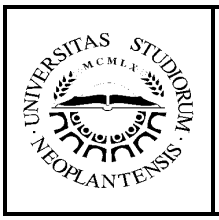


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

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: **Срђан Шобић**
Број индекса: **12772**

Тема рада: **Симулатор ИП мреже**

Ментор рада: **др Мирослав Поповић**

Нови Сад, јун, 2012

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**:		Завршни (Bachelor) рад	
Аутор, **АУ**:		Срђан Шобић	
Ментор, **МН**:		проф. др Мирослав Поповић	
Наслов рада, **НР**:		Симулатор ИП мреже	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		2012	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		7/51/0/5/23/0/0	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кључне речи, **ПО**:		ВВТ систем, HTTP Live Streaming протокол, D-ITG, Sirannon, FFmpeg	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		У оквиру рада је реализовано једно решење система за тестирање модерних дигиталних ТВ пријемника, са могућношћу мерења и анализе саобраћаја унутар локалне мреже.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	проф. Др Иштван Пап	
	Члан:	проф. Др Јелена Ковачевић	Потпис ментора
	Члан, ментор:	проф. др Мирослав Поповић	

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**:		Bachelor Thesis	
Author, **AU**:		Srđan Šobić	
Mentor, **MN**:		PhD Miroslav Popović	
Title, **TI**:		IP Network Simulator	
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**:		2012	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/51/0/5/23/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		BBT system, HTTP Live Streaming protocol, D-ITG, Sirannon, FFmpeg	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		In this paper was made one solution for testing of the modern digital TV receivers, with a possiblity of traffic measurment and analysis of a local network.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	PhD Ištvan Pap	
	Member:	PhD Jelena Kovačević	Menthor's sign
	Member, Mentor:	PhD Miroslav Popović	

Захвалност

Захваљујем се ментору, проф. др Мирославу Поповићу, на досадашњој сарадњи, као и на корисним, стручним, али и личним саветима и идејама током израде овог рада.

Такође бих желео да се захвалим породици, пријатељима и колегама на подршци, не само током израде овог рада, него и током целокупног времена проведеног на студијама.

САДРЖАЈ

Списак слика

Списак табела

Скраћенице

HLS – *HTTP Live Streaming, HTTP* оријентисан комуникациони протокол за дистрибуцију мултимедијалних токова података

DITG – *Distributed Internet Traffic Generator,* програмски пакет за генерисање и мерење мрежног саобраћаја,

VOD, *Video On Demand,* видео на захтев,

TSP, *Traffic Specification Protocol,* комуникациони протокол са стране *DITG*

TCP, *Transmission Control Protocol,* комуникациони протокол који припада транспортном слоју

IPC, *Inter Process Communication,* међупроцесна комуникација

EC, *Event communication,* комуникација догађајима

QoS, *Quality Of Service,* квалитет услуга

1. Увод

У раду је реализовано почетно решење симулатора ИП мреже за потребе тестирања модерних дигиталних тв пријемника. Задатак симулатора јесте да омогући генерисање, мерење и анализу ИП саобраћаја унутар локалне мреже, која припада систему за тестирање тв пријемника по принципу црне кутије (*Black Box Testing).* Симулатор ИП мреже треба развити коришћењем јавних и бесплатних решења и интегрисати у ББТ систем који је развијен у институту РТ-РК.

Задатак овог рада се може поделити у пар целина: прва целина би била упознавање са *HTTP live streaming* протоколом, начином рада и теоријским основама, другу целину би представљало упознавање са програмским алатима за генерисање и мерење мрежног саобраћаја (*D-ITG*), програмским пакетом који би се користио на пријемној страни (*Ffmpeg*), програмским пакетом који би се користио на предајној страни (*Sirannon*), док би трећу целину представљала имплементација система који би посредовао између клијента и сервера где се комуникација одвија на основу горе поменутог протокола, односно имплементација сервера посредника. Додатни задатак сервера посредника био би мерење саобраћаја који протиче кроз њега, као и да омогући додатан, оптерећујући саобраћај, који би омогућио да се сагледају карактеристике система сервера посредника за различите услове и стања мреже којој припада, изнесу субјективна и објективна запажања. Сви рачунари који ће припадати поменутом систему су рачунари унутар локалне мреже.

Рад се састоји од седам поглавља. У другом поглављу дате су теоријске основе за комуникациони протокол и наведене програмске алате који ће бити коришћени за имплементацију система. Треће поглавље сачињава концепт решења система сервера посредника, који је пропраћен одговарајућим дијаграмима размена порука, као и *SDL*

дијаграмима. Четврто поглавље представља програмско решење поменутог система, и дати су описи модула, као и атрибута и метода, од којих се модули састоје. У петом поглављу дати су резултати тестирања, а шесто поглавље представљају недостаци и мане имплементираног система. Седмо поглавље представља коришћена литература током израде рада.

2. Теоријске основе

У овом делу ће бити дате теоријске основе о комуникационом протоколу коришћеном за слање тока аудио и видео података између клијента и сервера, као и о програмским пакетима коришћеним на предајној страни и пријемној страни, програмским пакетима намењеним за мерење саобраћаја на серверу посреднику, и генерисање додатног оптерећујућег саобраћаја, како би се створили различити услови и стања мреже, за потребе тестирања система.

2.1 HTTP HLS Streaming (HTTP live streaming) протокол

HTTP HLS протокол или *HTTP live streaming* протокол је *HTTP* оријентисан комуникациони протокол имплементиран од стране *Apple* корпорације. Направљен је у циљу успостављања комуникације и слања аудио и видео података према *Apple*-овим уређајима. Спада у групу адаптивних комуникационих протокола, односно, протокола код којих је могуће подесити оптималну брзину слања података за различите типове уређаја на пријемној страни (рачунари, мобилни уређаји, преносни уређаји или таблети), односно за различите карактеристике успостављене везе. Наведен протокол подржава дистрибуцију и *LIVE* (уживо видео) и *VOD* (*video on demand* – видео на захтев) начина слања тока података. Ова два наведена начина имају сличне карактеристике, подржавају исте опције, с тим што техника видеа на захтев представља унапред складиштен материјал за слање, те је могуће извршити опције типа премотавања аудио и видео садржаја. Овај протокол се заснива на принципу да се укупан скуп података подели на више мањих делова, који су *HTTP* оријентисани и адаптивних битских брзина, повежу у листу, којима ће клијент приступати.

2.2 D-ITG (Distributed Internet Traffic Generator) програмски пакет

D-ITG (*Distributed Internet Traffic Generator*) представља бесплатан програмски алат, развијен на универзитету у Наполију (Италија), који служи за генерисање *IP* саобраћаја, као и мерење генерисаног саобраћаја и обраду добијених података. Његова намена је вишеструка и као такав, може се користити за:

a. Анализу мреже и оцену перформанси: квалитет протока, губитак пакета, анализа кашњења и промена вредности кашњења (*jitter*) код хетерогених мрежа,
b. Тестирање могућности уређаја: стони рачунари, лаптопови, мобилни уређаји,
c. Тестирање *QoS* архитектуре,
d. Анализу алгоритама за усмеравање (*Routing algorithm*),
e. Анализу прилагодљивости и понашања протокола.

D-ITG анализира генерисани мрежни саобраћај који се заснива на слању пакета по принципу пакет по пакет. Мрежни саобраћај је специфициран помоћу: међувремена слања пакета (*Inter Departure Time - IDT*) – време између успешног преноса два пакета, као и величином пакета – количина пакета која се шаље у пакету.

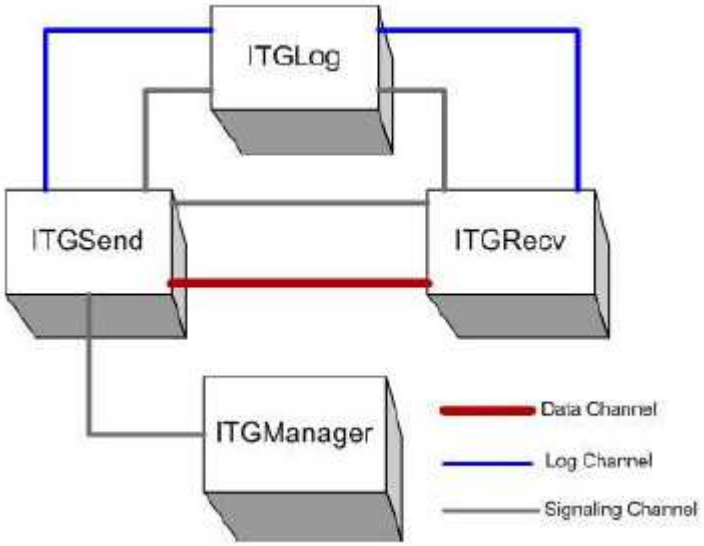
Неке од особина *D-ITG*:

a. компатибилан је са више оперативних система (*Linux, Windows*),
b. могуће је генерисати више токова података,
c. примаоц и пошиљаоц могу да сачувају информације о послатом/ примљеном саобраћају,
d. може да репродукује реалну шему саобраћаја.

D-ITG архитектуру сачињавају следећи елементи:

a. *ITGSend* – користи се за генерисање мрежног саобраћаја,
b. *ITGRecv* – користи се за пријем генерисаног саобраћаја од стране *ITGSend*,
c. *ITGDec* – користи се за анализу резултата послатог/примљеног саобраћаја,
d. *ITGLog* – користи се као сервер за прикупљање података о послатом/примљеном саобраћају између *ITGSend* и *ITGRecv*, опционо,
e. *ITGManager* – користи се за даљинско управљање са *ITGSend*, опционо.

Преко линија података се обавља слање/пријем података, док сигналне линије служе да преко њих елементи архитектуре користећи *TSP* протокол комуницирају међусобно. Трећи скуп линија се користи за слање података ка серверу где се обрађују резултати тестирања генерисаног/примљеног саобраћаја.



Слика 2.1 *D-ITG* архитектура

Уведен је нови протокол за за комуникацију између елемената *D-ITG* (*TSP – Traffic Specification Protocol*): пошиљаоц и примаоц одређују експерименталне захтеве и контролишу генерисани саобраћај помоћу *TSP*:

a. Успостави се веза између примаоца и пошиљаоца,
b. Врши се аутентификација примаоца,
c. Размена информација о генерисаним процесима,
d. Прекида се веза између примаоца и пошиљаоца,
e. Откривање генерисаних догађаја.

D-ITG имплементира *TSP* протокол у виду *TCP* сигналних линија између *ITGSend* и *ITGRecv*. Захваљујући вишенитној имплементацији *ITGSend* и *ITGRecv*, свака сигнална линија може бити искоришћена за генерисање више токова података. За генерисање више токова података, успоставља се веза између нити контролера *ITGRecv* и нити контролера *ITGSend*. Те нити имплементирају *TCP* протокол и одговорне су за инстанцирање и уништавање нити које генеришу пакете и које су задужене за пријем генерисаних пакета. Координација између нити контролера и нити које

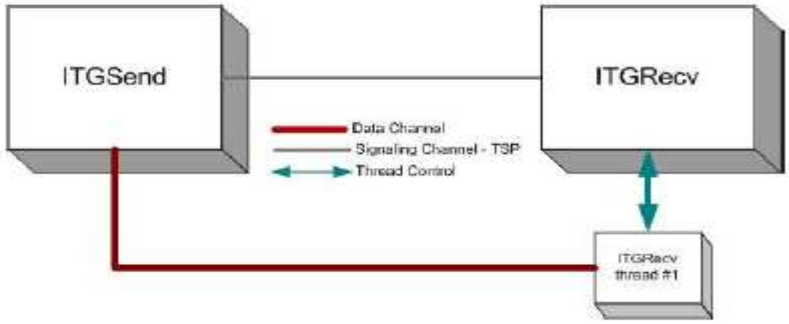
генеришу/примају генерисане пакете се врши помоћу међупроцесне комуникације у системима заснованим на *Unix*, или помоћу комуникације догађајима у *Windows* оперативном систему.

