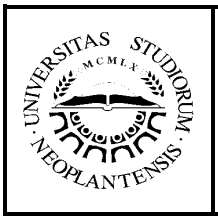


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

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: **Ђорђе Ђокић**
Број индекса: **РА100-2017**

Тема рада: **Имплементација програмског рјешења за конверзију дигиталног телевизијског сигнала у ХЛС формат дистрибуције**

Ментор рада: **Проф. Др. Илија Башићевић**

Нови Сад, фебруар, 2023.

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**:		Ђорђе Ђокић
Ментор, **МН**:		Проф. Др Илија Башићевић
Наслов рада, **НР**:		Имплементација програмског ријешења за конверзију дигиталног телевизијског сигнала у ХЛС формат дистрибуције
Језик публикације, **ЈП**:		Српски / латиница
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2023.
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**:		Дигитална телевизија
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		У овом раду приказано је једно програмско рјешење обраде дигиталног телевизијског сигнала, тачније МПЕГ—ТС пакета и његово прилагођавање ХЛС формату дистрибуције. Представљено је сегментирање појединачних ТС пакета у сегменте погодније за обраду. Са друге стране, направљен је ХТТП сервер, који прима ХЛС захтјеве од стране клијента и шаље ХЛС одговоре, у виду сегментираних пакета.
Датум прихватања теме, **ДП**:		20.06.2022.
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	
	Члан:	Потпис ментора
	Члан, ментор:	

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**:		Đorđe Đokić	
Mentor, **MN**:		Prof. Dr Ilija Bašićević	
Title, **TI**:		Implementation of software solution for conversion of digital television signal into HLS format of distribution	
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**:		2023.	
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**:		Digital television	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This paper presents one solution for processing digital television signal, more precisely MPEG-TS packages, and his adaptation for HLS format distribution. It represent segmentation of singles TS package into segment which easier to process. On the other hand, HTTP server, which is working with HLS protocol, send and receive segmented packages.	
Accepted by the Scientific Board on, **ASB**:		20.06.2022.	
Defended on, **DE**:			
Defended Board, **DB**:	President:		
	Member:		Menthor's sign
	Member, Mentor:		

Захвалност

Велику захвалност за помоћ при изради документације везане за дипломски рад дугујем свој академском ментору, Проф. Др Илији Башићевићу. Такође, огромна захвалност мом техничком ментору, Раденку Бановићу, који ми је био на располагању током практичне израде дипломског рада, али и свим колегама у оквиру *iWedia* групе.

Наравно, посебну захвалност дугујем својој породици, на огромној подршци за вријеме израде рада, а поготово током цјелокупног школовања.

САДРЖАЈ

Списак слика

Скраћенице

MPEG – *Moving Picture Experts Group,* група за стандатдизацију дигиталних формата

TS – *Transport Stream,* транспортни ток

HTTP – *HyperText Transfer Protocol,* протокол за пренос хипертекста

HLS – *HTTP Live Streaming,* протокол за пренос хипертекста уживо

DTV *– Digital television,* дигитална телевизија

DOA – *Direction Of Arrival,* путања доласка

DVB – *Digital Video Broadcasting,* дигитални видео пренос

ATSC – *Advanced Television System Committee,* напрадни комитет телевизијског система

IPTV – *Internet Protocol Television,* интернет протокол за телевизију

OSI – *Open Systems Interconnection model,* референтни модел за отворено повезивање система

HTML – *HyperText Markup Language,* језик за означавање хипертекста

TCP – *Transmission Control Protocol,* трансмициони контролни протокол

IP – *Internet Protocol,* интернет протокол

UTF-8 *– Unicode Transformation Format – 8-bit,* осмобитни кодирани транпортни формат

USB *– universal serial bus,* универзална серијска магистрала

PCR *– Program clock reference,* референца програмског сата

1. Увод

Телевизија представља електронски систем помоћу кога оптичку слику и звук претварамо у електронске сигнале, који се преносе до пријемника, гдје се претварају у оптичку слику и звук. У данашње вријеме, телевизија је једна од технологија која се најбрже развија. Највећу екпанзију у досадашњем развоју телевизије доживио је начин преноса сигнала. Те промјене се огледају у транзицији првобитних аналогних на дигиталне сигнале. Као и свака нова напредна технологија и дигитална телевизија је са собом донијела низ предности и отворила пут новим могућностима за даљи развој.

Систем „Смарт Антена“ представља једно од рјешења за адресирање оптимизације пријема дигиталног телевизијског пријемника у аутомобилу. Развојем интернета неколико новијих стандарда су преузели примат у дистрибуцији садржаја, а један од њих је *HLS* стандард. Као резултат овога велика већина модерних рјешења за репродукцију садржаја је базирано на овом стандарду (*Netflix*, *Amazon Prime*, *Apple TV*, *ExoPlayer*…). У циљу проширења компатибилности циљ рада је проширење програмског рјешења Смарт Антена уређаја опцијом да се садржај дистрибуира путем *HTTP* сервера и коришћењем HLS протокола. Рад је подјељен у неколико фаза:

У првој фази је имплементиратно прихватање и сегментирање података (подаци се примају у *MPEG-TS* формату). Неколико *MPEG-TS* пакета формира један *HLS* сегмент.

Друга фаза је представљала креирање *HTTP* сервера који заприма захтјеве од стране клијената те као одговор шаље захтјевани *HLS* сегмент.

Трећа фаза је креирање тестног окружења којим је потребно верификовати функционалност рјешења. Тестирање се врши коришћењем комерцијалних пријемника *HLS* сегмената, као и креирање тестне апликације.

Рад је организован у слиједеће цјелине:

- Теоријске основе – преглед технологија коришћених у изради задатка.
- Концепт рјешења – идејни опис рјешења.
- Програмско рјешење – детаљан опис имплементације рјешења.
- Испитивање реализованог система – приказ резултата тестирања.
- Закључак – опис испуњености задатка и приједлози будућег унапријеђења

2. Теоријске основе

У овом поглављу биће описани системи, протоколи као и програмске подршке које су коришћене приликом саме израде задатка.

2.1 Систем "Смарт антена"

Многи за смарт антене кажу паметне антене, али у стварности антене саме по себи не могу бити паметне. Могућност дигиталне обраде сигнала заједно са антенама чине овај систем "паметним". Иако изгледа да паметне антене представљају нову технологију, основни принципи на којима се заснивају нису нови и датирају из седамдесетих година прошлог вијека.

