta
    - auth_valid – Rezultat autentifikacije
- **UTIL_ERR_T parse_config_url (const char *config_url,char *address, char *port, char *url);**
  - Opis: Funkcija koja parsira sadržaj polja ConnectionRequestURL iz TR-106 modela podataka. Ona iz tog stringa izvlači adresu, port i pristupni URI. Ti podaci se koriste pri kreiranju TR-069 poslužioca i pri autentifikaciji korisnika
  - Parametri:
    - config_url – string iz kojeg se izvlače podaci parsiranjem
    - address – IP adresa dobijena izvlačenjem podataka iz config_url-a. Predstavlja adresu TR-069 poslužioca
    - port – Port dobijen izvlačenjem podataka iz config_url-a. Predstavlja port TR-069 poslužioca
    - url – URL vrednost se dobija izvlačenjem podataka iz config_url-a. Vrednost ovog parametra se koristi pri autentifikaciji.
  - Povratna vrednost: Indikacija uspešnosti parsiranja
    - UTIL_TRUE – Parsiranje nije završeno usled grešaka

- UTIL_FALSE – Parsiranje je uspešno izvršeno

8.2 Programska sprega TR-111 modula na strani krajnjeg uređaja

U ovom modulu realizovane su sve osnovne funkcionalnosti vezane za TR-111 zahtev za povezivanje na strani krajnjeg uređaja.

8.2.1 libcpe_tr111_main.h

- **TRMAIN_ERR_T trmain_init (void *ctx, NCCALLBACK net_notify_callback);**
  - Opis: Funkcija koja vrši inicijalizaciju celog TR-111 modula. Unutar ove funkcije vrši se inicijalizacija STUN klijenta, TR-111 poslužioca i komunikacionog modula
  - Parametri:
    - ctx – Kontekst CPE klijenta u kome se između ostalog nalaze i svi podaci i strukture vezane za TR-111 zahtev za povezivanje
    - net_notify_callback – funkcija koja se poziva prilikom promene stanja mreže
  - Povratna vrednost: Indikacija uspešnosti inicijalizacije TR-111 modula
    - TRMAIN_TRUE – Inicijalizacija neuspela usled grešaka
    - TRMAIN_FALSE – Inicijalizacija uspešna
- **TRMAIN_ERR_T trmain_start (void *ctx);**
  - Opis: Funkcija koja vrši pokretanje TR-111 modula. U sklopu ove funkcije vrši se pokretanje STUN klijenta, TR-111 poslužioca i komunikacionog modula
  - Parametri: ctx – Kontekst CPE klijenta
  - Povratna vrednost: Indikacija uspešnosti pokretanja TR-111 modula
    - TRMAIN_TRUE – Pokretanje TR-111 modula neuspešno usled grešaka
    - TRMAIN_FALSE – Pokretanje TR-111 modula uspešno izvršeno
- **TRMAIN_ERR_T trmain_stop (void *ctx);**
  - Opis: Funkcija koja obavlja zaustavljanje TR-111 modula. Ova funkcija vrši zaustavljanje STUN klijenta, TR-111 poslužioca i komunikacionog modula
  - Parametri: ctx – Kontekst CPE klijenta
  - Povratna vrednost: Indikacija uspešnosti zaustavljanja TR-111 modula

        - TRMAIN_TRUE – Zaustavljanje TR-111 modula neuspelo usled grešaka
        - TRMAIN_FALSE – Zaustavljanje TR-111 modula uspešno izvršena
- **TRMAIN_ERR_T trmain_dest (void *ctx);**
    - Opis: Funkcija koja obavlja deinicijalizaciju TR-111 modula. Unutar ove funkcije poziva se deinicijalizacija
    - Parametri: ctx – Kontekst CPE klijenta
    - Povratna vrednost: Indikacija uspešnosti deinicijalizacije TR-111 modula
        - TRMAIN_TRUE - Deinicijalizacija TR-111 modula neuspela usled grešaka
        - TRMAIN_FALSE – Deinicijalizacija TR-111 modula je uspešno izvršena
- **void net_notif_callback_tr111(void* ctx, int status);**
    - Opis: Funkcija koja se poziva prilikom promene stanja mreže
    - Parametri:
        - ctx – Kontekst CPE klijenta
        - satus – status mreže