2.2.1 ITGSend

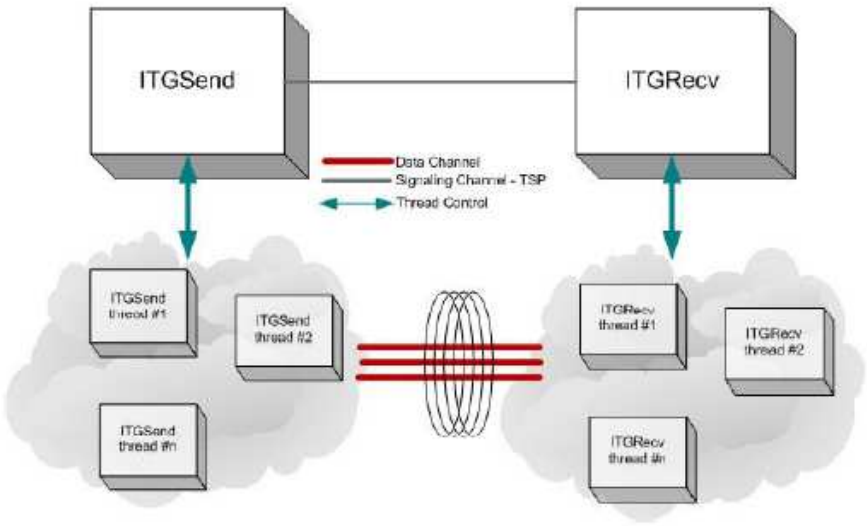
Представља програм за генерисање мрежног саобраћаја у зависности од параметара који му се прослеђују. Користи се на предајној страни. Подржава три начина рада:

a. Режим рада са једним током: генерише један ток података, користи се једна нит и она је одговорна за генерисање пакета и управљање сигналним линијама кроз *TSP* протокол.
b. Режим рада са више токова: генерише се више токова података (више токова података - више нити). Једна од нити имплементира *TSP* протокол и управља процесом генерисања.
c. Режим управљања: омогућава даљинским управљањем са *ITGSend* помоћу *ITGManager* користећи *ITGApi*.

ITGSend може генерисати *LOG* датотеку где се налази опис послатих пакета. Наведена датотека може бити сачувана на локалној машини или на серверу (даљински) где је активан *ITGLog*.



Слика 2.2 Режим рада са једним током података



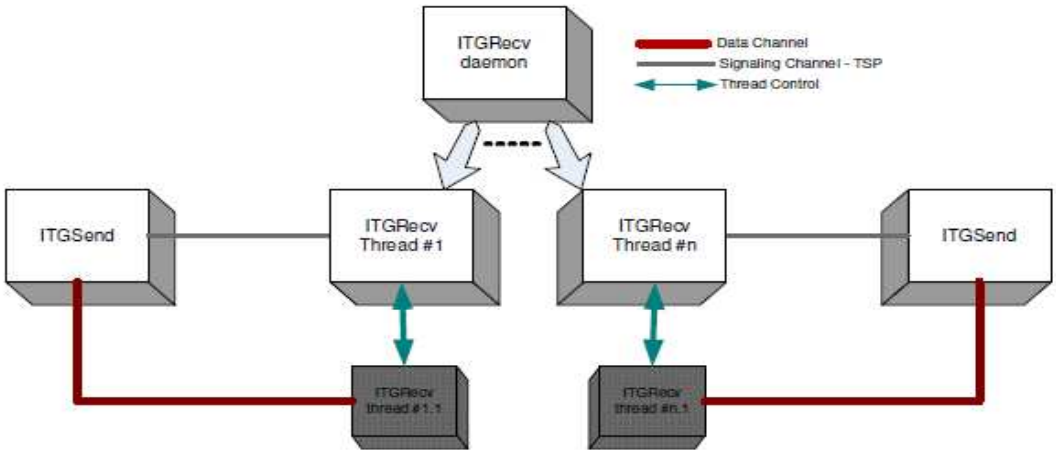
Слика 2.3 Режим рада са више токова података

Приликом рада са *ITGSend*, потребно је навести одговарајуће параметре за рад, на основу којих се успоставља веза и шаљу одговарајући пакети. У случају потребе генерисања више токова података, потребно је користити *script_file* у којем се налазе потребни параметри за сваки ток, опционо. У свим осталим случајевима није потребно користити *script_file*, довољно је навести параметре у командној линији.

2.2.2 ITGRecv

ITGRecv ради као конкурентни програм (стално је активан) на пријемној страни. Користи се за пријем генерисаног саобраћаја. Он листа да ли има нових *TSP* веза. Када стигне захтев за *TSP* везу, он генерише нову нит која је одговорна за имплементацију TSP протокола на пријемној страни.

Пре почетка експерименталног рада између *ITGRecv* и *ITGSend*, имплементира се аутентификациони протокол. Слично *ITGSend*, *ITGRecv* може да генерише *LOG* датотеку која се користи за анализу примљеног саобраћаја. Такође може бити сачуван на локалној машини или на серверу где је активан *ITGLog*.



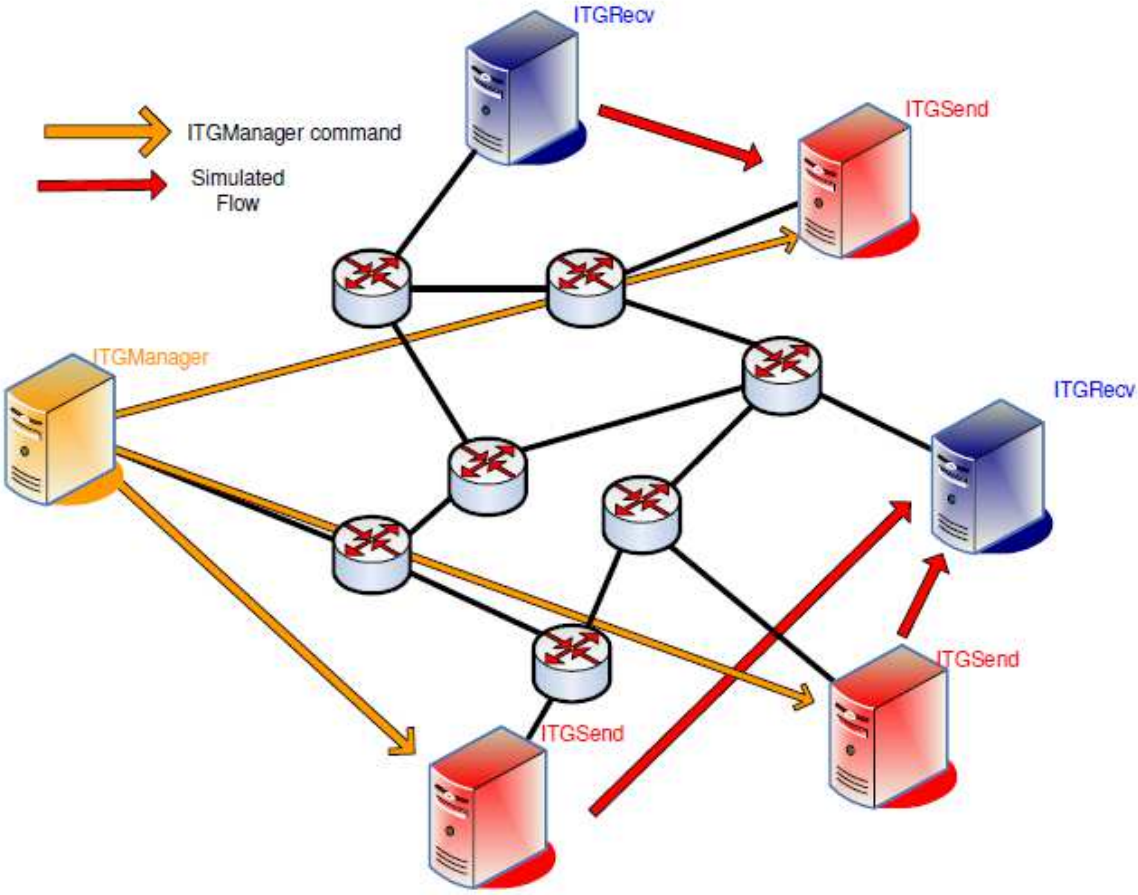
Слика 2.4 Принцип рада *ITGRecv*

Као што је био случај са *ITGSend*, и *ITGRecv* захтева одређене параметре за исправан рад.

2.2.3 ITGManager

ITGManager може бити у позадинском (*daemon*) режиму рада, где је у стању приправности све док не добије команду од *ITGManager*. *ITGManager* користи *ITGApi* за даљинско управљање са *ITGSend*. *ITGApi* је *API* који се користи за слање специфичне поруке *ITGSend*, где је први параметар адреса рачунара на којем је покренут *ITGSend*, а остало представљају параметри везани са *ITGSend*.

Може да контролише више од једног *ITGSend*, такође може да контролише цео саобраћај кроз мрежу.



Слика 2.5 Принцип употребе *ITGManager* у циљу управљања *ITGSend* у мрежи

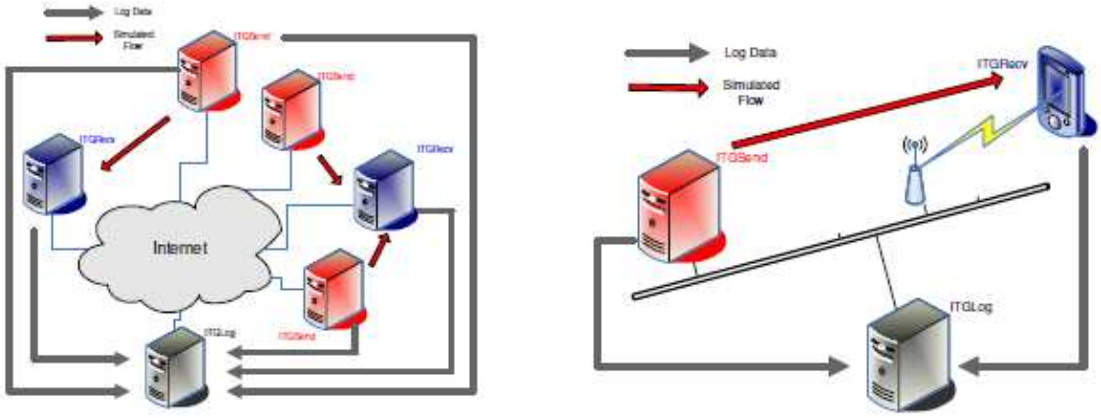
2.2.4 ITGLog

Прикупља групу података о генерисаним процесима између пошиљаоца (*ITGSend*) и примаоца (*ITGRecv*). У ту групу спадају: број токова за слање пакета који су у употреби, адреса изворишта, адреса одредишта, број послатих пакета, број изгубљених пакета, величина пакета, број примљених података, информације о кашњењу и одступању.

ITGLog може бити на подигнут на посебној машини (*LOG* Сервер), где може да прима податке између више примаоца и пошиљаоца, или на локалној машини примаоца или пошиљаоца. *LOG* информације се могу слати преко *UDP* или *TCP* везе успостављене између *LOG* сервера и примаоца и пошиљаоца.

ITGLog се може користити у разним случајевима као што су:

a. Регионална рачунарска мрежа (*WAN*): користи се у циљу једноставног скупљања информација од свих примаоца и пошиљаоца присутних у мрежи (слика лево);
b. Уређаји са ограниченим ресурсима: као што су нпр. *PDA* уређаји, мобилни телефони; због недостатка меморије на наведеним уређајима, *ITGLog* се користи за прикупљање и смештање информација уместо наведених уређаја (слика десно).



Слика 2.6 Принцип употребе *ITGLog*

2.2.5 ITGDec

Користи се за анализу прикупљених података и приказ битних информација као што су број генерисаних токова, укупно време слања, број послатих/примљених пакета, број изгубљених пакета, број послатих/примљених бита, вредности минималног, максималног и средњег кашњења, вредности одступавања, стандардног одступања, средња густина протока (два приказа – број бита у секунди и број пакета у секунди). Уколико постоји више токова, могућ је приказ за сваки ток појединачно и резултат за све токове заједно.

