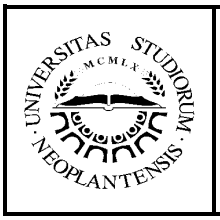


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

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: Ханед Кадрија
Број индекса: РА103/2018

Тема рада: Једна реализација парсера *MPEG* преносног тока у програмском језику *Python*

Ментор рада: Проф. др Илија Башичевић

Нови Сад, мај, 2024.

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**:		**Ханед Кадрија**
Ментор, **МН**:		**Проф. др Илија Башичевић**
Наслов рада, **НР**:		**Једна реализација парсера *MPEG* преносног тока у програмском језику *Python***
Језик публикације, **ЈП**:		Српски / ћирилица
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		**2024.**
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, *MPEG, transport stream, parser***
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		Дигитализација је кроз године увела револуционарне промене корисницима телевизије у свим деловима света. Компресијом сигнала и њиховом бржем и лакшем преносу до крајњих корисника, било то у моментима када је битно на време доставити податке или их пак то урадити квалитетно и без губитка квалитета слике и звука, постигнута је велика промена која је захтевала стандардизацију. У овом раду описан је *MPEG* преносни ток и дат је предлог решења реализације једног парсера *transport stream* пакета, писан у програмском језику *Python.*
Датум прихватања теме, **ДП**:		
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**:		**Haned Kadrija**
Mentor, **MN**:		**Ilija Bašičević, PhD**
Title, **TI**:		One implementation of an MPEG transport stream parser in the Python programming language.
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**:		2024.
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, MPEG, transport stream, parser**
UC		
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, **N**:		
Abstract, **AB**:		Digitization has brought revolutionary changes to television users worldwide over the years. By compressing signals and transmitting them faster and more easily to end users, whether at times when it is crucial to deliver data on time or to do so with quality and without loss of image and sound quality, a significant change has been achieved that required standardization. This paper describes the MPEG transport stream and provides a proposed solution for implementing a transport stream packet parser written in the Python programming language.
Accepted by the Scientific Board on, **ASB**:		
Defended on, **DE**:		
Defended Board, **DB**:	President:	
	Member:	Menthor's sign
	Member, Mentor:	

Захвалност

САДРЖАЈ

Списак слика

Списак табела

Скраћенице

MPEG - Moving Picture Experts Group, Група експерата за покретне слике

PSI – Program-specific information, Програм специфичне информације

PAT – Program Association Table, Табела асоциације програма

PMT – Program Map Table, Табела мапирања програма

PID – Program Identifier, Програмски идентификатор

1. Увод

Телевизија је кроз своју историју доживела значајан развој, од примитивних уређаја са ограниченим приказом слика попут Нипкоњог диска, до савремених дигиталних система који омогућавају висококвалитетан пренос аудиовизуелног садржаја. У почетним фазама величина ТВ екрана је била прилично мала уз слику која трепери, међутим сама помисао на емитовање покретних слика путем радио таласа је била изузетно узбуђујућа. Са развојем технологије телевизија, квалитет и функционалност ТВ уређаја се знатно побољшала. Величина екрана се повећала, а модерни телевизори пружају много боље искуство, захваљујући напретку како харверских тако и софтверских решења.

Са еволуцијом телевизије, дошло је и до потребе за стандардизацијом начина преноса података како би се омогућила широка компатибилност и интероперабилност између различитих уређаја и система. Један од кључних стандарда у области дигиталне телевизије је *MPEG* (енгл. Moving Pictures Experts Group), чије се име везује за радну групу стручњака која развија стандарде за аудио и видео компресију. *MPEG* транспортни ток (енгл. *MPEG Transport stream - TS*) представља један од кључних аспеката овог стандарда, омогућавајући ефикасан пренос аудио и видео садржаја путем различитих мрежа.

Овај дипломски рад истражује једну конкретну реализацију парсера за *MPEG* транспортни ток, а као алат за имплементацију одабран је програмски језик Python. Python се истиче својом флексибилношћу и једноставношћу употребе, што га чини погодним избором за имплементацију оваквих система.

Кроз анализу имплементације парсера у програмском језику Python, рад пружа увид у техничке изазове и решења која се могу применити у пракси. Овај рад доприноси разумевању практичне примене *MPEG* транспортног тока и значају парсера у контексту дигиталне телевизије, истовремено демонстрирајући предности употребе програмског језика Python у овом контексту.