8.2.2 libcpe_connection_manager.h

- **CM_ERR_T cm_init (void *ctx);**
    - Opis: Funkcija koja inicijalizuje komunikacioni modul koji omogućava primanje i slanje UDP poruka. Osim primanja i slanja ovaj modul ima mogućnost razvrstavanja poruka na STUN i TR-111 poruke
    - Parametri: ctx – Kontekst CPE klijenta
    - Povratna vrednost: Indikacija uspešnosti inicijalizacije komunikacionog modula
        - CM_TRUE – Komunikacioni modul nije inicijalizovan usled grešaka
        - CM_FALSE – Komunikacioni modul je uspešno inicijalizovan
- **CM_ERR_T cm_start (void *ctx);**
    - Opis: Funkcija koja je zadužena za pokretanje komunikacionog modula
    - Parametri: ctx – Kontekst CPE klijenta
    - Povratna vrednost: Indikacija uspešnosti pokretanja komunikacionog modula
        - CM_TRUE – Komunikacioni modul nije pokrenut usled grešaka

        - CM_FALSE – Komunikacioni modul je uspešno pokrenut
- **CM_ERR_T cm_stop (void *ctx);**
    - Opis: Funkcija čija je uloga zaustavljanje komunikacionog modula
    - Parametri: ctx – Kontekst CPE klijenta
    - Povratna vrednost: Indikacija uspešnosti zaustavljanja komunikacionog modula
        - CM_TRUE - Zaustavljanje komunikacionog modula nije uspešno izvršeno usled grešaka
        - CM_FALSE – Zaustavljanje komunikacionog modula uspešno izvršeno
- **CM_ERR_T cm_destroy (void *ctx);**
    - Opis: Funkcija koja je zadužena za deinicijalizaiju komunikacionog modula
    - Parametri: ctx – Kontekst CPE klijenta
    - Povratna vrednost: Indikacija o uspešnosti deinicijalizacije klijenta
        - CM_TRUE – Deinicijalizacije komunikacionog modula neuspešna usled grešaka
        - CM_FALSE – Deinicijalizacija modula uspešno izvršena
- **CM_ERR_T cm_send (void *connection_man_ctx, uint8_t *buffer,size_t buffer_size, cm_status status);**
    - Opis: Funkcija vrši slanje UDP poruka na određenu adresu. Koristi se pri slanju STUN zahteva. Pošto se pri TR-111 autentifikaciji ne šalje nikakva povratna informacija o statusu trenutne autentifikacije, ovu funkciju jedino koristi STUN modul
    - Parametri:
        - connection_man_ctx – struktura komunikacionog modula koja sadrži sve informacije potrebne za funkcionisanje komunikacionog modula
        - message – poruka koju je potrebno poslati
        - message_size – dužina poruke
        - status – odnosi se na tip STUN poruke koju je potrebno poslati. U zavisnosti od tipa poruke ona će biti poslata sa primarne ili sekundarne adrese
    - Povratna vrednost: Indikacija o uspešnosti slanja poruke
        - CM_TRUE – Slanje poruke nije uspelo usled grešaka u toku slanja
        - CM_FALSE – Slanje poruke uspešno izvršeno

- **CM_ERR_T cm_recv_stun (void *ctx, uint8_t *message, cm_status *status, CM_BOOL_T *res, timespec_cm timeout);**
    - Opis: Funkcija koja vrši primanje STUN poruka. Tačnije uloga ove funkcije nije da primi poruku, već da je preuzme iz reda STUN poruka koji se automatski popunjava pri dospeću nove STUN poruke
    - Parametri:
        - ctx – Kontekst CPE klijenta
        - message – poruka koja je primljena
        - status – status poruka predstavlja tip poruke. Ova promenljiva je od važnosti iz razloga što se u STUN modulu mogu odvijati dva procesa i otuda pomoću tipa poruke STUN modul odlučuje kom procesu je namenjena poruka
        - res – indikacija da li je poruka primljena/preuzeta
        - timeout – vreme posle koga se obustavlja primanje poruke. Ukoliko u datom vremenskom roku se ne primi STUN poruka proces primanja poruke se obustavlja
    - Povratna vrednost:
        - CM_TRUE – Primanje poruke obustavljeno usled greške
        - CM_FALSE – Funkcija primanja poruke je izvršena bez grešaka
- **CM_ERR_T cm_recv_tr111 (void *ctx, uint8_t *message, CM_BOOL_T *res, timespec_cm timeout);**
    - Opis: Funkcija vrši primanje TR-111 poruka. Kao i u slučaju cm_recv_stun ova funkcija zapravo vrši preuzimanje poruka iz reda koji se automatski popunjava sa stizanjem svake nove TR-111 poruke.
    - Parametri:
        - ctx – Kontekst CPE klijenta
        - message – poruka koja je primljena/preuzeta
        - res - indikacija da li je poruka primljena
        - timeout - vreme posle koga se obustavlja primanje poruke. Ukoliko u datom vremenskom roku se ne primi TR-111 poruka proces primanja poruke se obustavlja
    - Povratna vrednost:
        - CM_TRUE – Primanje poruke obustavljeno usled greške
        - CM_FALSE – Funkcija primanja poruke je izvršena bez grešaka