2.3 Sirannon програмски пакет

Sirannon је програмски пакет који се користи као универзални мултимедијални сервер за слање различитих аудио и видео података или као пријемник у циљу пријема наведених података и репродукције истих. Развијен је на универзитету у Генту

(Белгија). Подржава различите типове аудио и видео података за слање или репродукцију, као и различите врсте комуникационих протокола помоћу којих се остварује веза између пријемне и предајне стране и врши слање података. У ту групу спадају *RTSP/RTP/UDP, RTMP/TCP, RTMPT/HTTP, HTTP, RTP/UDP, TCP*. Од интереса је употреба наведеног програмског пакета на предајној страни, као *HTTP* сервер. Приликом извршавања програма прослеђују му се путања до конфигурационе датотеке и одређени параметри, на основу којих се врши подешавање програма за рад у складу са прослеђеним параметрима. Пример покретања програма из командног прозора је следећи*: sirannon dat/xml/media-server-std.xml dat/media.*

2.4 FFmpeg програмски пакет

FFmpeg је потпун програмски пакет намењен за обраду, емитовање и слање аудио и видео датотека. Подржава различите типове аудио и видео података. Садржи скуп библиотека које могу бити коришћене у апликацијама. Садржи програме који могу бити коришћени у циљу слања аудио и видео тока података, конвертовање података у други тип, као и емитовање примљених података. Из програмског пакета издвајају се следећи програми:

1. *ffmpeg* – апликација која се позива из командног прозора за конвертовање датотека у други облик,
2. *ffserver* – користи се као сервер за уживо слање аудио и видео података,
3. *ffplay* – једноставан медиа пријемник, заснован на SDL и Ffmpeg дијаграмима,
4. *ffprobe* – користи се за анализу аудио и видео тока података.

За потребе овог рада користиће се само *ffplay* програм на пријемној страни у циљу пријема тражених аудио и видео података.

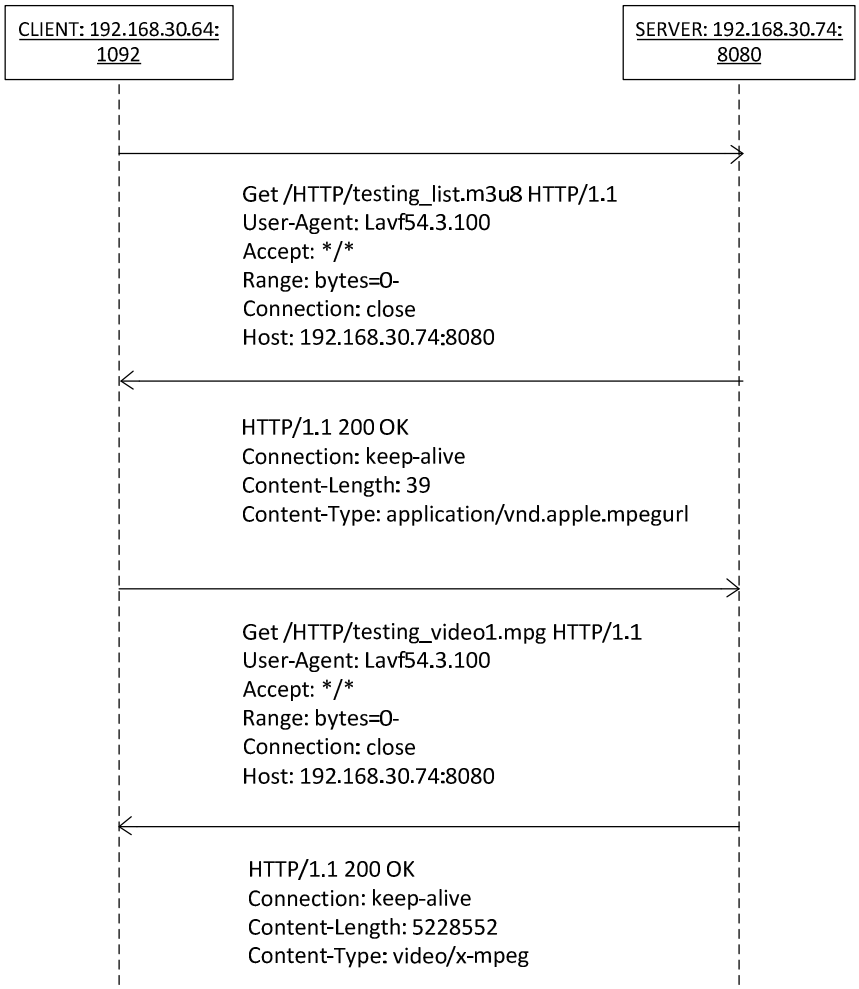
3. Концепт решења

Сервер посредник или *Proxy* сервер је замишљен као транспарентан сервер посредник између клијента и сервера, чији би задатак био да успостави везу са клијентом и стварним сервером, као и да прослеђује поруке од клијента ка серверу и обрнуто. Помоћу програма за анализу мрежног саобраћаја снимљена је размена порука између клијента и сервера, на основу чега би се створила основа за пројектовање сервера посредника. У наставку су дати дијаграми размене порука између клијента и сервера.

Слика 3.1 Размена порука између клијента и сервера – послат је захтев за датотеку

На слици је приказана ситуација када клијент шаље серверу *HTTP* захтев за слање аудио и видео тока података датотеке *testing_video1.mpg* у виду следеће адресе:

http://192.168.30.74:8080/HTTP/testing_video1.mpg. Сервер одговара потврдном поруком достављајући му тражене податке.

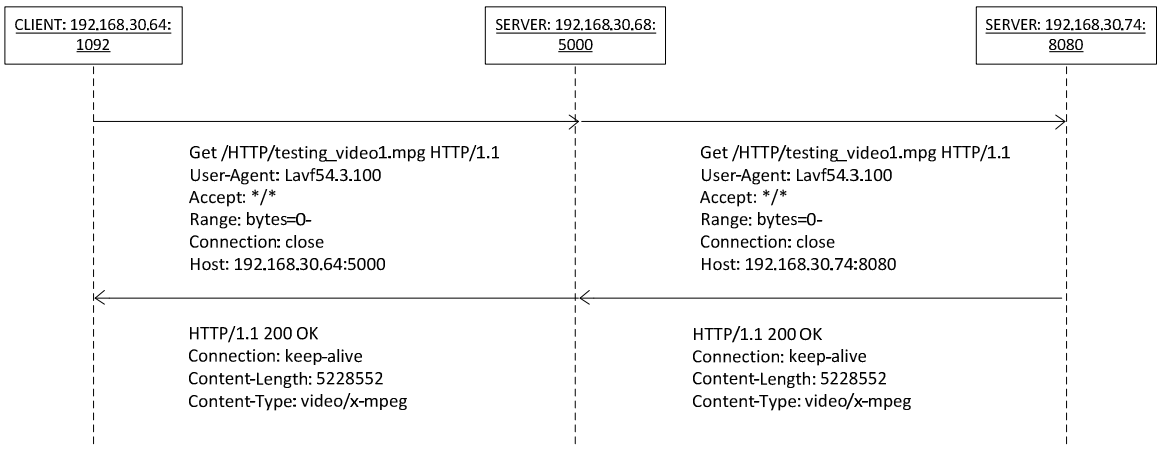


Слика 3.2Размена порука између клијента и сервера – послат је захтев за листу датотека

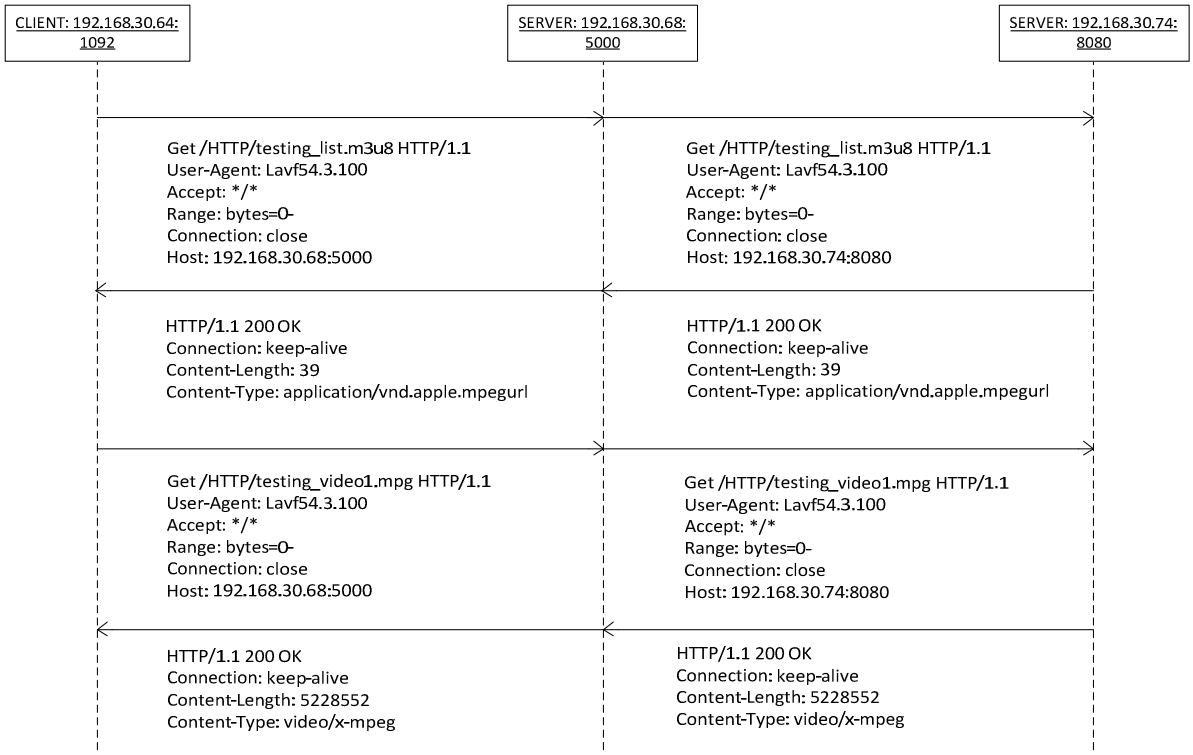
Друга ситуација је када клијент шаље серверу *HTTP* захтев који уместо датотеке садржи назив листе у облику следеће адресе: *http://192.168.30.74:8080/HTTP/list.m3u8,* након чега сервер одговара потврдном поруком и шаље клијенту списак датотека из листе. Након пријема листе датотека, клијент шаље редом захтев за сваку датотеку, појединачно, као у првом случају.

На основу горњих дијаграма размене порука види се да би задатак сервера посредника требао да буде не само размена порука, већ и измена заглавља поруке примљене од клијента, где уместо адресе и пролаза сервера посредника, треба убацити адресу и пролаз сервера. У наставку следе дијаграми размене порука између клијента и

сервера, где је још убачен и сервер посредник између њих, и преко њега се одвија цела размена порука:



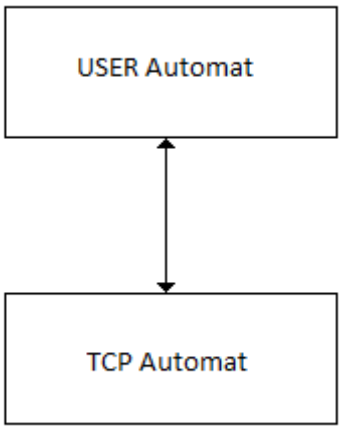
Слика 3.3 Размена порука између клијента, сервера посредника и сервера – послат је захтев за датотеку



Слика 3.4 Размена порука између клијента, сервера посредника и сервера – послат је захтев за листу датотека

На основу ових дијаграма размена порука, следи реализација самога сервера посредника. Сервер посредник је реализован као систем два аутомата, где је један аутомат задужен за успоставу везе са пријемном и предајном страном, разменом порука

између њих, окренут је ка транспортном и мрежном делу, а други аутомат је окренут кориснику, односно обавештава корисника о успостављеној вези, управља пристиглим и послатим порукама. Они чине хијерархију аутомата где је *USER* аутомат на врху хијерархије и управља са *TCP* аутоматом, који је на нивоу мањем од поменутог аутомата.