У другом поглављу је детаљно размотрена еволуција телевизије кроз историју, истражујући пионирске дане аналогног емитовања и кључне прелазне тачке ка дигиталним системима, како бисмо стекли основно разумевање дигиталне телевизије.

Након анализе историјског контекста, фокус је пребачен на *MPEG* стандард као кључни оквир за компресију и пренос аудиовизуелног садржаја. Размотрени су основни принципи и карактеристике овог стандарда, истражујући како доприноси ефикасном преносу података у дигиталном телевизијском окружењу.

У наредном поглављу, пажња је посвећена *PAT* табели, *PMT* табели кључним елементима *MPEG* транспортног тока. Анализирана је структура ових табела и њихова улога у организацији и идентификацији аудиовизуелних токова, што је од суштинског значаја за правилно декодирање

Концепт решења детаљно је описан у поглављу посвећеном процедурама анализе и интерпретације *MPEG* транспортног тока. Детаљно је истажено како се тражи *PAT* табела, како се парсира, те како се потом користи за идентификацију *PMT* табела, представљајући кључни корак у процесу декодирања телевизијског сигнала.

У склопу програмског решења, представљена је имплементација парсера за *MPEG* транспортни ток у програмском језику Python. Размотрене су кључне функционалности програма, а посебан нагласак је стављен на флексибилност и једноставност Python-а у овом контексту.

Поглавље посвећено испитивању пружа увид у резултате анализе, користећи ДВБ инспектор као референтни алат. Приказани су излази програма за одабране *MPEG* транспортне токове, наглашавајући кључне информације које се добијају кроз имплементирано програмско решење.

2. Теоријске основе

У овом поглављу ћемо истражити теоријске основе које су кључне за разумевање дигиталне телевизије и MPEG стандарда. Проучићемо историјски контекст дигитализације телевизије. Такође ћемо се упознати са основним појмовима и карактеристикама MPEG стандарда који омогућавају ефикасно кодирање, пренос и декодирање аудио и видео садржаја.. Осим тога, истражићемо и алате и технологије које користимо за развој софтвера за анализу MPEG Transport stream датотека, како бисмо боље разумели њихову улогу и примену у нашем пројекту.

2.1 Историја и еволуција дигиталне телевизије

Почеци телевизије датирају још из средине 20. века када је аналогна технологија била неприкосновена. Првобитни телевизијски системи користили су аналогни сигнал за пренос слика и звука. Овај пионирски период обележен је црно-белом телевизијом, која је уведена током 1940-их и пружила прве слике из света директно у домове гледалаца.

Аналогна телевизија доживела је значајан развој током 20. века, укључујући увођење боје, побољшање квалитета слике и звука, као и проширење броја канала. Међутим, постојала су ограничења, као што су ограничен капацитет преноса, слабија репродукција слике и звука на већим удаљеностима, и склоност ка сметњама.

Након низа година доминације аналогне технологије, почетком деведесетих година 20. века, свет је сведочио преласку ка дигиталном добу у многим сферама, укључујући и телевизију. Ова трансформација означена је развојем првих генерација дигиталне телевизије. Долазак дигиталне телевизије представио је револуцију у начину на који се емитује и прима телевизијски сигнал. Дигитална телевизија користи бинарни систем за пренос података, што омогућава бољи квалитет слике и звука, већи капацитет преноса,