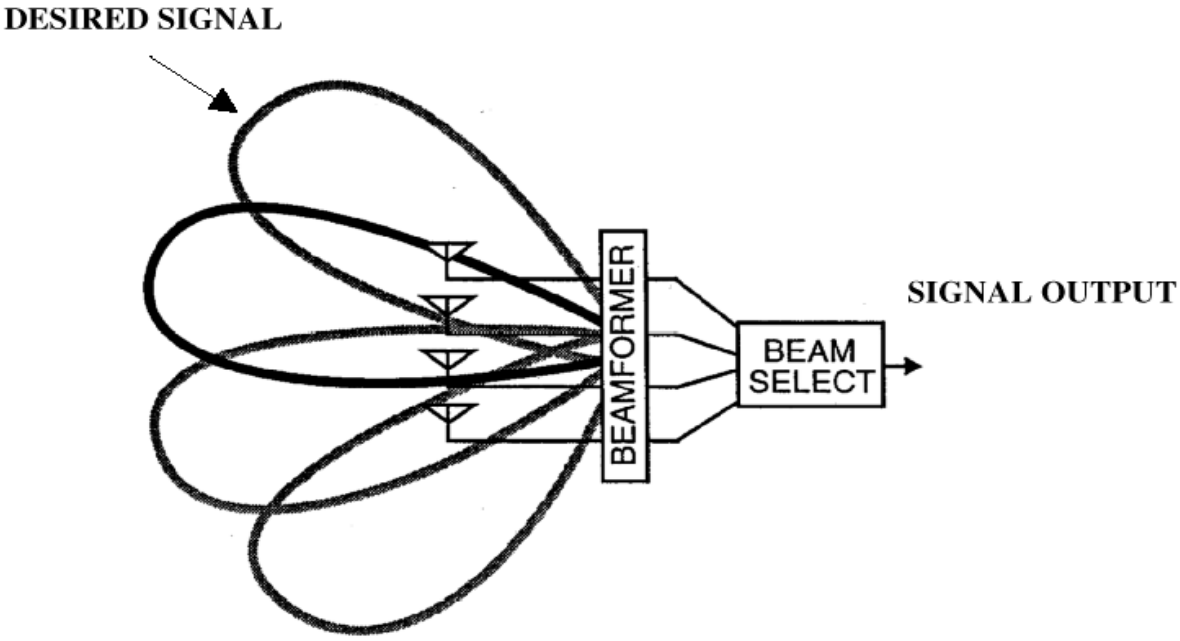
Паметне антене представљају антенски низ са алгоритмима за обраду сигнала у циљу аутоматске оптимизације и прилагођења предајног и/или пријемног дијаграма зрачења антенског низа на конкретни сценарио сигнала. Функционише тако што користи предности ефекта диверзитета примопредајника бежичног система, који је и извор и одредиште. Израз ефекат диверзитета односи се на пренос и пријем више радио-феквенција које се користе за смањење грешке приликом комуникације подацима, као и за повећање брзине примопредаје између извора и одредишта.

Главна разлика у односу на обичне антене је у начину на који се оба система суочавају са проблемима узрокованим ширењем таласа који имају више могућих путева.

Када се бежични сигнал шаље на велику удаљеност, он на том путу прелази многе препреке, као што су високе зграде, планине, други сигнали, итд. То резултује

појављивањем више сигнала на пријему, који могу бити изобличени, непотпуни или одсјечени.

Систем паметне антене ријешава овај проблем својеврсних алгоритама(као што су *Matrix Pencil* method или *Beamforming method*). *Matrix Pencil* метода се своди на то да се антена понаша као сензор у коме се бира просторни спектар низа таласа и *DOA* се сазнаје из врхова овог спектра, док се у *Beamforming* методу прво траже мете на које се сигнали шаљу, а затим се креира дијаграм зрачења антенског низа додавањем фаза сигнала.

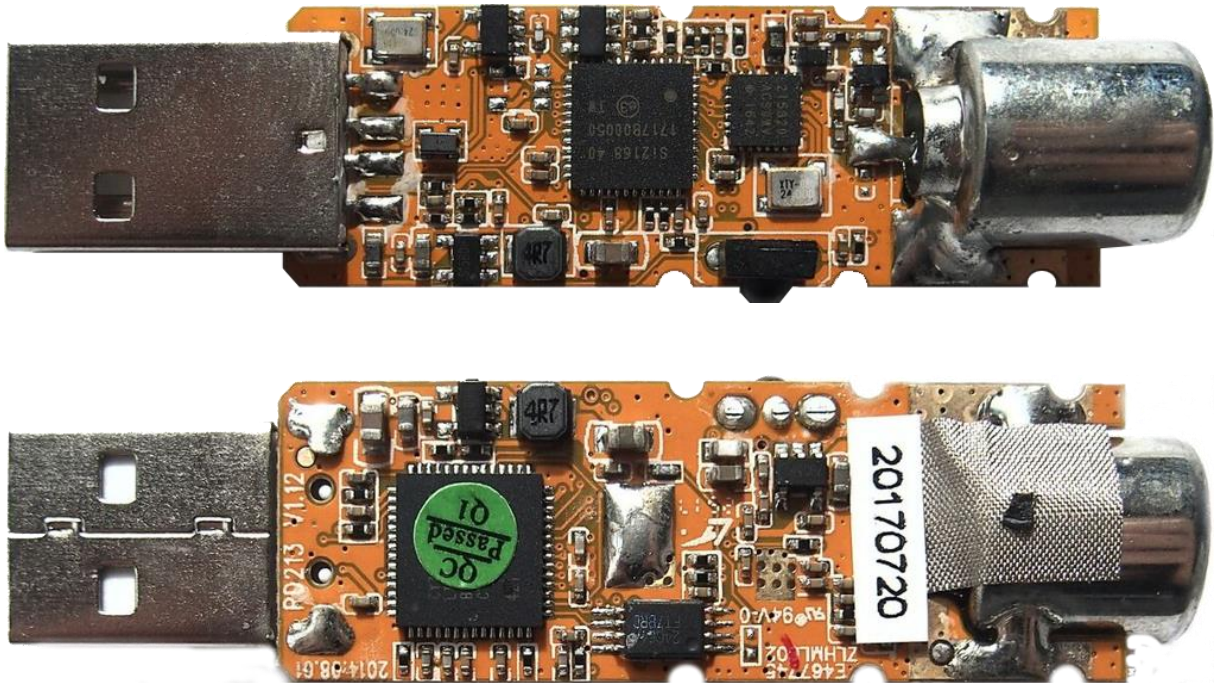


Слика 2.1 – Структура паметне антене

2.2 *USB TV* тунер

ТВ тунер је рачунарска компонента која омогућава пријем телевизијских сигнала путем рачунара.