8.2.3 libcpe_stun_client.h

- **STUN_ERR_T libcpe_stun_init (void *ctx);**
  - Opis: Funkcija koja vrši inicijalizaciju STUN klijenta
  - Parametri: ctx – Kontekst struktura CPE klijenta koja između ostalog sadrži i parametre vezane za STUN klijent modul
  - Povratna vrednost: Indikacija uspešnosti inicijalizacije STUN klijenta
    - STUN_TRUE – Incijalizacija STUN klijenta nije uspela usled grešaka prilikom inicijalizacije
    - STUN_FALSE – Inicijalizacija STUN klijenta uspešno izvršena
- **STUN_ERR_T libcpe_stun_start (void *ctx);**
  - Opis: Funkcija vrši pokretanje STUN klijenta. Pokretanjem klijenta pokreće se i automat stanja čija je uloga da reguliše rad STUN klijenta i da sinhronizuje sve procese koji se odvijaju u toku njegovog rada
  - Parametri: ctx – Kontekst CPE klijenta
  - Povratna vrednost: Indikacija uspešnosti pokretanja STUN klijenta
    - STUN_TRUE – Pokretanje klijenta obustavljeno usled grešaka
    - STUN_FALSE – Pokretanje klijenta uspešno izvršeno
- **STUN_ERR_T libcpe_stun_stop (void *ctx);**
  - Opis: Funkcija koja je zadužena za zaustavljanje STUN klijenta
  - Parametri: ctx- Kontekst CPE klijenta
  - Povratna vrednost: Indikacija uspešnosti zaustavljanja STUN klijenta
    - STUN_TRUE – Zaustavljanje klijenta je neuspešno izvršeno usled grešaka
    - STUN_FALSE – Zaustavljanje klijenta je uspešno izvršeno
- **STUN_ERR_T libcpe_stun_destroy (void *ctx);**
  - Opis: Funckija koja vrši deinicijalizaciju STUN klijenta
  - Parametri: ctx – Kontekst struktura CPE klijenta
  - Povratna vrednost: Indikacija uspešnosti deinicijalizacije STUN klijenta
    - STUN_TRUE – Deinicijalizacija neuspešna usled grešaka
    - STUN_FALSE – Deinicijalizacija uspešno izvršena

8.2.4 libcpe_stun_utils.h

- **STUN_UTILS_ERR_T libcpe_stun_make_request (void *ctx);**
  - Opis: Funkcija je zadužena za generisanje STUN zahteva. Zbog činjenice da STUN klijent može imati dva stanja, održavanje veze i eksperimentalno

određivanje perioda veze, pa se iz tog razloga mogu generisati dve vrste poruka. libcpe_stun_make_request na osnovu trenutnog stanja u kom se STUN klijent nalazi zaključuje kakav tip zahteva je neophodno generisati

- Parametri: ctx – Kontekst struktura CPE klijenta
- Povratna vrednost: Indikacija uspešnosti generisanja STUN zahteva
  - STUN_UTILS_TRUE – Generisanje poruke neuspelo usled grešaka
  - STUN_UTILS_FALSE – Generisanje poruke je uspešno izvršeno

- **STUN_UTILS_ERR_T libcpe_stun_read_message (void *ctx, char *mapped_addr, uint16_t *mapped_port);**
  - Opis: Funkcija koja je zadužena za čitanje i analiziranje STUN odgovora
  - Parametri:
    - ctx – Kontekst struktura CPE klijenta
    - mapped_addr - adresa koja je otkrivena na usmerivaču protokola preko koje CPE klijent prima poruke
    - mapped_port – port koji je otkriven na usmerivaču protokola preko kojeg CPE klijent prima poruke
  - Povratna vrednost: Indikacija uspešnosti analiziranja STUN odgovora
    - STUN_UTILS_TRUE – Obrada primljenog STUN odgovora je neuspešna usled grešaka koje su nastale u toku analiziranja poruke
    - STUN_UTILS_FALSE – Obrada prispele STUN poruke je uspešno izvršena
- **void libcpe_stun_get_message_size (uint8_t *message, unsigned *message_size);**
  - Opis: Funkcija koja određuje dužinu primljene STUN poruke
  - Parametri:
    - message – STUN poruka čiju je dužinu potrebno odrediti
    - message_size – određena dužina STUN poruke