Слика 3.5 Хијерархија система аутомата

Приликом покретања програма потребно је као параметре у командној линији проследити *IP* адресу сервера, дозволу за мерење саобраћаја између предајне и пријемне стране, а који протиче кроз сервер посредник као и дозволу за протицање осталог саобраћаја, који није везан за пренос аудио и видео података, односно представља непотребан саобраћај – користиће се касније у циљу ометања примарног саобраћаја и испитивања карактеристика система за различите услове рада. Први параметар је обавезан, а преостала два ако се не наведу, подразумева се рад без укључених тих додатака.

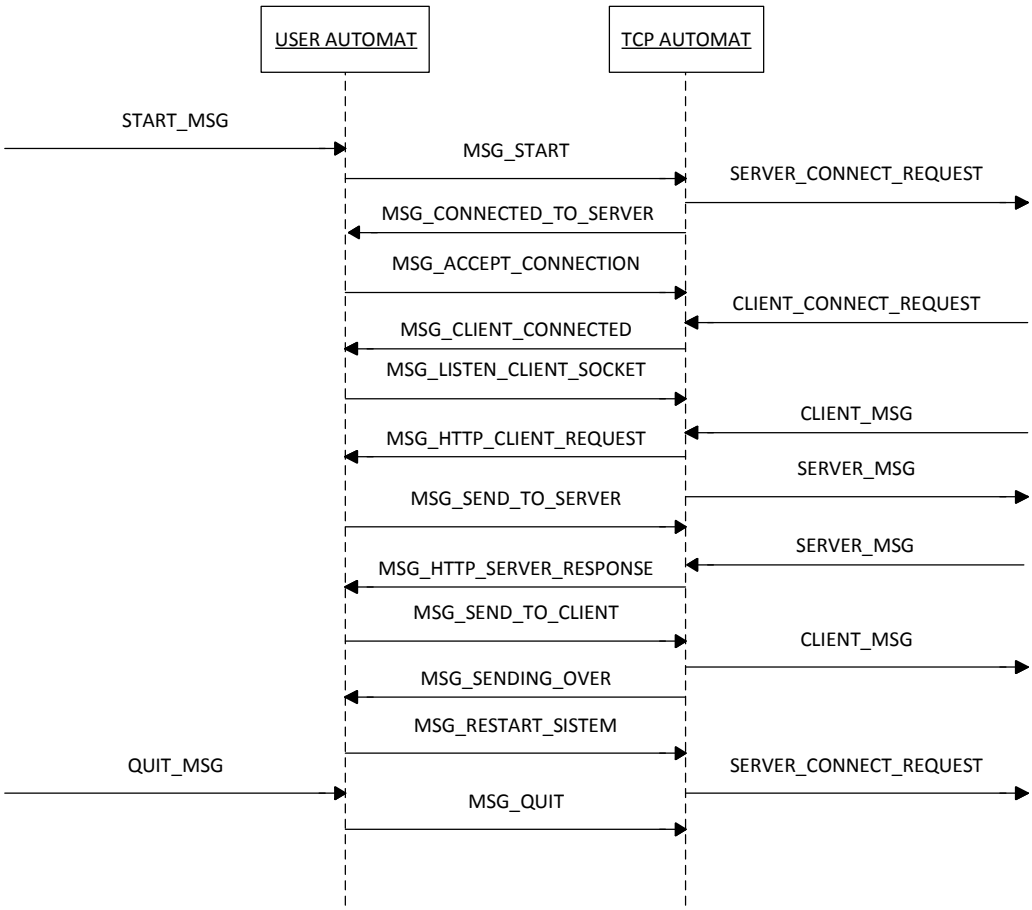
Након прослеђивања параметара и покретања програма, од клијента се очекује да унесе *start* за покретање система аутомата или *quit* за прекид рада програма или прекид рада система аутомата и програма уколико је систем већ био покренут. Након што корисник унесе команду за почетак, *USER* аутомат добија поруку за почетак рада, и ту исту поруку прослеђује *TCP* аутомату. *TCP* аутомат подешава параметре за повезивање са сервером, и након успешног успостављања везе, обавештава о томе *USER* аутомат. *USER* аутомат прима поруку од *TCP* аутомата и обавештава га да слуша одређен пролаз у циљу успостављања везе са клијентом. Након што се успостави веза са клијентом, о

томе обавештава *USER* аутомат, при чему од њега добије одговор и слуша утичницу према клијенту како би примио *HTTP* захтев од клијента упућен серверу.

Примљени захтев проследи *USER* аутомату, исти аутомат обради захтев, мења заглавље поруке где се налази адреса и пролаз сервера посредника и убацује адресу и пролаз стварног сервера, и прослеђује измењени захтев *TCP* аутомату. *TCP* аутомат прослеђује измењени захтев серверу и чека на одговор од сервера. Након што прими *HTTP* одговор од сервера, обавештава о томе *USER* аутомат, који му одговара поруком, како би могао да врши пријем података од сервера и прослеђивање ка клијенту. Након што се изврши слање аудио и видео података, обавештава о томе *USER* аутомат, који му одговара поруком о прекиду веза са клијентом и сервером.

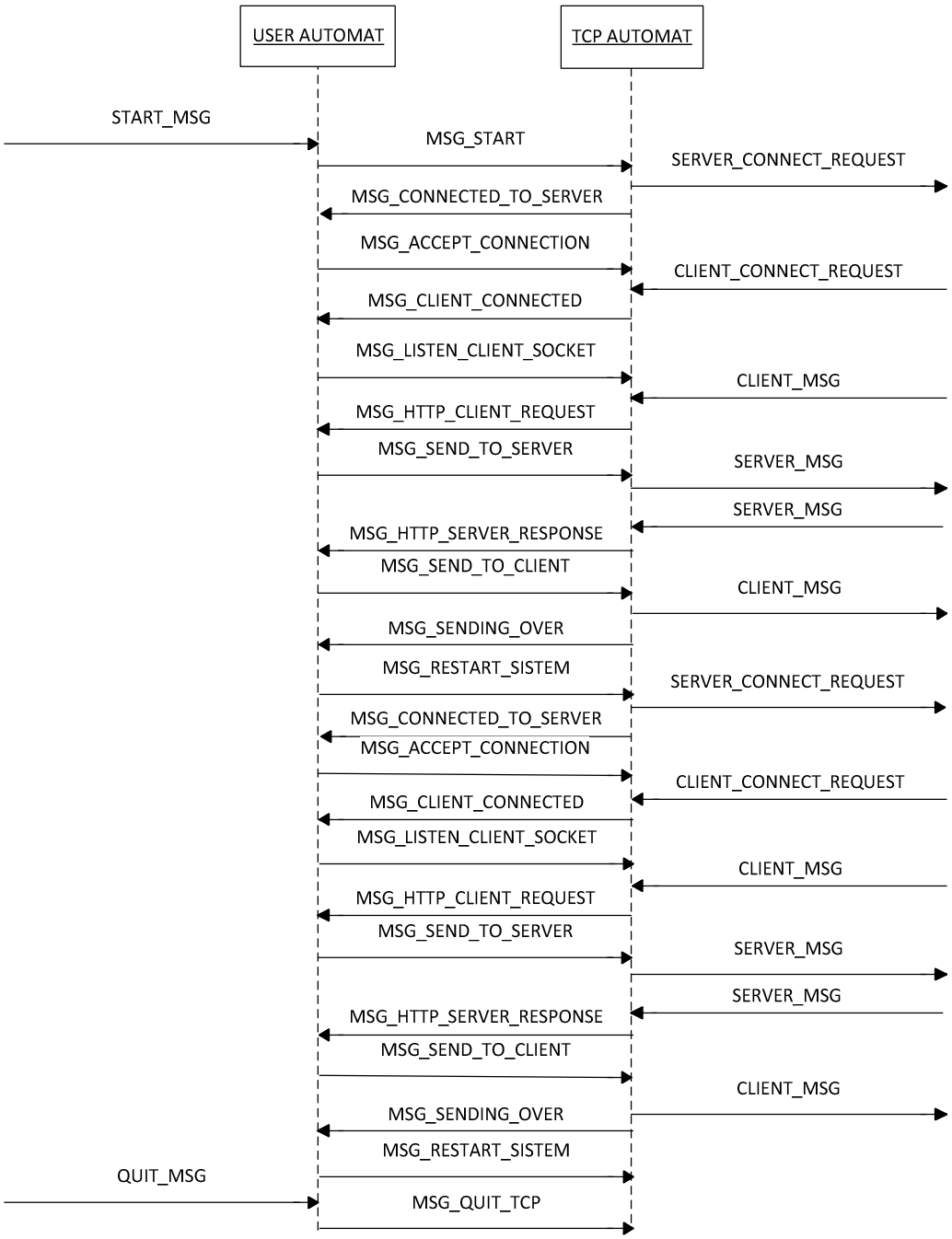
После раскида веза, *TCP* аутомат шаље поруку *USER* аутомату и добија одговор ради поновног повезивања са сервером, односно понављају се претходни кораци у циљу поновне успоставе везе са клијентом и сервером и слања аудио и видео података ка клијенту. У току рада система аутомата, клијент може послати захтев за прекид рада, у случају да се не врши слање података од сервера преко сервера посредника ка клијенту. У случају да је наведена неисправна адреса сервера, или је клијент навео неисправан назив датотеке у својем захтеву, односно шаље захтев за датотеком која се не налази на серверској страни, тада ће главни аутомат обавестити корисника о насталој грешци и систем аутомата ће завршити са својим радом.

Целокупан систем аутомата је заснован на следећим дијаграмима секвенци, као и *SDL* дијаграмима, датих у наставку.



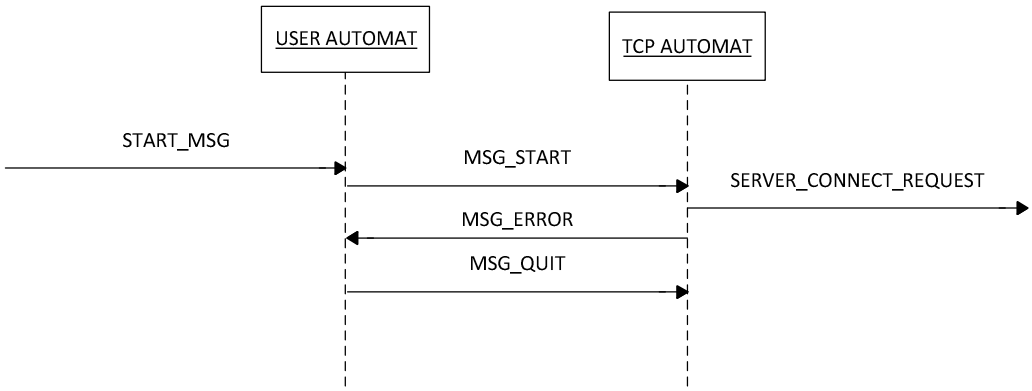
Слика 3.6 Размена порука између аутомата у случају исправног рада – послат је захтев за датотеку

Слика представља рад система аутомата у случају када клијент пошаље захтев серверу за слање тока аудио и видео података једне датотеке. Размена порука између аутомата у систему је описана на претходној страни.