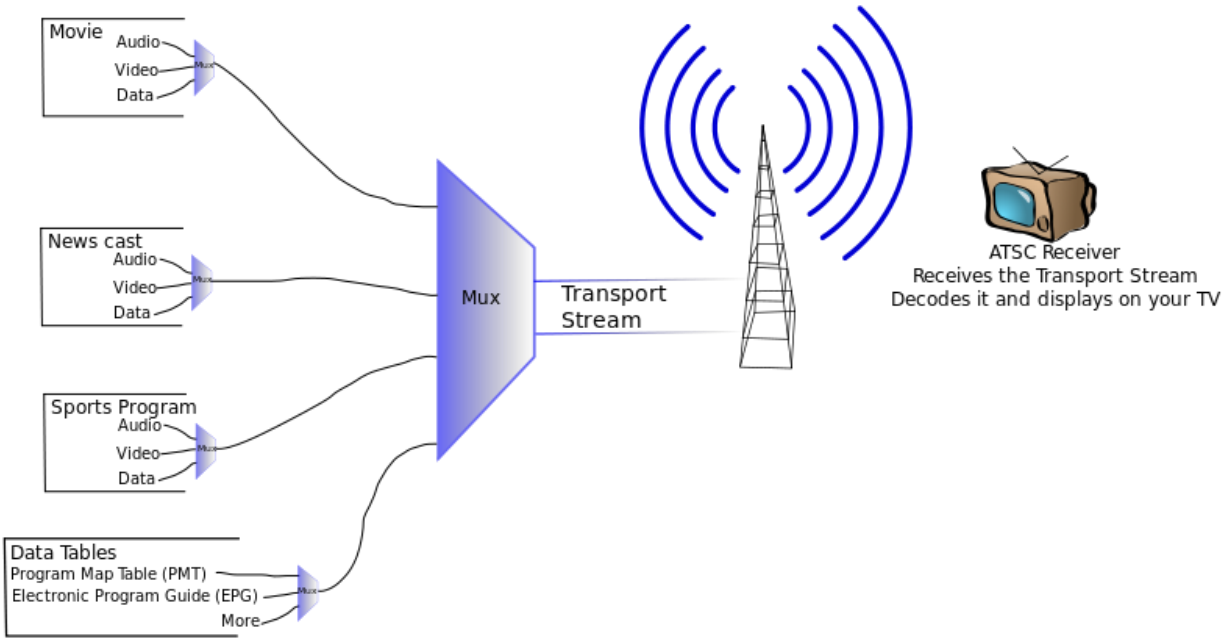
У сврху овог задатка, због немогућности добијања праве паметне антене, коришћен је *Geniatech MyGica USB TV tuner Stick T230A*. ТВ тунер је у овом пројекту коришћен за пријем телевизијског терестриал сигнала који даље обрађујемо.



Слика 2.2 – Попречни пресјек *Geniatech MyGica USB TV* тунера са обје стране

2.3 MPEG-TS

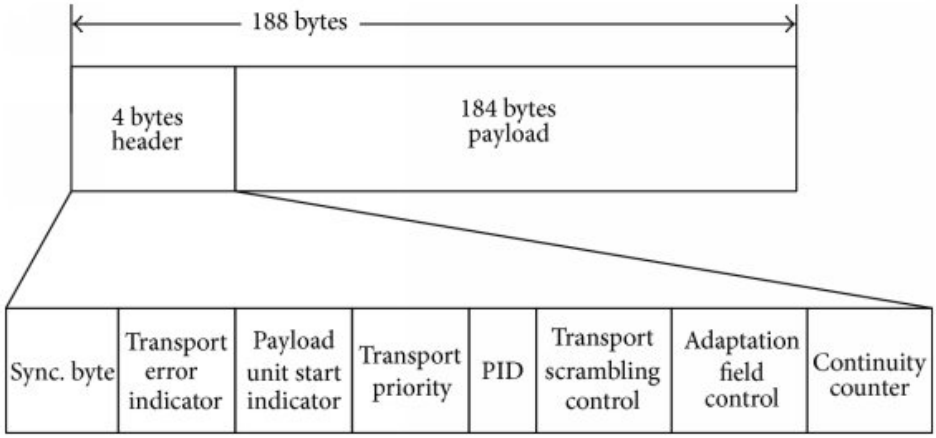
MPEG транспортни ток (*MPEG-TS*, *MTS*) или једноставно транспортни ток (*TS*) је стандардни дигитални формат контејнера за пренос и складиштење аудио-видео записа. Користи се у системима емитовања као што су *DVB*, *ATSC* и *IPTV*.



Слика 2.3 – Креирање и пренос *MPEG-TS* пакета

Транспортни ток специфицира формат контејнера који енкапсулира пакетизоване елементарне токове, са карактеристикама корекције грешака и обрасца синхронизације за одржавање интегритета преноса када је комуникациони канал који преноси ток деградиран.

MPEG-TS пакет се састоји од 188 бајта, од којих 4 бајта чине заглавље, док осталих 184 бајта чине саму информацију. На основу заглавља сазнајемо све информације о датом пакету (као што су њен индентификациони број, да ли је шифрован, колико је корисне информације унутар пакета, да ли се радио о аудиу или видео покатку итд.) које нам могу помоћи о начину на који тај пакет желимо обрадити.