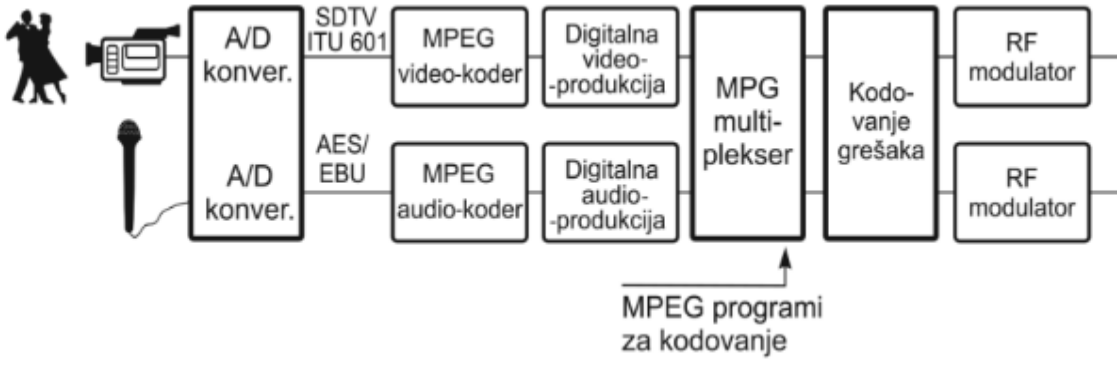
више канала и напредне функције као што су интерактивне услуге и висока дефиниција (енгл. High Definition - HD) [1].

Међународна Телекомуникациона Унија (*ITU*) играла је кључну улогу у постављању стандарда за дигиталну телевизију. Званично је 1982. године усвојен стандард који је дефинисао основне параметре за дигитализацију телевизијског сигнала, представљајући први корак у преласку са аналогног на дигитално [2].

У модерном контексту дигиталне телевизије, користе се како континуални, тј. аналогни електронски сигнали, тако и дискретни, тј. дигитални електронски сигнали, уз употребу одговарајућих уређаја за њихову обраду. Технички параметри у телевизији прилагођавају се потребама људске перцепције боје, сјаја и звука у покретним сликама [1].

2.1.1 Пренос дигиталног сигнала

На слици 2.1 можемо видети комплетан процес настанка и преноса дигиталног сигнала. Видео и аудио снимци са стварних сцена или архивски аналогни снимци програма прво се претварају у дигитални формат користећи одговарајуће аналогно-дигиталне конверзије и у складу са стандардима, као што је *SDTV 601* за видео сигнал и *AES/EBU* за аудио сигнал. Након тога, ови сигнали се компресују коришћењем *MPEG* видео и аудио кодера. Након обраде у одвојеним дигиталним видео и аудио продукцијама, оба сигнала се интегришу у *MPEG* мултиплексеру у јединствени програмски и транспортни ток, познат као stream, односно кодовану дигиталну секвенцу битова [1].



Слика 2.1 Упрошћени приказ дигиталног телевизијског система [1]

2.2 Увод у MPEG

Са порастом интересовања за видео комуникацију, потреба за глобалним стандардима аудио и видео компресије је такође расла. Покретна слика The Moving Picture Experts Group (MPEG), формирана у партнерству између Међународне организације за стандардизацију (ISO) и Међународне електротехничке комисије (IEC), представља кључну алијансу радних група које постављају стандарде за различите аспекте кодирања медија. Њихов обухватни рад обухвата компресију звука, видео материјала, графике и геномске податке, уз дефинисање формата за пренос и датотеке прилагођене разноврсним апликацијама. У оквиру ширег оквира ISO/IEC JTC 1/SC 29 – Кодирање звука, слике, мултимедија и хипермедија информација, *MPEG*, заједно са сродним стандардом непокретне слике - *JPEG*, чини темељ глобалних смерница за ефикасно и стандардизовано кодирање информација [6].