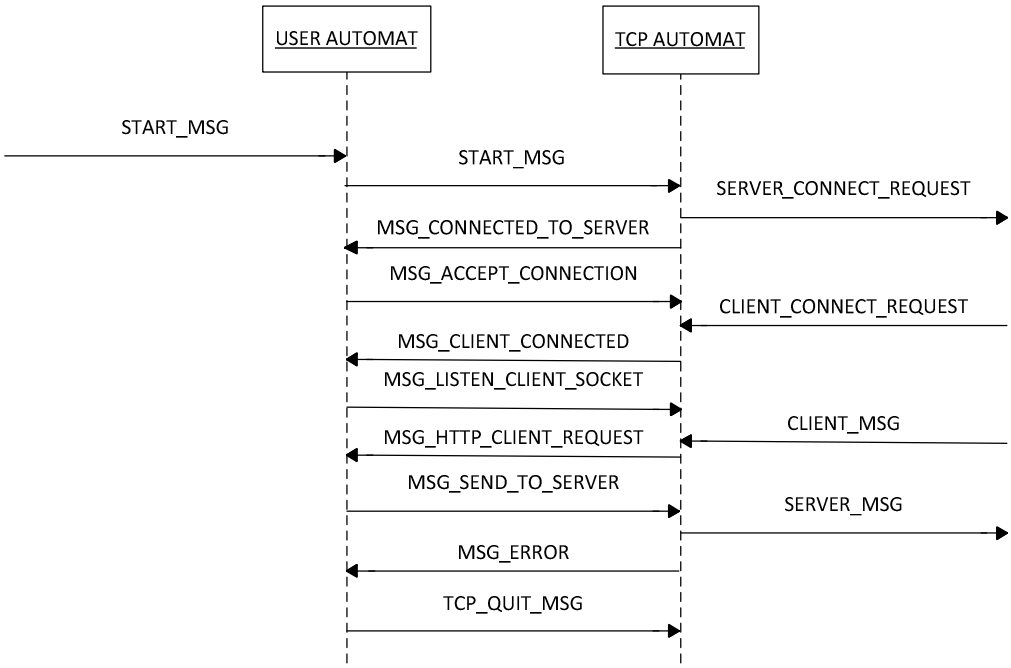
Слика 3.7 Размена порука између аутомата у случају исправног рада – послат је захтев за листу датотека

Друга могућа ситуација јесте та да клијент пошаље *HTTP* захтев серверу у којој се налази назив листе која садржи информације о датотекама. Сервер у том случају пошаље списак датотека клијенту, након чега клијент шаље *HTTP* захтев за сваку датотеку из листе, понаособ. Тада се систем понаша идентично, као у првом случају.



Слика 3.8 Размена порука између аутомата у случају неуспешног повезивања са сервером

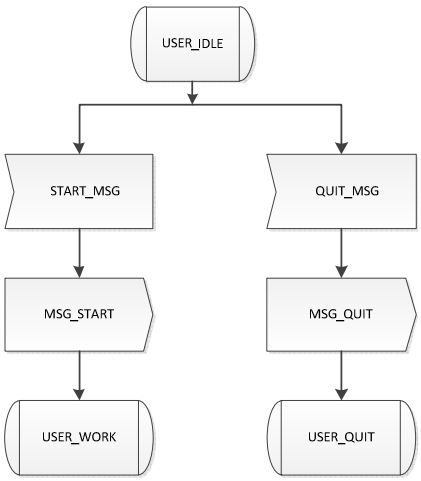
На слици је приказан пример неуспешног повезивања сервера посредника са сервером. Изврши се иницијализација система, покушава се успоставити веза са сервером. У случају неуспешног повезивања TCP_Automat шаље поруку о насталој грешци USER_Automat-у, који обавештава корисника о типу настале грешке и прекида се рад система аутомата.



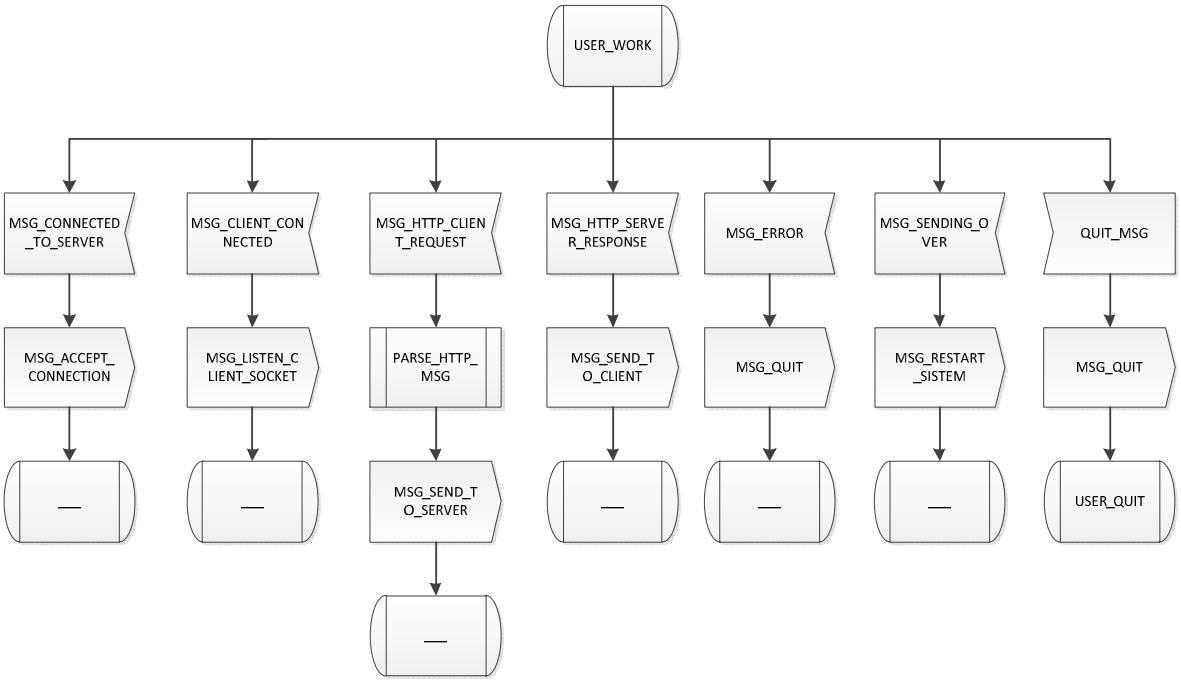
Слика 3.9 Размена порука између аутомата у случају када клијент пошаље захтев са неисправним називом датотеке.

На слици је приказан пример слања *HTTP* захтева са неисправним називом датотеке, односно називом датотеке која не постоји на серверу. Сервер посредник се повеже са сервером, успостави везу са клијентом који му шаље *HTTP* захтев. Сервер посредник измени садржај поруке, пошаље је серверу и уместо одговора, прекине се веза са сервером. Тада TCP_Automat шаље поруку USER_Automat-у који обавештава корисника о типу настале грешке, и прекида се рад система аутомата.

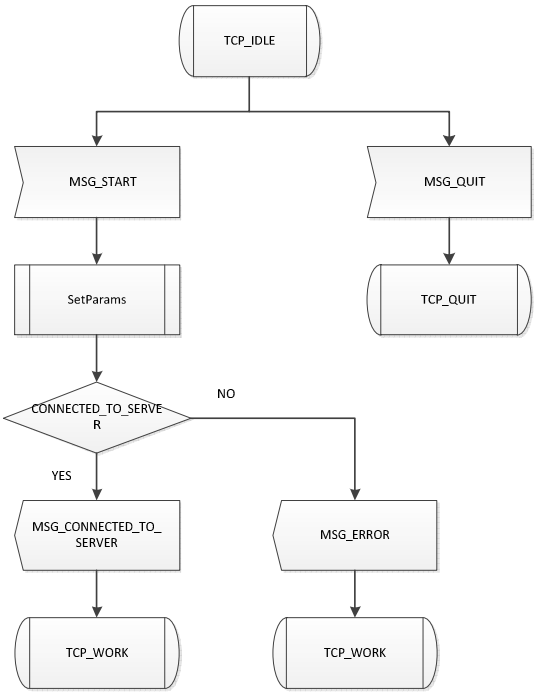
У складу са претходним дијаграмима секвенци, урађени су и *SDL* дијаграми за систем аутомата који су дати у наставку.



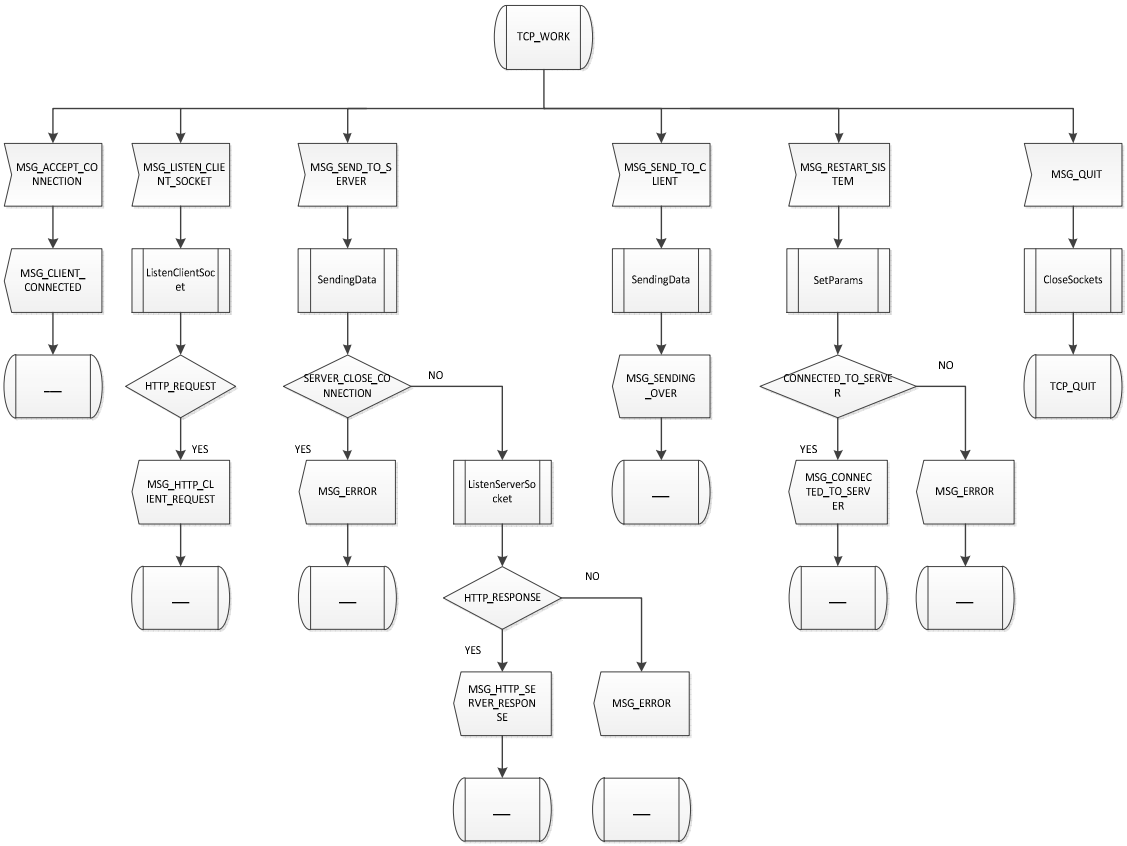
Слика 3.10 *SDL* дијаграм USER_Automat-а у стању *IDLE*