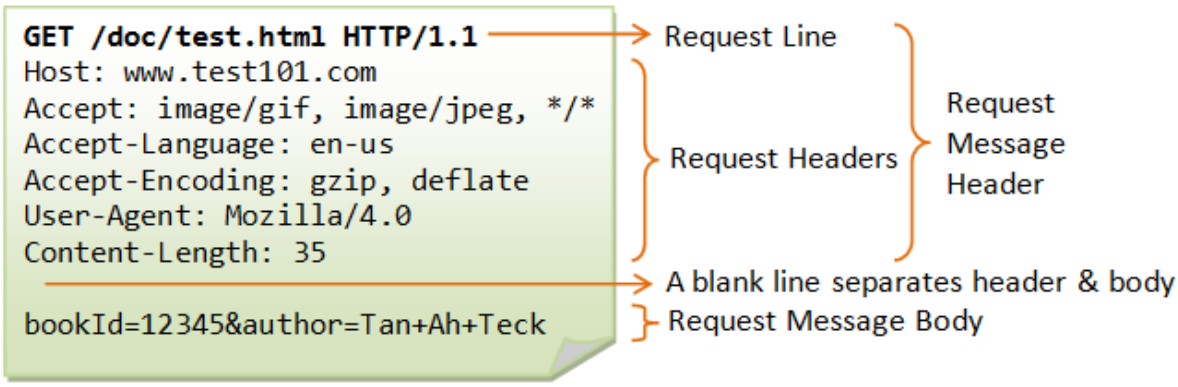


Слика 2.4 – Изглед једног *MPEG-TS* пакета

2.4 *HTTP* протокол

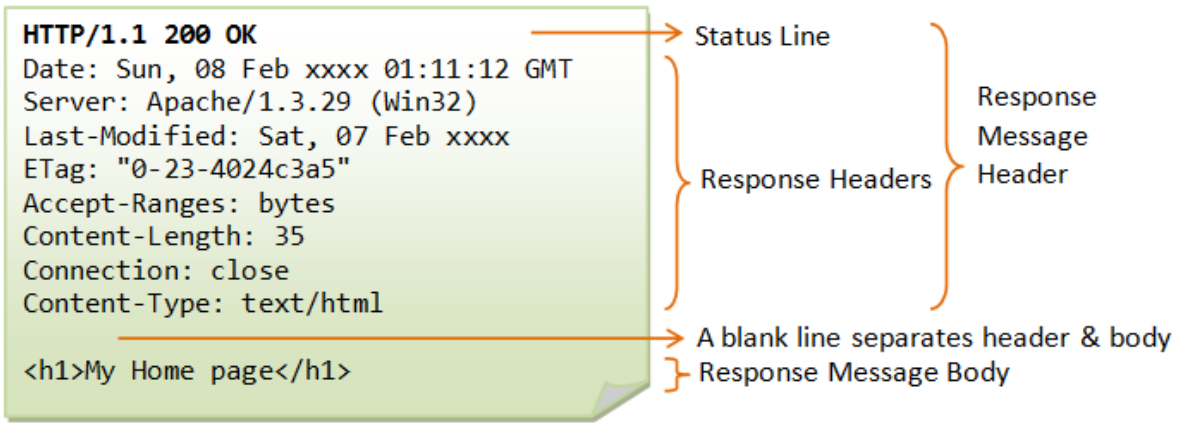
HTTP је најраспрострањенији интернет протокол који се данас користи и који је био неопходан у самој изради овог задатка. Припада апликативном слоју *OSI* референтног модела, а његова основна намјена је испоручивање *HTML* докумената. Комуникација се обавља по систему клијент-сервер. Сервер ослушкује на одређеном порту, а клијент иницира пренос података упостављањем *TCP*/*IP* везе са сервером.

Клијент шаље захтјев који се састоји од заглаља, једне празне линије и тијела захтјева. Дефинисан је низ метода које означавају жељену операцију коју клијент жели да изврши над ресурсима. Најбитнија метода за овај пројекат биће *GET* метода, која врши добављање жељеног ресурса. Још неке од метода које клијент може да ижврши над ресурсима су: *POST*(тражи од сервера да прихвати и најчешће сачува податке у тијелу захтјева), *DELETE*(брисање одређеног ресурса), *PATCH*(модификације постојећег ресурса), итд.



Слика 2.5 – Примјер *GET* методе *HTTP* захтјева клијента

Уколико је захтјев у исправној форми долази до његове обраде на серверу и враћања одговора клијенту. Одговор се, као и захтјев састоји од заглавља, празне линије и тијела одговора(опционо), са обзиром на то да је изглед заглаља другачији у односу на клијентски захтјев. Статусни код *HTTP* одговора представља троцифрени број и кратки описни текст. Најчешћи примјери статусних кодова су: 100 *Continue*, 200 *OK*, 301 *Moved Permanently*, 400 *Bad Request*, 404 *Not Found*, 500 *Internal Server Error*.

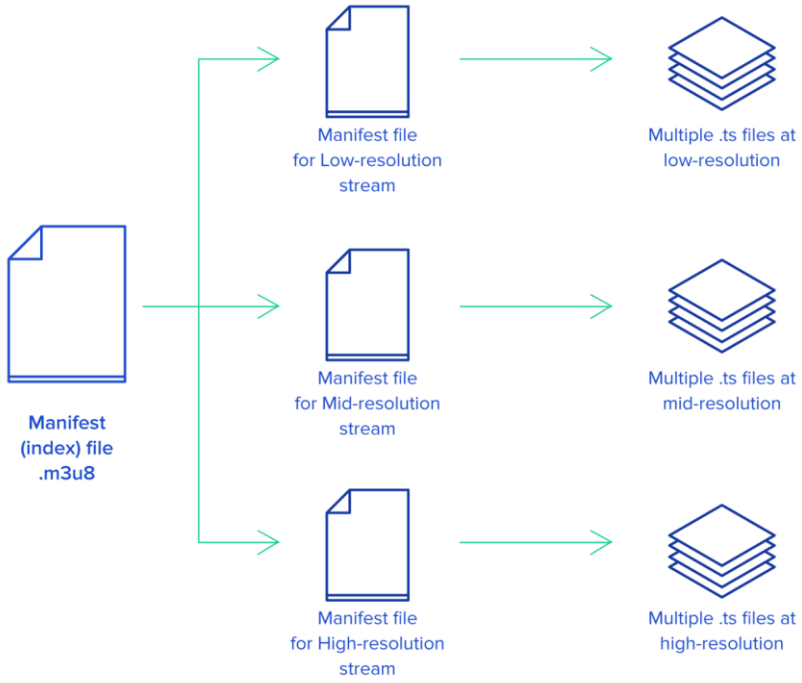


Слика 2.6 – Примјер успјешног одговора од стране *HTTP* сервера

2.5 *HLS* стандард

HLS представља проток за комуникацију са прилагодљивом брзином протокола заснованом на *HTTP*-у. Тренутно има подршку од стране многих медијских плејера, веб претраживача, мобилних уређаја и сервера за стримовање медија.

Функционише тако што разбија цјелокупни ток у низ малих преузимања датотека, заснованих на *HTTP*-у, при чему свако преузима одређени кратак дио укупног потенцијално неограниченог транспортног тока. Листа доступних токова, кодираних са различитим брзинама битова, шаље се клијенту помоћу проширене *M3U8* листе за репродукцију.

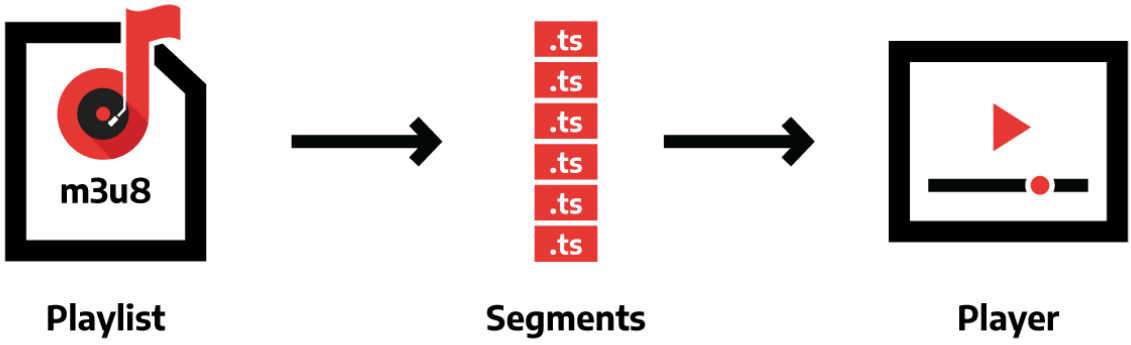


Слика 2.7 – Приказ рада *HLS* стандарда

2.6 *M3U8* фајл

Представља стандардни формат коришћен од стане аудио и видео програма за репродукцију за чување плејлисте. Плејлиста се састоји од интернет веб путања или *URL*-ова, заједно са информацијама о сваком елементу плејлисте, као што је његово трајање. Користи стандардно *UTF*-8 енкодовање.

Временом је схваћено да би ови фајлови могли имати примјену и у стриминг сервисима, уз њихову модулацију. Наиме, када би фајл константно мјењали, односно у њега уносили нове податке и омогућили клијентској апликацији да периодично приступа том фајлу и уочи измјене у њему, то би било адекватно и жељено понашање за видео стримовање.