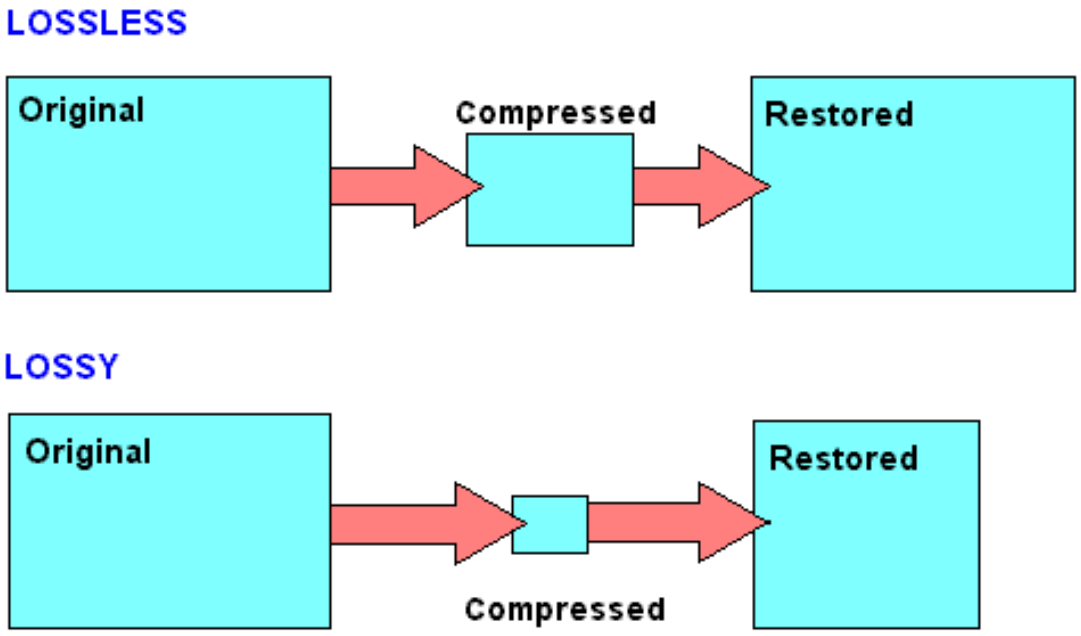
2.2.1 Потреба за компресијом

Обзиром да се самом дигитализацијом видео сигнала добија велика количина бинарних података коју је потребно складиштити или ефикасно и робусно пренети, јавила се потреба за компресијом и смањењем капацитета потребних меморија за меморисање слика, смањење количине и брзине бинарних података које процесори треба да обрађују, као и смањење фреквенцијског опсега телекомуникационих канала којим треба преносити дигиталне ТВ видео и аудио сигнале.

Основна идеја у компресији података произлази из претпоставке да већина информационих извора од практичног интереса није насумична, већ поседује одређену структуру. Кључни аспекти постизања ефикасне компресије леже у препознавању и искоришћавању те структуре. Степену компресије доприноси количина редундантности или структуре која се може идентификовати у подацима. Циљ сваког компримовања јесте да се пошаље променљиви и користан део слике тј. ентропија, а да се редунданса тј.непроменљиви део смањи на минимум. У реалним условима је пожељно пренети и део редундансе како би се, уколико дође до губитака у преносу, информација могла лакше и брже реконструисати.

У зависности од захтева и потреба, можемо одвојити компримовање са губицима и компримовање без губитака. Циљ компримовања без губитака јесте редуковати

величину слике или видеа за складиштење и пренос, али и очувати квалитет, тако да на излазу буде идентичан оригиналу. Насупрот томе, циљ компримовања са губицима јесте постизање задатог бит рате-а за пренос, што често резултира смањеним квалитетом видеа, као и оптимизација слике, поштујући критеријуме оптимизације. Компресија се мора спроводити обазриво, како шум не би постао превише уочљив, што би проузроковало појављивање шума и у свим наредним сликама што доводи до појаве која се зове пропагација грешке [2].



Слика 2.2 Пример компримовања без и са губицима [7]