Слика 3.11 *SDL* дијаграм USER_Automat-а у стању *WORK*



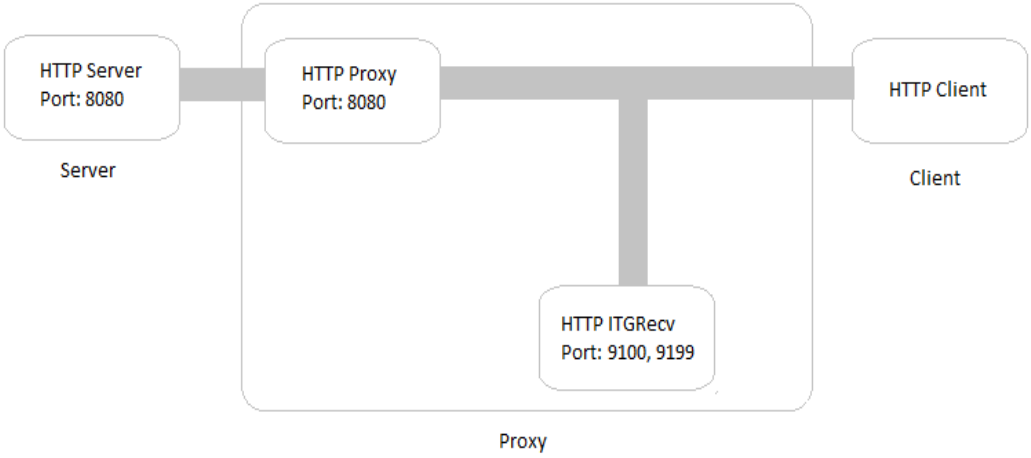
Слика 3.12 *SDL* дијаграм TCP_Automat-а у стању *IDLE*



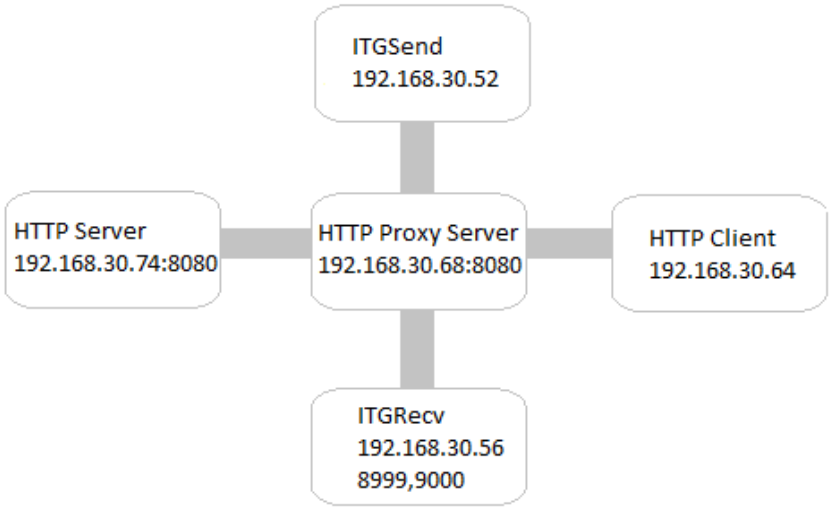
Слика 3.13 *SDL* дијаграм TCP_Automat -а у стању *WORK*

Сервер посредник, осим што посредује у размени порука и података између клијента и сервера, такође може да мери саобраћај који протиче кроз њега, као и да размењује додатни други саобраћај, не везан за првобитни главни саобраћај. Идеја је да се омогући протицање додатног саобраћаја кроз сервер посредник, како би се утицало на размену основног саобраћаја и како би се добиле карактеристике сервера посредника за различите услове и стања мреже којој припадају сви елементи. Наравно, користе се аудио и видео датотеке различитих карактеристика, дужине трајања, густине записа и остало. У наставку су дате структуре система у случајевима са и без мерења саобраћаја и додатног саобраћаја.

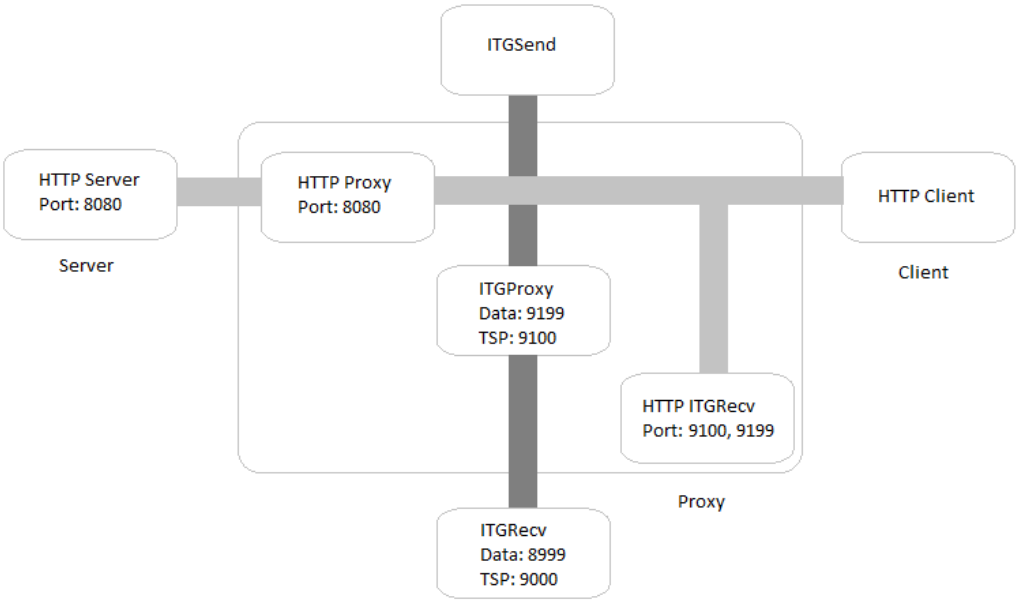
Слика 3.14 Упрошћена архитектура система



Слика 3.15 Детаљна архитектура система са могућношћу мерења саобраћаја



Слика 3.16 Архитектура система са додатним елементима за генерисање и мерење додатног саобраћаја



Слика 3.17 Детаљна архитектура система са додатним елементима за генерисање и мерење саобраћаја.

4. Програмско решење

Систем аутомата се састоји од два модула: *USER_Automat.cpp* и *TCP_Automat.cpp*. Поред ових модула, постоји скуп датотека за рад са аутоматима из скупа *FSM*, као и две помоћне датотеке *TCP_File.cpp* и *String_Parser.cpp* у којима се налазе функције за обраду порука између пријемне и предајне стране, као и подршка за пријем и прослеђивање саобраћаја између *ITGSend* и *ITGRecv*.

USER_Automat.cpp садржи класу *USER_Automat* из скупа *FSM* који садржи следеће атрибуте и методе:

1. Атрибути:
   a. int quitFlagUser.

2. Методе:
   a. void GetCommandFromUser(char *input, char *paramAddress, char *paramDITG, char *paramOtherTraffic, int *quitFlag)
   b. void AcceptConnection()
   c. void ListenClientSocket()
   d. void parseHTTPRequest()
   e. void parseHTTPResponse()
   f. void restartSistem()
   g. void QuitUser()
   h. void ErrorProcessing()

TCP_Automat.cpp садржи класу *TCP_Automat* из скупа *FSM* који садржи следеће атрибуте и методе:

1. Атрибути:

a. SOCKET server_socket,
b. SOCKET client_socket,
c. SOCKET proxy_socket,
d. SOCKET sockfd,
e. sockaddr_in sraddr,
f. sockaddr_in claddr,
g. sockaddr_in pxaddr,
h. unsigned int contentLength,
i. int numberOfFIles,
j. int serverPort.

2. Методе:

a. void StartTCP()
b. void QuitTCP()
c. void ConnectToServer()
d. void AcceptConnectionFromClient()
e. void ListenClientSocket()
f. void SendHTTPRequestToServer()
g. void sendToClient()
h. static DWORD WINAPI AcceptConnectionFromClientThread(LPVOID lpParam)
i. static DWORD WINAPI DITGProxyThread(LPVOID lpParam)

TCP_File.cpp садржи следеће атрибуте и методе:

1. Атрибути:

a. static struct Net_Struct,
b. static bool tcpEnd,
c. static SOCKET cl_socket,
d. static SOCKET cl_socketL,
e. static SOCKET sr_socket,
f. static SOCKET sr_socketL,
g. static SOCKET px_socket,
h. static SOCKET px_socketL,

2. Методе:

a. void SendInitialMsgToServer(SOCKET *server_socket, sockaddr_in *sraddr)
b. void SendQuitMsgToServer(SOCKET *server_socket, sockaddr_in *sraddr)
c. void SetBuffer(BYTE *buffer)
d. DWORD WINAPI PortSignallingThread(LPVOID lpParam)
e. DWORD WINAPI ClientSocketThread(LPVOID lpParam)
f. DWORD WINAPI ServerSocketThread(LPVOID lpParam)
g. void GetIPFromStruct(sockaddr_in *ipStruct, BYTE *ipAddress)
h. void CloseDITGSockets()

String_Parser.cpp садржи следеће методе:

1. Методе:

a. int compareString(char *originalString, char *compareString, int *beginPosition, int *endPosition)
b. int getContentLength(char *originalString, unsigned int *contentLength)
c. int changeHostAddress(char *originalString, char *newString)
d. int getNumberOfFiles(char *originalString)

4.1 USER _Automat.cpp

4.1.1 void GetCommandFromUser(char *input, char *paramAddress, char *paramDITG, char *paramOtherTraffic, int *quitFlag)

Користи се за пријем команди од корисника на основу којих се покреће рад система аутомата (прелази из стања *IDLE* у стање *WORK*) или се евентуелно прекида рад (ако претходно већ није био покренут).

Параметри:

input – команда коју систем прима од корисника (може бити *start* или *quit*);
paramAddress – адреса сервера на коју сервер посредник треба да се повеже
paramDITG – омогућавање мерења саобраћаја на серверу посреднику
paramOtherTraffic – омогућавање додатног саобраћаја кроз сервер посредник
quitFlag – ознака која обавештава о прекиду рада програма.

4.1.2 void AcceptConnection()

Функција се позива када USER_Automat добије поруку од TCP_Automat-а о успостављеној вези са сервером, након чега шаље поруку TCP_Automat-у да слуша пролаз ради евентуелног успостављања везе са пријемником.

4.1.3 void ListenClientSocket()

Функција се позива након што се успостави веза са клијентом, после чега USER_Automat шаље поруку TCP_Automat-у како би он могао да слуша утичницу према пријемнику зарад пријема *HTTP* захтева од њега.

4.1.4 void parseHTTPRequest()

TCP_Automat када прими *HTTP* захтев од клијента, проследи га USER_Automat-у који измени заглавље захтева где се наводи адреса и пролаз сервера посредника, на његово место ставља адресу и пролаз стварног сервера. Измењени *HTTP* захтев шаље TCP_Automat-у, који га затим прослеђује серверу.

4.1.5 void parseHTTPResponse()

TCP_Automat прими део *HTTP* одговора од предајника, односно део заглавље поруке са подацима о датотеци која се шаље. Тај део се проследи USER_Automat-у који обавештава корисника о пријему одговора од предајника, шаље поруку TCP_Automat-у, како би он могао врши пријем података од предајника и слање ка пријемнику.

4.1.6 void restartSistem()

Шаље поруку TCP_Automat-у зарад поновног успостављања везе са предајником као и слушање пролаза на којем очекује пријемнике и пријем захтева од пријемника.

4.1.7 void QuitUser()

Позива се када се систему проследи параметар за завршетак рада аутомата и аутомат прелази у стање *QUIT*.

4.1.8 void ErrorProcessing()

Уколико TCP_Automat има проблема са успостављањем везе са сервером, или клијент пошаље *HTTP* захтев са неисправним називом датотеке, обавештава USER_Automat слањем поруке о насталој грешци, као и са параметром грешке. Испише се која је грешка у питању и прекида се рад система аутомата.

4.2 TCP_Automat.cpp

4.2.1 void StartTCP()

Након пријема поруке од USER_Automat-а за покретање TCP_Automat-а, подешава потребне параметре у циљу успостављања везе са предајником и аутомат прелази из стања *IDLE* у стање *QUIT*.

4.2.2 void QuitTCP()

Након пријема поруке од USER_Automat-а за прекид рада TCP_Automat-а, из тренутног стања прелази у стање *QUIT*.

4.2.3 void ConnectToServer()

Попуњавају се одређене структуре са подацима сервера, сервера посредника и клијента, као и повезивање са сервером. У случају успешног повезивања са сервером, о томе обавештава USER_Automat-у, а ако постоји проблем приликом повезивања, тада се шаље порука USER_Automat-у о грешци.