Слика 2.8 – Поступак конверзије *M3U8* фајла у аудио/видео плејер

2.7 *TSDuck* програмска подршка

TSDuck представља програмску подршку која се користи у дигиталној телевизији. Чини је сет алата који се користе за разне манипулације са *MPEG* преносним токовима. Под поменутим манипулацијама, подразумјевамо и тестирање и надгледање преносног тока података. Такође, користи се и за интеграцију нових софтверских компоненти у систем, као и за њихово тестирање, и отклањање евентуалних грешака.

Имплементација алата је рађена у програмском језику *C++*. Архитектура *TSDuck* програмске подршке је модуларног карактера, што значи да се на релативно лак начин, нова компонента која има одређену функционалност, може интегрисати у систем, тестирати, а у складу са потребама корисника, додатно оптимизовати и прилагођавати крајњој употреби.

2.8 *Mongoose* библиотека

Mongoose је мрежна библиотека за *C/C++*. Она имплементира неблокирајуће *API*-је вођене догађајима за *TCP*, *UDP*, *HTTP*, *WebSocket*, итд. Дизајниран је за повезивање уређаја и њихово довођење на мрежу. На тржишту је од 2004. године, те га користи велики број *open-source* и комерцијалних производа. *Mongoose* чини програмирање уграђене мреже брзим, робусним и лаким.

У овом пројекту коришћено је као „омотач“(*wrapper*) око ког је направљен *HTTP* сервер ради једноставнијег приступа и преношења података.

Библиотека *Mongoose* је добро документована, добро подржана, стабилна и пуна функција. Има доказани запис, очигледна текућа ажурирања и побољшања и функционалност. Након тестирања Mongoose и истраживања карактеристика био је јасан победник. Интеграција је била једноставна и трајала је неколико дана. Време подршке и одзива за све упите је било изванредно.

3. Концепт решења

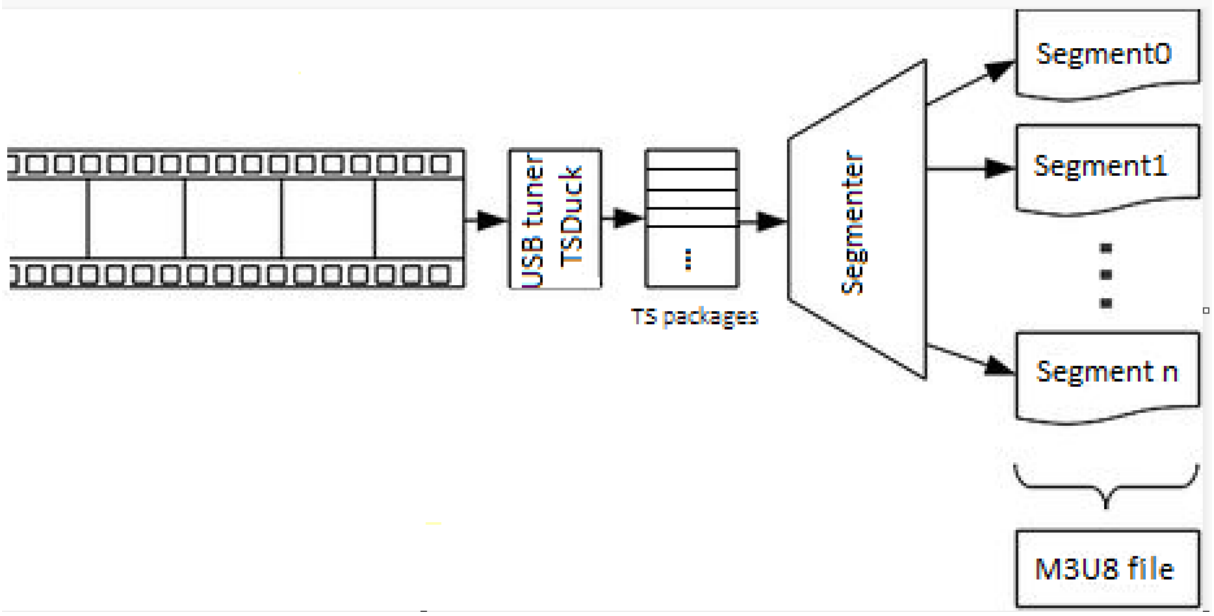
Рјешење је имплементрирано у три корака:

- прихватање и сегментација примљених података
- креирање *HTTP* сервера и механизма одговора на клијентски захтјев
- тестирање и верификација

3.1 Прихватање и сегментација примљених података

Први корак који је било потребно имплемплементирати је прихват *MPEG-TS* пакета. Коришћена је комбинација *TSDuck* и *USB* тунера. Наиме преко *USB* тунера примамо телевизијски сигнал одабиром жељене фреквенције, док преко *TSDuck* програмске подршке успјевамо жељени сигнал прилагодити нашем програму

Затим, како је *HLS* слој изнад *MPEG-TS* слоја(односно *HLS* сегмент енкапсулира неколико *MPEG-TS* пакета који су величине 188 бајта) након прихвата пакета биће потребно наћи одговарајући податак који ће нам одговорити на питање када се завршава један HLS сегмент, а када почиње други. Унутар *MPEG-TS* пакета, налази се информација о томе колико је до датог сегмента прошло врмена, односно *PCR* поље унутар заглавља из кога можемо израчунати тај податак. То нам омогућава да правимо *HLS* сегменте жељеног трајања. За потребе задатка, прављени су сегметни трајања од двије секунде. Након завршетка уписа у један – сегмент, потребно је информацију о том сегменту уписати у *M3U8* фајл који садржу локацију и основне податке о самом сегменту. *M3U8* фајл се стално ажурира, што омогућава стално приказивање нових сегмената.



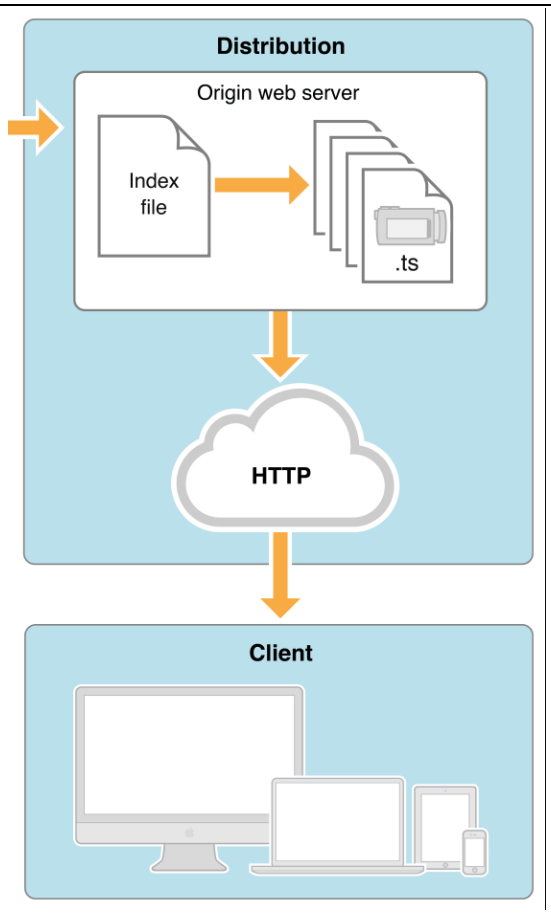
Слика 3.1 – Принцип прихватања и сегментације примљених пакета

Циљ ове фазе био је спремити аудио визуелни садржај за пренос преко интернета коришћењем HTTP протокола.