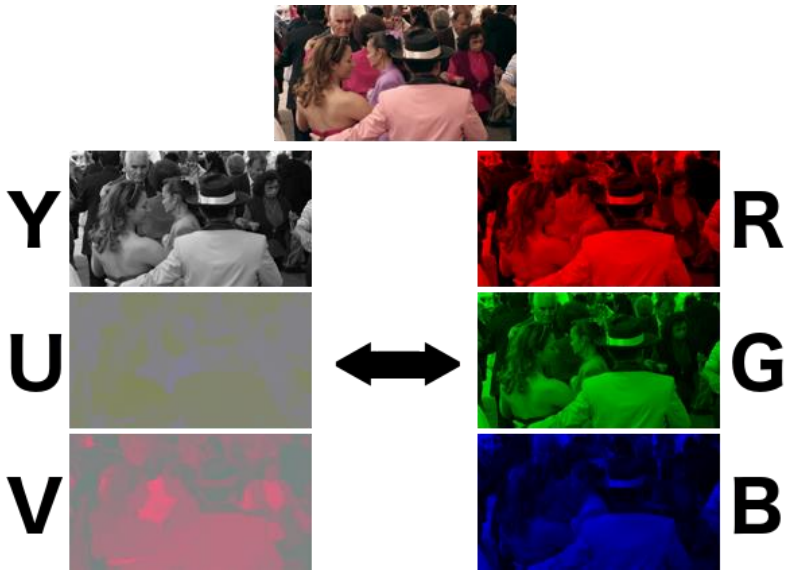
2.2.2 RGB i YUV

RGB и *YUV* су два кључна приступа репрезентацији боје у домену дигиталне обраде слике и видеа. У *RGB* формату, боје се дефинишу интензитетима црвене, зелене и плаве компоненте, док *YUV* формат користи основни канал за светлосну јачину (Y - луминанса) и два додатна канала (*U* и *V* – хроминанса) за репрезентацију боја.

RGB, иако интуитиван за људско око, може заузети значајно више простора, будући да свака боја захтева засебну информацију. Са друге стране, *YUV* формат се често преферира у контексту компресије слике и видеа, захваљујући својој способности да ефикасно задржи квалитет слике уз смањење укупне количине података. Овде, луминансу (Y) користимо за опис светлости слике, док се боје репрезентују кроз хроминантне канале (*U* и *V*) [8].

Повезаност ових формата са *MPEG* стандардима постоји кроз стратегију компресије података. *MPEG* стандарди, као што су *MPEG*-2 и *MPEG*-4, често користе

YUV репрезентацију боје у процесу компресије. Ова одлука је мотивисана чињеницом да људско око има већу осетљивост на промене у светлости (луминанси) него на промене у боји. Кроз коришћење *YUV* формата, *MPEG* стандарди постижу значајне резултате у ефикасности преноса дигиталних ТВ, аудио и видео сигнала путем телекомуникационих канала, чиме се смањује укупна количина података без значајног губитка квалитета .



Слика 2.3 Приказ слике преко YUV I RGB простора боја [8]

2.2.3 MPEG-1

Овај стандард из 1993.године је био први корак ка компресији аудио и видео садржаја за дигитално скаладиштење, омогућавајући кодирање на брзинама до 1,5 Мбит/с, са губицима. *MPEG-1* је данас постао најраспрострањенији формат за компримовање аудио и видео сигнала са губицима, и користи се у великом броју производа и технологија. Најпознатији део овог стандарда је прва верзија *MP3* аудио формата која је представљена.

2.2.4 MPEG-2

Три године након *MPEG-1* појавио се стандард који је био значајно шири у опсегу, подржавајући преплитање слика и високу дефиницију. Иако *MPEG-2* није толико ефикасан као новији стандарди, сматра важним јер је изабран као шема компресије за дигиталну телевизију преко ваздушног сигнала *ATSC*, *DVB* и *ISDB*, дигиталне сателитске ТВ услуге као што је *Dish Network*, дигиталне кабловске ТВ сигнале, *SVCD* и *DVD* Видео. Такође се користи на *Blu-ray* дисковима, али се обично користи *MPEG-4 Part 10* или *SMPTE VC-1* за висококвалитетни садржај. Битни потоци могу варирати од 4 Мбит/с за

телевизију ниске резолуције - *LDTV* (енгл. Low Definition Television), па све до 300 Мбит/с за *HDTV* сигнал.