4.2.4 void AcceptConnectionFromClient()

Врши се подешавање пролаза на којем сервер посредник очекује клијенте. Након што се подесе сви параметри, креира се нит задужена за успостављање везе са клијентом.

4.2.5 static DWORD WINAPI AcceptConnectionFromClientThread(LPVOID lpParam)

Функција нити која је задужена за слушање пролаза на којем сервер посредник очекује клијенте. Након што се прихвати веза од стране клијента, шаље се порука USER_Automat-у, који се обавештава о успешно успостављеној вези.

4.2.6 void ListenClientSocket()

Користи се за слушање утичнице на којој сервер посредник очекује поруке од клијента. Када сервер посредник прими *HTTP* захтев од клијента, прослеђује је USER_Automat-у, који врши обраду тог *HTTP* захтева.

4.2.7 void SendHTTPRequestToServer()

Користи се за слање *HTTP* захтева примљен од клијента ка серверу, као и пријем *HTTP* одговора(заглавља) од сервера. Ако клијентов захтев садржи назив листе, од сервера ће добити одговор и списак датотека из листе.

4.2.8 void sendToClient()

Врши пријем преосталог *HTTP* одговора од сервера и прослеђује га клијенту. У зависности од параметара које је корисник проследио систему аутомата, могуће је измерити текући саобраћај.

4.2.9 static DWORD WINAPI DITGProxyThread(LPVOID lpParam)

Креира нити за успостављање везе са *ITGSend* и *ITGRecv*, као и за размену података између њих, односно представља на неки начин сервер посредник између њих.

4.3 TCP_File.cpp

4.3.1 void SendInitialMsgToServer(SOCKET *server_socket, sockaddr_in *sraddr)

Користи се како би се послале иницијалне поруке према *ITGRecv*. Након успешно послатих порука могуће је отворити линију за податке којом се шаљу пакети према *ITGRecv* да би се саобраћај касније могао анализирати и да би се могле одредити карактеристике мреже.

Параметри:

server_socket – утичница према серверу; *sraddr* – структура која садржи податке о серверу (адреса, пролаз).

4.3.2 void SendQuitMsgToServer(SOCKET *server_socket, sockaddr_in *sraddr)

Користи се како би се послала порука *ITGRecv* о завршетку слања података, односно пакета, након чега се раскида веза између њих.

Параметри:

server_socket – утичница према серверу; *sraddr* – структура која садржи податке о серверу (адреса, пролаз).

4.3.3 void SetBuffer(BYTE *buffer)

Служи за уписивање одређених вредности у прослеђени бафер.

Параметри:

buffer – бафер у који треба уписати скуп одређених вредности

4.3.4 DWORD WINAPI PortSignallingThread(LPVOID lpParam)

Користи се за успостављање везе са *ITGRecv* , у зависности од параметара. Потребно је отворити две линије, једна је за сигналне поруке, а друга је за податке.

Параметри:

lpParam – ако се проследи 1, отвара се линија за сигналне порука, а ако се проследи 2, онда се отвара линија за податке.

4.3.5 DWORD WINAPI ClientSocketThread(LPVOID lpParam)

Функција нити која се користи зарад пријема података од *ITGRecv* и слања истих ка *ITGSend.*

Параметри:

lpParam – структура која садржи структуре везане за *ITGSend* и *ITGRecv,* као и утичнице према њима

4.3.6 DWORD WINAPI ServerSocketThread(LPVOID lpParam)

Функција нити која се користи зарад пријема података од *ITGSend* и слања истих ка *ITGRecv.*

Параметри:

lpParam – структура која садржи структуре везане за *ITGSend* и *ITGRecv,* као и утичнице према њима.

4.3.7 void GetIPFromStruct(sockaddr_in *ipStruct, BYTE *ipAddress)

Користи се за извлачење адресе из структуре у облику низа од 4 вредности.

Параметри:

ipStruct – структура која садржи адресу; *ipAddress* – низ који ће се попуњавати са вредностима адресе.

4.3.8 void CloseDITGSockets()

Користи се за затварање утичница према *ITGSend* и *ITGRecv* и прекидање веза са њима.

4.4 String_Parser.cpp

4.4.1 int compareString(char *originalString, char *compareString, int *beginPosition, int *endPosition)

Врши проверу да ли се један стринг налази у другом стрингу, ако се налази одређује вредности почетка и краја садржања једног стринга у другом.

Параметри:

originalString – оригинални стринг,

compareString – стринг који се испитује да ли се налази у оригиналном стрингу,

beginPosition – почетна позиција садржавања прослеђеног стринга у оригиналном стрингу,

endPosition – крајња позиција садржавања прослеђеног стринга у оригиналном стрингу.

4.4.2 int getContentLength(char *originalString, unsigned int *contentLength)

Из прослеђеног стринга извлачи информацију о величини података која треба да се проследи од сервера, преко сервера посредника ка клијенту.

Параметри:

originalString – стринг који садржи величину података,

contentLength – адреса у коју се смешта вредност величине података за слање.

4.4.3 int changeHostAddress(char *originalString, char *newString)

Врши измену ХТТП поруке коју сервер посредник прими од клијента, део у којем се налази адреса и пролаз сервера посредника замењује са адресом и пролазом правог сервера.

Параметри:

originalString – стринг са адресом и пролазом сервера посредника,

newString – измењени стринг са адресом и пролазом стварног сервера.

4.4.4 int getNumberOfFiles(char *originalString)

Из прослеђеног списка датотека одређује се број датотека у листи.

Параметри:

originalString - стринг који садржи називе датотека.

Повратна вредност: број датотека

5. Резултати

Систем је тестиран за пар различитих случајева:

1. Када не постоји додатан саобраћај у мрежи,
2. Када је мрежа оптерећена минималним саобраћајем,
3. Када је мрежа оптерећена средњим саобраћајем,
4. Када је мрежа преоптерећена.

Приликом тестирања коришћено је пар видео датотека са различитим карактеристикама – резолуција, битска брзина, број слика у секунди. Карактеристике коришћених видео датотека су дате у следећој табели:

рд. бр.	назив датотеке	трајање	величина	резолуција	битска брзина
1	testing_video1.mpg	0:00:30	4.98MB	352x288	1374kbps
2	testing_video2.mp4	0:00:30	11.7MB	1280x720	3015kbps
3	testing_video3.mp4	0:01:00	22.4MB	1280x720	2948kbps
4	testing_video4.mp4	0:01:00	5.24MB	640x360	714kbps

Табела 5.1 Карактеристике видео датотека коришћених за тестирање могућности система

На основу тих видео датотека добијени су следећи резултати који су приказани у табелама, за сваку видео датотеку понаособ:

рд. бр.	тип вредности	без оптерећења	аналогија	са малим оптерећењем	аналогија	са средњим оптерећењем	аналогија	са великим оптерећењем	аналогија
1	Total time(s):	28.521	28.510	30.462	30.435	31.957	31.888	33.961	33.895
2	Total packets:	1727	1727	1727	1727	1727	1727	1727	1727
3	Minimum delay(s):	/	0.002000	/	0.004000	/	0.011000	/	0.012000
4	Maximum delay(s):	/	0.061000	/	0.115000	/	0.083000	/	0.124000
5	Average delay(s):	/	0.000188	/	0.010754	/	0.022831	/	0.024413
6	Average jitter(s):	/	0.000941	/	0.002865	/	0.004961	/	0.005147
7	Delay standard deviation(s):	/	0.002716	/	0.005707	/	0.005632	/	0.007135
8	Bytes Received	5228552	5229356	5228552	5229356	5228552	5229356	5228552	5229356
9	Average Bitrate(Kbit/s):	1466.583	1467.375	1373.134	1374.564	1308.897	1311.931	1231.660	1234.248
10	Average Packet rate(Pkt/s):	60.552	60.575	56.694	56.744	54.041	54.158	50.852	50.951

Табела 5.2 Резултати мерења за датотеку *testing_video1.mpg*

рд. бр.	тип вредности	без оптерећења	аналогија	са малим оптерећењем	аналогија	са средњим оптерећењем	аналогија	са великим оптерећењем	аналогија
1	Total time(s):	29.050	29.059	28.875	29.176	38.697	38.761	64.164	64.111
2	Total packets:	4056	4055	4056	4056	4056	4056	4056	4056
3	Minimum delay(s):	/	0.002000	/	0.008000	/	0.014000	/	0.006000
4	Maximum delay(s):	/	0.062000	/	0.327000	/	0.148000	/	0.107000
5	Average delay(s):	/	0.001218	/	0.015665	/	0.034520	/	0.021452
6	Average jitter(s):	/	0.001521	/	0.003849	/	0.005915	/	0.004873
7	Delay standard deviation(s):	/	0.002709	/	0.006723	/	0.007904	/	0.006197
8	Bytes Received	12281478	12278540	12281478	12281568	12281478	12281568	12281478	12281568
9	Average Bitrate(Kbit/s):	3382.163	3380.306	3402.661	3367.581	2539.004	2534.830	1531.261	1532.538
10	Average Packet rate(Pkt/s):	139.623	139.543	140.468	139.018	104.814	104.641	63.213	63.265

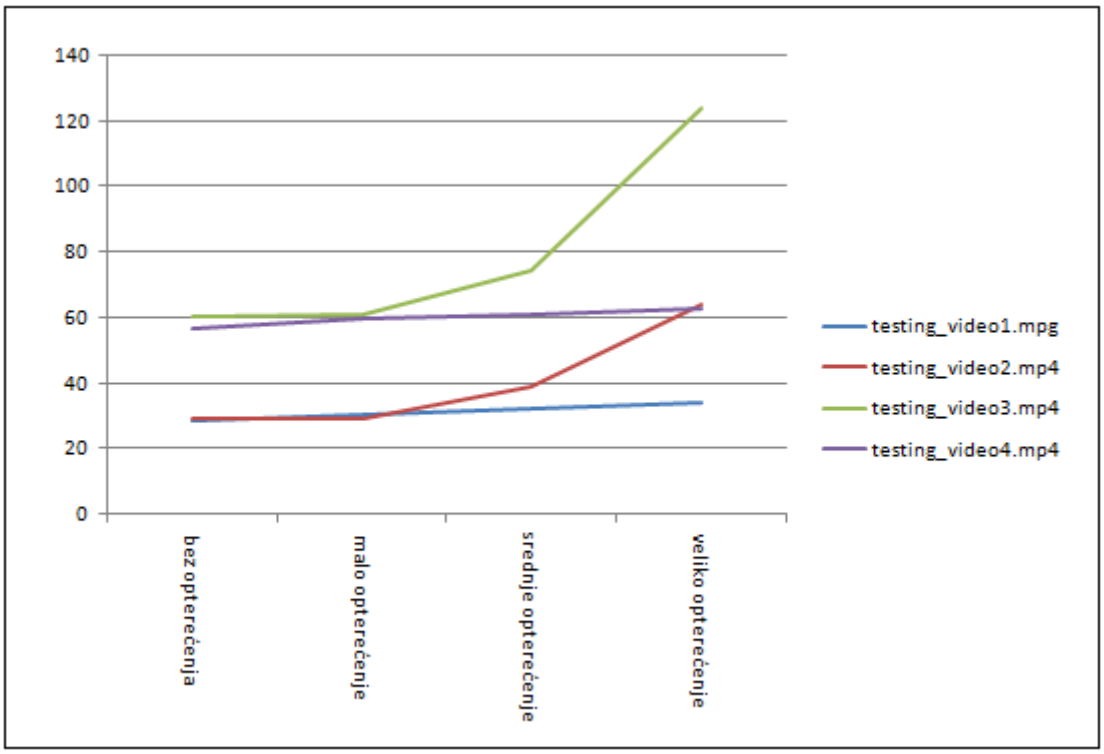
Табела 5.3 Резултати мерења за датотеку *testing_video2.mp4*