3.2 Креирање HTTP сервера

Идеја је била да *HTTP* сервер служи за складиштење података те за ослушкивање клијентских захтјева од стране самог плејера и слање захтјеваних података. Увођењем *M3U8* фајла, ово је доста олакшано јер је плејеру само потребно слати ову датотеку, јер ће клијентски плејер захтјевати само њу и из ње извлачити све неопходне информације.

Само креирање сервера је било доста олакшано коришћењем *mongoose* библиотеке која је уграђена у програм. Уз помоћ ње је веома лако отворена конекција између сервера и клијента, те је спремна за складиштење података на серверској страни која ће касније бити кључна за крајње пуштање аудио визуелног садржаја на клијентској страни.



Слика 3.2 – Поступак комуникације између клијента и *HTTP* сервера

Циљ ове фазе био је да након израде пројекта цјелокупан аудио визуелни садржај буде пренесен до клијената чиме је завршен процес израде ријешења.

3.3 Тестирање и верификација

Тестирање и верификација је одрађена кроз двије фазе.

Прва фаза представља сам пренос једног *MPEG-TS* пакета преко *HTTP* сервера, односно провјера ваљаности тог преноса. У ту сврху направљена је тестна клијентска апликација која врши преузимање једног пакета те затим провјерава преузети пакет у односу на оригинални. Након провјере исправности за један пакет, клијентска апликација је проширена да ради са више пакета, појединачно или групно. Када смо били сигурни да се пренесени садржај поклапа са оригиналним на нивоу бајта могли смо потврдити да су пакети успјешно пренесени, те смо прешли на следећу фазу тестирања.

Друга фаза тестирања представља тестирање фајлова који упућују на сегментиране пакете. Ова фаза тестирања подразумјева коришћење комерцијалне апликације за репродукцију садржаја(коришћен *VLC* плејер базиран на *FFMPEG*, као и *iWedia* плејер). У овој фази потребно је без прекида(глатко) и без приказивања нежељених артефаката приказати садржај примљен са *HTTP* сервера.

4. Програмско решење

У овом поглављу описан је начин имплементације рјешења описаног у претходном поглављу. Прво ће бити описано само повезивање уређаја са рачунаром, затим пренос података од пријемника до *HTTP* сервера као и само креирање *HTTP* сервера, синхронизација пристиглих података са пријемника са подацима на серверу, те на крају и тестирање и верификација ваљаности читавог система. За имплементацију је коришћен програмски језик *C/C++*.

4.1 Повезивање *USB* тунера са нашим рачунаром

Као што је речено у претходном поглављу за пријем података коришћен је *USB* тунер. Због тога што је пројека рађен у *Linux* оперативном систему није требало бити потешкоћа са инсталирањем додатних драјвера јер су за *Linux* оперативни систем, тачније за *Ubuntu* преко верзије кернела 3.19, драјвери већ подешени за овај уређај. За верзије испод 3.19 потребно је подесити драјвер, што је на крају морало бити урађено због проблема са самим уређајем, а за подешавање је кориштен линк који се налази у литератури[1].

4.2 Пријем података са *USB* тунера

За пријем података са *USB* тунера коришћена је програмска подршка *TSDuck*. TSDuck нуди изобиље функционалности, као и веома широк ибор параметара зе те

функционалности. Прво је било потребно открити на којој фреквенцији се налазе канали који су доступни преко нашег *USB* тунера. То је урађено покретањем функције ***tsscan*** из терминала, те смо тако установили да је средња фреквенција наших канала 498*MHz*.

Након овога за почетак било је потребно пустити сигнал који стиже са *USB* тунера да би се увјерили у исправност овога података што је урађено следећом функцијом.

- **tsp -I dvb --delivery-system DVB-T2 --bandwidth 8-MHz --frequency 498000000 -O play**

Наиме ова функција као параметре улаза прима информације о томе са које врсте уређаја стиже наш сигнал што је у нашем случају био *dvb,* тачније *DVB-T2* (*-I dvb --delivery-system DVB-T2).* Следећи параметар представља информацију о опсегу у ком се налазе сви доступни канали у односу на средњу фреквенцију(*bandwidth 8-MHz*). Послиједњи параметар представља шта ћемо користити као излаз, што је у овом почетном случају било само емитовање стрима(*-O play*).

Када смо се увјерили да је функција валидна, те да нам неометано и без проблема пушта доступне канале, било је потребно обрадити и модификовати тако да је могуће жељени сигнал обрадити на одговарајући начин, што би у нашем случају био упис у меморију из које је могуће читати тај податак.

- **tsp -I dvb --delivery-system DVB-T2 --bandwidth 8-MHz --frequency 498000000 -O memory**

Промјеном послиједњег параметра са *play* на *memory* омогућавамо упис стрима у меморију који ће касније бити модификован и обрађен за потребе пројекта. Међутим, ова функција непрестано уписује податке како они пристижу у меморију што би значило велико заузеће меморије, те је због тога написан модул *shared_memory_writer* помоћу кога ћемо један по један *TS* пакет уписивати у меморију те ће након тога подаци пристигли у меморију моћи бити прочитани од стране нашег модула за читање података које ћемо обрађивати, а о чему ће више бити ријечи у следећем поглављу.

Унутар модула *shared_memory_writer* претходна функција се покреће кроз сам модул, са тиме што је изабрани излаз меморија система(*-O memory)*. Упис у меморију се врши тако што се прво прими са улаза неколико *TS* пакета, те се затим дијеле на

појединачне пакете, те се тако, један по један, уписују у заједничку меморијску локацију.

- **ts::TSPacket::Copy(shared_memory, &(packets[i]), (size_t)1)**

Ова функција у дијељену меморију (*shared_memory*)уписује један по један садржај *TS* пакета у њу итеративно (*&(packets[i])*), док послиједњи параметар ове функције представља колико меморијског простора нам је потребно за тај податак у бајтима. Пошто знамо да сваки *TS* пакет заузима тачно 188 бајта, због тога смо и узели толико меморијског простора, што је у *TSDuck* помоћној библиотеци представљено са *size_t.*

Послиједње, али не и мање битно, било је иницијализовати дијељену меморију, да би апликација за читање података из меморије знала на којој адреси да приступи подацима. То је урађено са следеће двије функције:

- **shmid = shmget(SHM_KEY, 188, 0666 | IPC_CREAT)**
- **shared_memory = (ts::TSPacket*)shmat(shmid, NULL, 0)**

Функција *shmget* редом узима као параметре адресни простор на ком ће подаци бити доступни(*SHM_KEY*), затим величину меморије која ће нам бити заузета, што је у нашем случају било исто као и величина пакета(188), те на крају опцију како ће се изабрана меморија користити. *IPC_CREAT* значи да се изабрана меморија први пут креира. Као повратну вриједност ова функција прима индентификациони број меморије. Следећа функција заправо заузима меморију на индентификационом броју добијену из претходне функције и говори да ће тим података уписан у меморију бити типа *ts::TSPacket**.