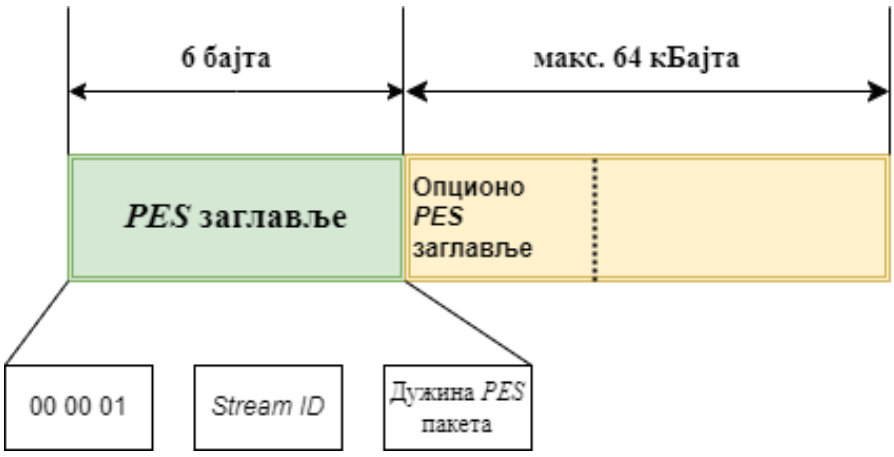
2.2.5 MPEG-4

MPEG-4 унапредио је компресију и подржао је заштиту интелектуалне својине, омогућавајући развој напредних апликација и интерактивних мултимедијалних садржаја. Овај стандард нашао је примену у различитим областима, укључујући стриминг видеа, мобилне апликације и видео конференције. *MPEG-4 Part 10* (*H.264 AVC*), намењена је за пренос ТВ сигнала преко интернета и за мобилну телевизију. Користи се и у *HDTV* преносу, са протоком од 6 Мбита/с [1][3].

2.3 Пакетизација програма и транспортни ток

Низ бита који се генерише у видео и аудио *MPEG* кодерима назива се елементарни ток за видео податке и елементарни ток за аудио податке. Да би се ови подаци пренели на веће удаљености, они се организују у пакете или се пакетизују путем посебног модула, чиме се добија пакетизовани елементарни ток или *PES*. Сваки пакет почиње са фиксним заглављем дужине 6 бајтова, при чему прва 3 бајта представљају "префикс стартног кода", чији је садржај увек 00 00 01 и користи се за идентификацију почетка *PES* пакета. Четврти бајт означава да ли се преноси видео или аудио информација, док пети и шести бајт означавају битну дужину пакета. Ако су оба бајта постављена на нулу, може се очекивати *PES* пакет чија дужина може премашити 64 *kB*. *MPEG* декодер затим мора користити друге аранжмане, попут стартног кода, како би пронашао границе *PES* пакета.

Након ових 6 бајтова *PES* заглавља, преноси се "опционо *PES* заглавље", које представља опционално проширење *PES* заглавља прилагођено тренутним захтевима тренутно преноса елементарног тока. Контролише се помоћу 11 застава у укупно 12 бита у овом опционом *PES* заглављу. Ове заставе показују које компоненте заиста постоје у "опционим пољима" опционог *PES* заглавља, а које не. Укупна дужина *PES* заглавља приказана је у пољу "*PES* дужина заглавља". Опциона поља у опционом заглављу садрже, између осталог, "временске ознаке презентације" (*PTS*) и "временске ознаке декодирања" (*DTS*), које су важне за синхронизацију видео и аудио сигнала. На крају опционог *PES* заглавља могу се налазити и бајтови за пуњење. Након потпуног *PES* заглавља преноси се стварни payload елементарног тока, који обично може бити дуг до 64 *kB* или чак дужи у посебним случајевима, минус опционо заглавље .



Слика 2.4 *PES* пакет

Програмски ток, илити , настаје комбинацијом пакетизованих видео и аудио токова, обогаћен додатним информацијама за телетекст и специфичним програмским подацима за прецизно усклађивање времена програма (енгл. PCR - Program Clock Reference). Такође, садржи неопходне информације за синхронизацију *MPEG* декодера на страни пријема (енгл. SCR – System Clock Reference)

Транспортни ток се формира ради преноса на веће удаљености, као што је случај приликом преноса дигиталних ТВ сигнала до гледалаца. С обзиром на то да се транспортни токови могу преносити брзином до 40 *Mb/s* путем конвенционалних ТВ канала, користећи канално кодирање у складу са *DVB* стандардом, ови токови садрже неколико програмских токова. Да би се максимално искористио дозвољени проток битова, транспортни токови од више пружалаца могу се комбиновати у мултиплексеру за транспортни ток и формирати јединствен транспортни ток на излазу.