рд. бр.	тип вредности	без оптерећења	аналогија	са малим оптерећењем	аналогија	са средњим оптерећењем	аналогија	са великим оптерећењем	аналогија
1	Total time(s):	60.537	60.773	61.289	62.007	74.618	74.500	123.579	123.529
2	Total packets:	7800	7800	7750	7750	7750	7749	7800	7800
3	Minimum delay(s):	/	0.002000	/	0.004000	/	0.009000	/	0.014000
4	Maximum delay(s):	/	0.235000	/	0.738000	/	0.151000	/	0.137000
5	Average delay(s):	/	0.001308	/	0.012847	/	0.028212	/	0.031196
6	Average jitter(s):	/	0.001484	/	0.003964	/	0.005483	/	0.004776
7	Delay standard deviation(s):	/	0.004575	/	0.009728	/	0.007499	/	0.006797
8	Bytes Received	23618400	23618400	23467000	23467000	23467000	23463972	23618400	23618400
9	Average Bitrate(Kbit/s):	3121.185	3109.065	3063.127	3027.658	2515.961	2.517.931	1528.9588	1529.578
10	Average Packet rate(Pkt/s):	128.847	128.346	126.450	124.986	103.862	103.944	63.118	63.143

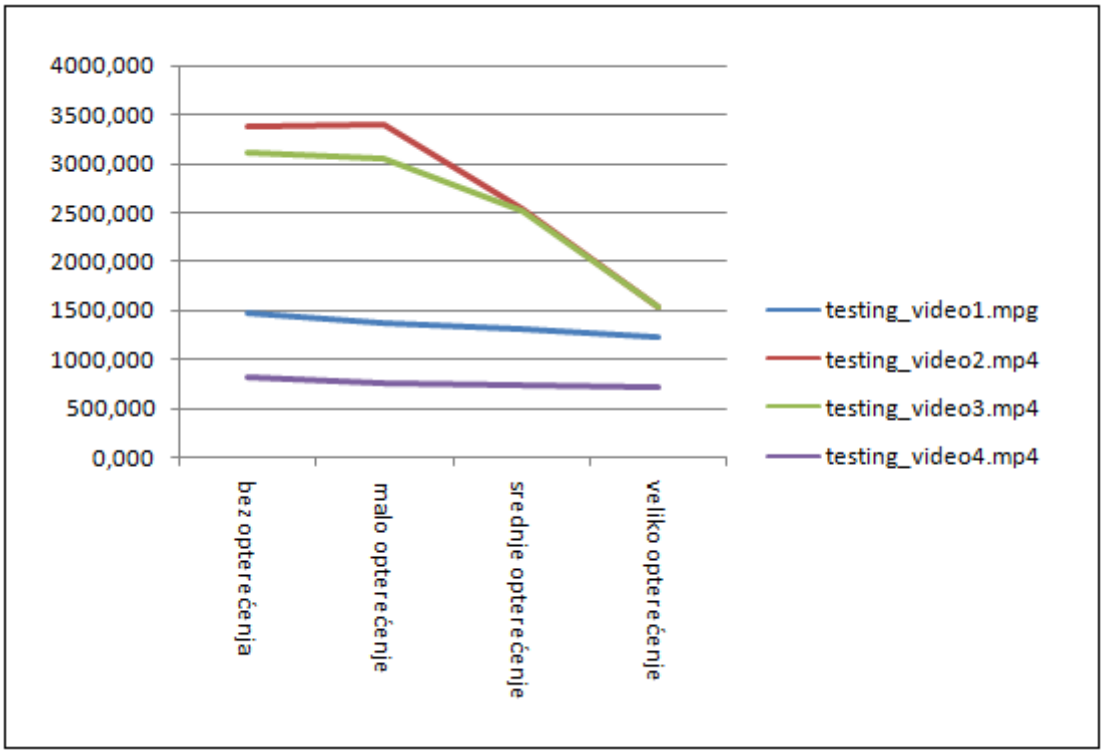
Табела 5.4 Резултати мерења за датотеку *testing_video3.mp4*

рд. бр.	тип вредности	без оптерећења	аналогија	са малим оптерећењем	аналогија	са средњим оптерећењем	аналогија	са великим оптерећењем	аналогија
1	Total time(s):	56.576	56.543	59.895	59.848	60.624	60.563	62.787	62.754
2	Total packets:	1900	1900	1850	1850	1850	1850	1850	1850
3	Minimum delay(s):	/	0.002000	/	0.006000	/	0.005000	/	0.008000
4	Maximum delay(s):	/	0.102000	/	0.080000	/	0.078000	/	0.170000
5	Average delay(s):	/	0.000308	/	0.011505	/	0.018883	/	0.022208
6	Average jitter(s):	/	0.001077	/	0.003191	/	0.005516	/	0.006062
7	Delay standard deviation(s):	/	0.004553	/	0.004240	/	0.005173	/	0.011322
8	Bytes Received	5753200	5753200	5601800	5601800	5601800	5601800	5601800	5601800
9	Average Bitrate(Kbit/s):	813.518	813.993	748.216	748.804	739.219	739.963	711.180	713.462
10	Average Packet rate(Pkt/s):	33.583	33.603	30.887	30.912	30.516	30.547	29.385	29.480

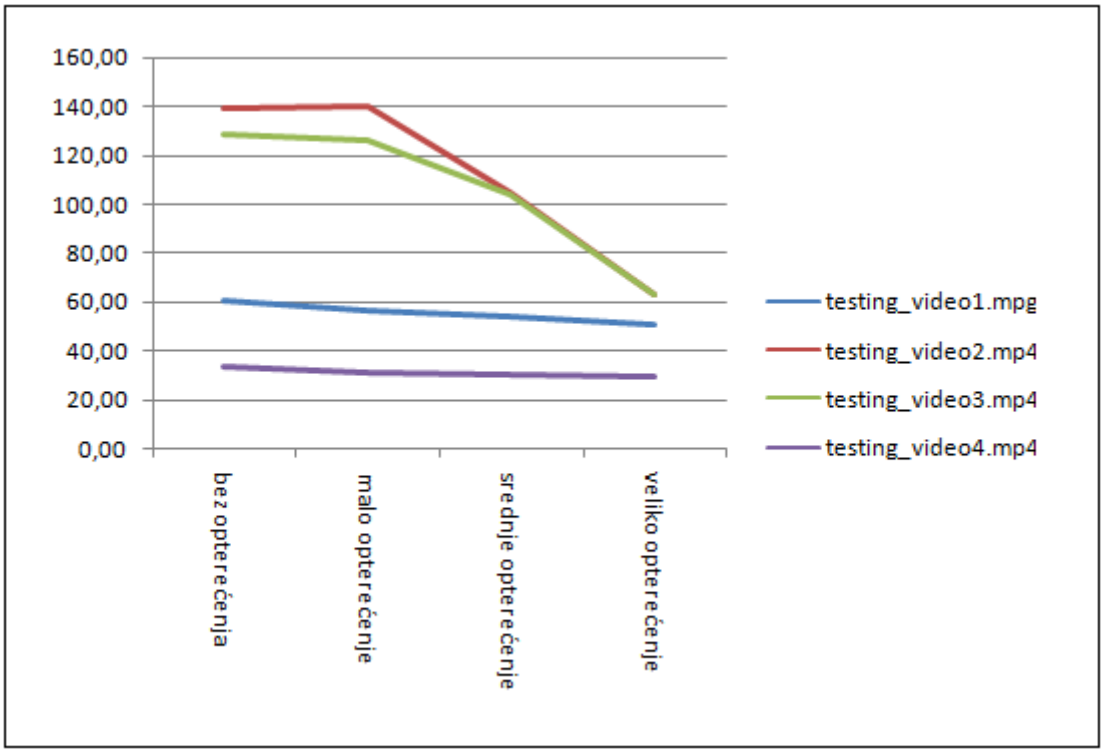
Табела 5.5 Резултати мерења за датотеку *testing_video4.mp4*



Слика 5.1 Утицај оптерећења мреже на време слања тока података



Слика 5.2 Утицај оптерећења мреже на битску брзину слања тока података



Слика 5.3 Утицај оптерећења мреже на густину слања пакета

Може се приметити да повећање додатног саобраћаја највише утиче на слање видео датотека са већим битским брзинама, док се код видео датотека са смањеним битским брзинама, тј. са смањеном количином информација са којом располажу не примете драстичне разлике са повећањем саобраћаја. Повећање саобраћаја има за последицу повећање времена слања података према клијенту, односно смањење оригиналне битске брзине, којом се подаци шаљу, па се јављају краткотрајне паузе и прекиди између емитовања, односно нарушен је визуелни изглед. Повећање интензитета оптерећујућег саобраћаја утиче на повећање кашњења, временских изобличења и осталих штетних фактора који утичу на квалитет везе, у зависности од карактеристика аудио и видео датотека.

Повећање оптерећујућег саобраћаја је имало великог утицаја на рад реализованог система, али није довело до насилног прекида рада система, односно систем је успео да испуни захтеве који су били постављени пред њим, за различите услове и стања мреже којој припада.

6. Закључак

У раду је реализовано једно решење симулатора ИП мреже у циљу тестирања модерних дигиталних ТВ пријемника, чији је задатак генерисање, мерење и анализа саобраћаја у локалној мрежи. Делови симулатора ИП мреже представљају систем клијент – сервер посредник – сервер, чија комуникација се заснива на *HTTP Live* протоколу, као и систем за мерење и анализу саобраћаја, и за генерисање додатног, оптерећујућег саобраћаја, што је утицало на тестирање система у различитим условима и стањима мреже.

Сервер посредник је реализован на основу дијаграма размене порука између одређеног клијента и одређеног сервера, тако да би за неке друге програмске алате вероватно постојала потреба за изменом модела система – у суштини , размена порука би била идентична.

За потребе овог рада урађено је упознавање са основама *HTTP Live* протокола, принципима рада *D-ITG, Sirannon* и *FFmpeg* програмских пакета, реализован је сервер посредник за прослеђивање података између клијента и сервера на основу дијаграма размене порука, сервер посредник за програме из пакета *D-ITG,* и урађено је тестирање система за различите аудио и видео садржаје, на основу чега се види понашање система на основу измерених података.

Предност система се огледа у самој робусности система, додатку дела за мерење саобраћаја, као и у програмском решењу, где је обраћена пажња на евентуелне грешке које би могле настати приликом повезивања са сервером или у случају неисправног назива датотеке.

Мане реализованог система су :

1. Систем може да измери само битску брзину, као и брзину слања пакета, док се остали резултати добијају на основу аналогије са референтним системом.

2. Ограничен је само на један комуникациони протокол, у случају употребе система за рад са другим комуникационим протоколом, требало би мењати цео систем.
3. Не постоји могућност истовременог повезивања више клијената на систем.
4. Систем не ради у ситуацијама када клијент наведе у свом захтеву назив листе са датотекама уместо назива датотеке.
5. Такође, постоји проблем када се јавља потреба за мерењем података током већег временског интервала. Наиме, елемент задужен за прикупљање и анализу података, током дужег временског интервала не може да буде у стању да прими сигнал од сервера посредника за прекид прикупљања и мерења, што доводи до потребе за насилним прекидом апликације – ипак, резултати ће постојати и након тога, неће се изгубити, нити оштетити.

Правци даљег рада су следећи:

1. Убацити временско печаћење у пакете који се крећу од сервера посредника ка елементу за анализу саобраћаја, како би се могли добити исправни подаци, без да се користи аналогија,
2. Потребно је радити на робусности система, да би систем успешно радио када му клијент у захтеву наведе назив листе,
3. Радити на функционалности елемента за пријем и мерење саобраћаја.

7. Литература

[1]…….. http://ffmpeg.org/index.html

[2]…… http://sirannon.atlantis.ugent.be/

[3]…… http://www.grid.unina.it/software/ITG/

[4]…….http://en.wikipedia.org/wiki/HTTP_Live_Streaming

[5]…..…. http://www.rt-rk.uns.ac.rs/