4.3 Обрада примљених података

За обраду примљених података направљен је модул *shared_memory_write.*

Унутар овог модула прво је потребно подесити два параметра. Први параметар представља дужину сегметана на коју хоћемо да наши фајлови буду исјечени док други параметар представља колико сегметана желимо да запамтимо. Након тога, уписујемо заглавље *M3U8* фајла који прима стандардне параметре, међу којима су и параметри које смо поставили на почетку задатка. Затим, прије него што смо почели читати пакете

из меморије неопходно је да прво иницијализујемо дијељену меморију на исти начин на који смо то радили као код уписа меморије са истим параметрима, јер се наравно користи иста адреса.

Тада може почети читање параметара из меморије. Читамо пакет по пакет, уписујемо га у наше сегметне, те провјеравамо да ли се у њему налази *PCR* поље које садржи информацију о томе колико је до тог пакета протекло времена. Када наиђемо на пакет до кога је садржано онолико временског податка колико смо декларисали на почетку, тада се завршава упис у тај сегмент, он се затвара и подаци о том сегменту се уписују у *M3U8* фајл.

Ово се понавља све док се не упише за један сегмент више него што смо поставили други параметар задатка, те се након тога први уписани сегмент брише, како из меморије тако и из *M3U8* фајла те смо тиме добили константан број сегмената који чувамо, односно временску интервал који чувамо је једнак производу почетна два параметра.

У цијелом овом процесу је најважније редовно освјеживати фајлове како они пристижу те их ажурирати у *M3U8* фајлу. Једино *M3U8* фајл прослијеђујемо серверу те његов садржај садржи све информације о локацији и трајању фајлова.

```
#EXTM3U
#EXT-X-VERSION:4
#EXT-X-TARGETDURATION:10
#EXT-X-MEDIA-SEQUENCE:3
#EXTINF:2,0187
http://192.168.234.66:1234/files_ts/segment3.ts
#EXTINF:2,0187
http://192.168.234.66:1234/files_ts/segment4.ts
#EXTINF:2,0187
http://192.168.234.66:1234/files_ts/segment5.ts
#EXTINF:2,0187
http://192.168.234.66:1234/files_ts/segment6.ts
#EXTINF:2,0187
http://192.168.234.66:1234/files_ts/segment7.ts
```

Слика 4.1 –M3U8 фајл након уписа неколико сегмената у њега

На почетку је дато стандардно заглавље, након тога параметар **EXT-X-TARGETDURATION** који нам говори о укупној дужини садржаја М3У8 датотеке, односно производ већ поменута почетна два параметра. Затим слиједи параметар **EXT-**

X-MEDIA-SEQUENCE који представља индекс првог сегмента, који редовно мора бити ажуриран након брисања сегмената. Затим слиједе подаци о сегментима, односно прво њихова дужина параметром **EXTINF** те и локација на којим се фајлови налазе, што је повезано са нашим сервером.

4.4 Синхронизација примљених података са подацима на серверу

У току уписа и читања података у меморију могуће је да се јави грешка ако једна од ове двије функције ради брже од друге, те би се због тога могло десити да податак прочитамо више пута али исто тако би се могло и десити да поједине податке не прочитамо никако, у случају да брже уписујемо податке у меморију него што их читамо. Због тога смо морали одрадити синхронизацију уписаних и прочитаних података.

Синхронизација између уписа и читања података из дијељене меморије одрађена је преко семафора са дијељеном меморијом. Наиме, семафори нам служе да бисмо забранили, односно одобрили приступ неким подацима. Пошто је било неопходно да су семафори доступни у оба програма, овај проблем је такође ријешен дијељеном меморијом. Пошто нам је потребам и семафор за читање података као и семафор за упис података у меморију направљена је структура података.

```
struct semapfore {
  sem_t  sem_writer;
  sem_t  sem_reader;
};
```

Иницијализација ових семафора урађена је функцијом **sem_init(sem_t*, int, unsigned int)** која као параметре прима редом: сам семафор који ћемо иницијализовати, да ли ће семафори бити дијељени између процеса (што није случај код нас јер смо за мапирање података одабрали дијељену меморију), те да ли је при иницијализацији семафор блокиран или не. У нашем случају семафор **sem_writer** смо направили као неблокирајући јер је прво потребно да упишемо неки податак док је семафор **sem_reader** постављен као блокирајући, јер немамо шта да читамо док се неки податак не упише у нашу дијељену меморију.

Двије основне функције за реализацију овог задатка, барем што се тиче семафора, биле су **sem_wait()** и **sem_post()**. Обје функције као параметар примају сам семафор. Разлика између ове двије функције је у томе што **sem_wait()** функција блокира семафор и даље функционисање програма док се тај семафор опет не одблокира, а функција **sem_post()** служи да би одблокирала дати семафор те дозвољава његову радњу. Да би се лакше разумјео начин на који раде ове двије функције узећемо примјер из модула *shared_memory_writer*.

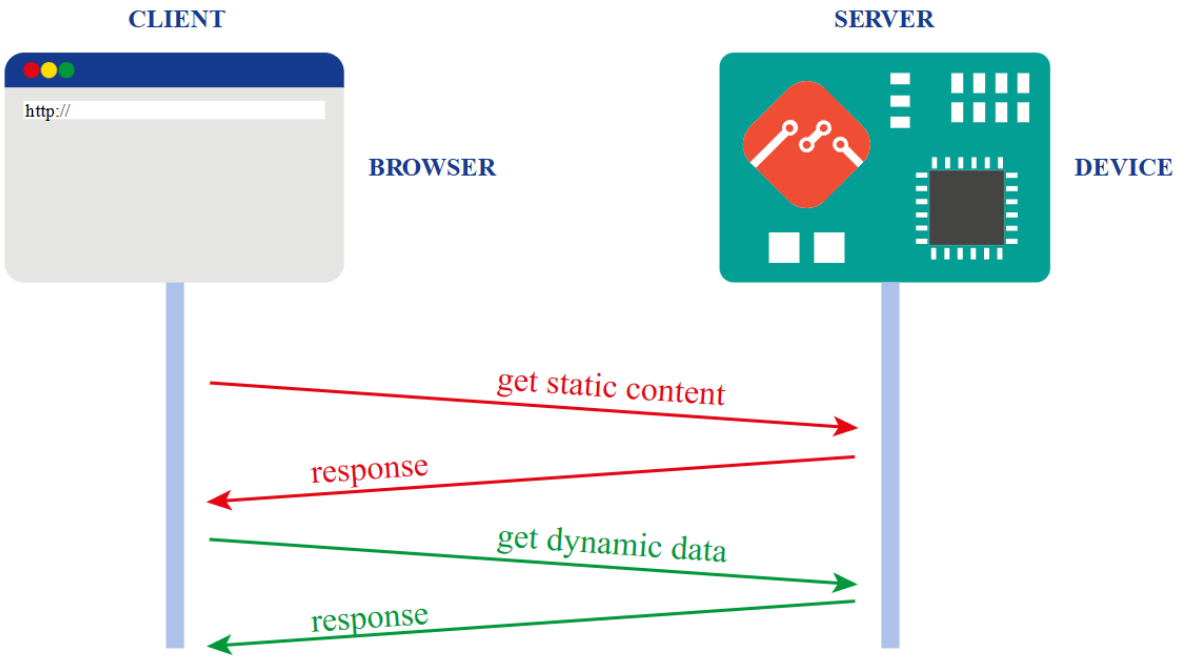
```
sem_wait(&sem->sem_writer);
...
ts::TSPacket::Copy(shared_memory, &(packets[i]), (size_t)1);
...
sem_post(&sem->sem_reader);
```