8.2.5 libcpe_udp_server.h

- **UDP_ERR_T libcpe_create_udp_server (void* ctx);**
  - Opis: Funkcija koja je zadužena za inicijalizaciju TR-111 poslužioca
  - Prametri: ctx – Konteks struktura CPE klijenta
  - Povratna vrednost: Indikacija uspešnosti inicijalizacije TR-111 poslužioca
    - UDP_TRUE – Inicijalizacija TR-111 poslužioca neuspešna usled grešaka tokom inicijalizacije

        - UDP_FALSE – Inicijalizacija TR-111 poslužioca je uspešno izvršena
- **UDP_ERR_T libcpe_start_udp_server (void *ctx);**
    - Opis: Funkcija je odgovorna za pokretanje TR-111 poslužioca.
    - Parametri: ctx – Kontekst struktura CPE klijenta
    - Povratna vrednost: Indikacija uspešnosti pokretanja TR-111 poslužioca
        - UDP_TRUE – Pokretanje TR-111 poslužioca nije uspelo usled grešaka u procesu pokretanja poslužioca
        - UDP_FALSE – Pokretanje TR-111 poslužioca uspešno izvršeno
- **UDP_ERR_T libcpe_stop_udp_server (void *ctx);**
    - Opis: Funkcija koja je zadužena za stavljanje van funkcije TR-111 poslužioca
    - Parametri: ctx – Kontekst struktura CPE klijenta
    - Povranta vrednost: Indikacija uspešnosti zaustavljanja TR-111 poslužioca
        - UDP_TRUE – Zaustavljanje TR-111 poslužioca nije uspešno izvršeno usled nastalih grešaka
        - UDP_FALSE – Zaustavljanje TR-111 poslužioca uspešno izvršeno
- **UDP_ERR_T libcpe_delete_udp_server (void *ctx);**
    - Opis: Funkcija vrši deinicijalizaciju TR-111 poslužioca
    - Parametri: ctx – Kontekst struktura CPE klijenta
    - Povratna vrednost: Indikacija uspešnosti deinicijalizacije TR-111 poslužioca
        - UDP_TRUE – Deinicijalizacija TR-111 poslužioca nije uspešno izvršena usled grešaka u toku deinicijalizacije
        - UDP_FALSE – Deinicijalizacija TR-111 poslužioca uspešno izvršena

8.2.6 libcpe_queue.h

- **Q_ERR_T q_add(q_element** start, q_element** end, uint8_t *message, int size, int status);**
    - Opis: Funkcija čija je uloga dodavanje elementa u red. Element se uvek dodaje na kraj reda. Element predstavlja UDP poruku i status te poruke pomoću koga se diferenciraju tipovi poruka
    - Parametri:

        - start – pokazivač na početak reda
        - end – pokazivač na kraj reda
        - message – UDP poruka koja se snima u red
        - status – status poruke koja se unosi u red, takođe čini deo sadržaja elementa reda
    - Povratna vrednost: Indikacija uspešnosti dodavanja u red
        - Q_TRUE – Dodavanje u red nije uspelo usled grešaka
        - Q_FALSE – Dodavanje u red uspešno izvršeno
- **Q_ERR_T q_remove_top (q_element** start, q_element** end);**
    - Opis: Funkcija čija je uloga brisanje elementa sa početka reda
    - Parametri:
        - start – pokazivač na početak reda
        - end – pokazivač na kraj reda
    - Povratna vrednost: Indikacija uspešnosti brisanja elementa sa vrha reda
        - Q_TRUE – brisanje elementa sa vrha reda nije uspelo zbog grešaka koje su nastale u toku brisanja
        - Q_FALSE – brisanje elementa sa vrha reda uspešno izvršeno
- **Q_ERR_T q_get (q_element** start, uint8_t *message, int *status);**
    - Opis: Preuzimanje elementa sa vrha reda
    - Parametri:
        - start – pokazivač na početak reda
        - message – UDP poruka koja je preuzeta iz reda
        - status – status UDP poruke pomoću koga se vrši razvrstavanje poruka
    - Povratna vrednost: Indikacija uspešnosti preuzimanja elementa iz reda
        - Q_TRUE – Preuzimanje elementa iz reda neuspelo usled grešaka
        - Q_FALSE – Preuzimanje elementa iz reda uspešno izvršeno
- **void q_clear_queue (q_element** start, q_element** end);**
    - Opis: Brisanje svih elemenata reda
    - Parametri:
        - start – pokazivač na početak reda
        - end – pokazivač na kraj reda
- **void q_empty (q_element** start, uint8_t *res);**
    - Opis: Funkcija koja proverava da li je red prazan
    - Parametri:

- start – pokazivač na početak reda
- res – rezultat provere stanja popunjenosti reda

- **void q_count_elements (q_element** start, unsigned *count);**
  - Opis: Funkcija koja vrši brojanje elemenata reda
  - Parametri:
    - start – pokazivač na početak reda
    - count – broj elemenata reda

8.3 Programska sprega TR-069 modula na strani ACS poslužioca

U ovom modulu realizovane su sve osnovne funkcionalnosti vezane za TR-069 zahtev za povezivanje na strani ACS poslužioca.

8.3.1 TR069ConnectionRequestBean

- **connectionRequestMethod (DeviceIdStruct deviceIdStruct)**
  - Opis: Vrši određivanje tipa zahteva za konekciju koji će biti korišćen za određenog CPE klijenta. Na osnovu parametra NATDetected iz TR-106 modela podataka određuje se koji tip zahteva za konekciju će biti korišćen.
  - Parametri: deviceIdStruct – Jednoznačno određuje o kom klijentu se radi. Predstavlja ključ po kojem se određuje za kojeg klijenta je potrebno izvršiti zahtev za konekciju
  - Povratna vrednost: Indikacija uspešnosti pozvanog zahteva za konekciju. U slučaju TR-069 zahteva to je kod poruke koju je klijent dobio kao odgovor od TR-069 poslužioca. U slučaju TR-111 zahteva povratna vrednost predstavlja kod poruke HTTP poslužioca preko kojeg se podaci neophodni za kreiranje TR-111 zahteva prenose do STUN poslužioca
- **tr069ConnectionRequestMethod (DeviceIdStruct deviceIdStruct)**
  - Opis: Vrši izvršavanje TR-069 klijenta. Ova funkcija je zadužena za prikupljanje potrebnih parametara iz baze u kojoj se nalaze vrednosti iz TR-106 modela podataka i za izvršavanje klijentske aplikacije TR-069 zahteva za konekciju
  - Parametri: deviceIdStruct – Jednoznačno određuje o kom klijentu se radi

- Povratna vrednost: Indikacija uspešnosti izvršenja autentifikacije. Predstavlja kod poruke odgovora TR-069 poslužioca

8.3.2 TR069ConnectionRequestClient

- **sendGetRequestBasic ()**
  - Opis: Funkcija šalje HTTP GET zahtev u čijem zaglavlju nema parametara
  - Parametri: nema
  - Povratna vrednost: Indikacija uspešnosti slanja GET zahteva. Ukoliko dođe do greške funkcija vraća vrednost nula, u suprotnom povratna vrednost funkcije predstavlja kod HTTP poruke koja predstavlja odgovor TR-069 poslužioca
- **sendGetRequestAuth ()**
  - Opis: Funkcija šalje HTTP GET zahtev sa svim potrebnim parametrima u zaglavlju poruke. Time su stvoreni svi uslovi za uspešnu autentifikaciju TR-069 klijenta.
  - Parametri: nema
  - Povratna vrednost: Indikacija uspešnosti autentifikacije TR-069 klijenta. Ukoliko dođe do greške funkcija vreća vrednost nula, u suprotnom povratna vrednost funkcije predstavlja kod HTTP poruke koja predstavlja odgovor TR-069 poslužioca
- **establishConnection ()**
  - Opis: Funkcija koja vrši uspostavljanje konekcije između TR-069 poslužioca i TR-069 klijenta
  - Parametri: nema
  - Povratna vrednost: nema
- **parseData ()**
  - Opis: Funkcija koja vrši parsiranje HTTP odogovora. Koristi se prilikom dobijanja odgovora od TR-069 poslužioca da je autentifikacija odbijena. U toj poruci se sadrže sve neophodne informacije za uspešnu autentifikaciju
  - Parametri: nema
  - Povratna vrednost: nema