Као што се види из приложене функционалности прво се чека да се одблокира семафор sem_writer, што је на почетку нашег програма и био случај, закључава га, те се након тога податак уписује у меморију даље се семафор за читање одблокира што значи да се у модулу за читање након откључавања овог семафора може извршити обрада податка, док се прије тога семафор sem_reader закључао, а након обраде податка се откључава семафор за писање што значи да следећу податак може безбједно бити уписан у меморију. Тиме је ријешен проблем синхронизације података.

4.5 Креирање *HTTP* сервера

Као што је већ раније било споменуто, за имплементацију *HTTP* сервера коришћена је помоћна библиотека *mongoose* која је имплементирана у програмском језику *C*. Помоћу ње успијели смо поприлично једноставно направити динамични сервер који ће одговарати упите клијенту/клијентима.

На почетку је потребно подесити параметре као што су адреса на којој ће сервер радити, односно попримати захтјеве клијената као и локална путања на којој се налазе подаци са сервера. Потребно је било направити динамични сервер, што би значило да ће клијент константно, односно динамично приступати *M3U8* фајлу који се налази на серверу те из њега читати податке како се они мјењају.



Слика 4.2 – Принцип рада динамичког сервера

5. Испитивање реализованог система

Испитивање реализованог система је одрађено у три фазе:

- Креирање клијенске апликације за класично пребацивање фајлова путем сервера
- Рад са апликацијом преко *VLC* плејера
- Рад са апликацијом преко *iWedia* плејера

5.1 Креирање клијенске апликације

За почетак је креирана клијенска апликација која ће слати упит на сервер за један *TS* пакет, затим за више *TS* пакета унутар једног фајла те затим и за више *TS* пакета унутар појединачних фајлова. Клијенска апликација само шаље упите за те *TS* пакете који се налазе унутар сервера а за тестирање је направљена скрипта која ради са алатом *PCMCompare* који ради тако што пореди два фајла која су му прослијеђена те уочава разлике између њих те их уписује у засебан фајл. Важно је напоменути да је за ово тестирање коришћен оперативни систем *Windows*.

```
: Generate logs
"..//tools//PCMCompare.exe" InFiles//segment1.ts OutFiles//segment1.ts 2> OutTxt//Difference1.txt

"..//tools//PCMCompare.exe" InFiles//segment1.ts OutFiles//segment1.ts 2> OutTxt//Difference1.txt

"..//tools//PCMCompare.exe" InFiles//segment1.ts OutFiles//segment1.ts 2> OutTxt//Difference1.txt
```

Слика 5.1 – Изглед скрипте за провјеру ваљаности сервера

Резултат овог поређења добили смо у излазним текстуалним датотекама, те након тога што смо установили да је садржај свих датотека исти(***No differences encountered!***) увјерили смо се да је заиста преузети саджај са серверске апликације у потпуности

поклапа са самим подацима на серверу те смо након тога прешли на следећу фазу тестирања.

5.2 Рад са апликацијом преко *VLC* плејера

За рад преко *VLC* плејера било нам је потребно прослиједити податак са сервера који нам говори све о нашем стриму. Тај податак је наравно била *M3U8* датотека која је садржала све информације о валидним сегментима. Она је била прављена на страни обраде података и прослијеђивана серверу као и сви подаци о сегментима. Самим тим, приступајући *M3U8* фајлу на серверу успјели смо пустити канале које смо добијали преко свог *USB* тунера. Пуштени су канали са националном фреквенцијом као што су РТС, ПИНК, ПРВА и РТВ. Овим смо се увјерили

5.3 Рад са апликацијом преко *iWedia* плејера

Пошто је финални корак био пуштање преко *iWedia* плејера, апликацију смо морали прилагодити могућностима плејера. Због тога је направљен још један *M3U8* фајл који указује на већ направљени *M3U8* фајл, те детељније описује податке који се налазе у сегментима.

```
#EXTM3U
#EXT-X-VERSION:4
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1823000,RESOLUTION=1280x720
http://192.168.234.66:1234/files_m3u8/files.m3u8
```

Слика 5.2 – Изглед прилагођеног *M3U8* фајла за *iWedia* плејер

Након овога, морали смо промјенити и податке који ће стизати на *USB* тунер због тога што канали на националној фреквенцији користе аудио кодек који није подржан на овом плејеру. Узет је локално смјештени стрим који смо успјели пустити на једном рачунару. Да би се потпуно доказала ваљаност система пустили смо стрим да непрестано, кроз направљене модуле, ради 24 сата без прекида на више рачунара.

На овај начин, увјерили смо се да се садржај емитује без икаквих проблема и потврдили да је имплементација нашег програма одрађена како треба, те да се његова функционалност не доводи у питање.

6. Закључак

У оквиру овог рада приказано је једно програмско рјешење обраде дигиталног телевизијског сигнала, тачније *MPEG-TS* пакета и његово прилагођавање *HLS* формату дистрибуције. Идеја је била да рјешење буде што генеричније, те да се уз минималне измјене може искористити за стварне индустријске потребе.

Кроз практичну израду рада, задовољени су сви првобитно дефинисани услови, и реализоване све функционалности неопходне за правилно и ваљано понашање коначног система, у који смо интегрисали наш програм.

Тестирањем је доказано да систем ради како треба, те да се његово понашање не мијења у времену, односно да имамо константан квалитет репродуковања аудио – видео садржаја.

Највећа предност овог програмског ријешења је његова једноставност, што нам омогућава да уз употребу минималних ресурса успјевамо задовољити све потребе овог задатка.

7.Литература

[1]……...

[2]……

[3]……

[4]…….

[5]……..