2.3.1 Транспортни ток (енгл. Transport Stream)

"Пакетизовани елементарни ток" (*PES*) са својим релативно дугим структурама пакета, није погодан за пренос, а посебно не за емитовање неколико програма у једном сигналу. Са друге стране, у *MPEG*-2, циљ је саставити више независних ТВ или радио програма како би се формирао један заједнички мултиплексирани *MPEG*-2 сигнал. Овај сигнал са подацима се затим преноси путем сателита, кабла или земаљских преносних веза. У ту сврху, дуги *PES* пакети додатно се деле на мање пакете константне дужине. Из *PES* пакета се узимају комади од 184 бајта, а њима се додаје још једно заглавље од 4 бајта, чиме се формирају пакети дужине 188 бајта названи "пакети транспортног тока" који се затим мултиплеџирају. За то, прво се мултиплеџирају пакети транспортног тока једног програма. Програм може садржати један или више видео и аудио сигнала, а екстремни пример је пренос Формуле 1 са неколико углова камере (стаза, гледаоци,

аутомобил, хеликоптер) и презентован на различитим језицима. Сви мултиплеџирани подаци свих програма затим се поново мултиплеџирају и комбинују како би формирали комплетан ток података који се назива "*MPEG*-2 транспортни ток" (*TS*).

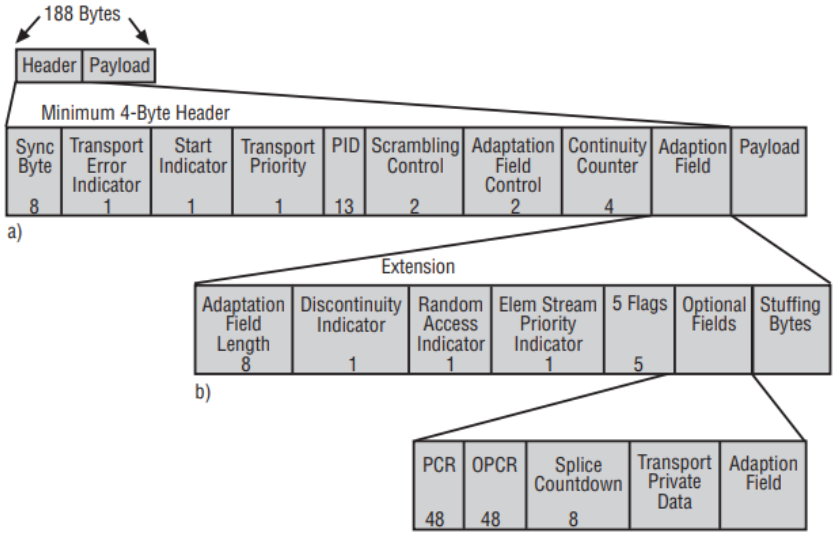
У *MPEG*-1, видео *PES* пакети се једноставно мултиплексирају са аудио *PES* пакетима и складиште на носачу података. Максимална брзина преноса података износи 1,5 *Mbit/s* за видео и аудио, а ток података укључује само видео ток и аудио ток. *MPEG*-1 се користи на видео ЦД-овима. *MPEG*-2 транспортни ток садржи пакете транспортног тока од 188 бајта са програмима и њиховим видео, аудио и осталим сигналима. У зависности од брзина преноса података, пакети једног или другог елементарног тока појављују се већом или мањом фреквенцијом у *MPEG*-2 транспортном току. За сваки програм постоји један *MPEG* кодер који кодира елементарни ток, генерише *PES* структуру, а затим пакетизује ове *PES* пакете у пакете транспортног тока. Брзина преноса података за сваки програм обично износи приближно 2-8 *Mbit/s*, али укупна брзина преноса података за видео, аудио и податке може бити константна или варирати у складу са тренутним садржајем програма. Ово се назива "статистички мултиплекс". Транспортни токови свих програма затим се комбинују у мултиплеџирани *MPEG*-2 податковни ток како би се формирао један укупни транспортни ток који може имати брзину преноса података до отприлике 40 *Mbit/s*.

Често се у једном транспортном току налази 6, 8, 10 или чак 20 програма. Брзине преноса података могу варирати током преноса, али укупна брзина преноса података мора остати константна. Програм може садржавати видео и аудио, само аудио (аудио емитовање) или само податке, па је структура флексибилна и може се мењати током преноса. Да би се могла утврдити тренутна структура транспортног тока током декодирања, транспортни ток такође носи листе које описују структуру, тзв. "табеле".

Слика 2.5 Мултиплексирани аудио и видео *PES* пакети једног програма [4]

Сваки пакет транспортог тока је константе дужине од 188 бајта – минимум 4 бајта заглавље и 184 бајта садржај. На слици 4.3 приказан је пример са свим пољима заглавља, међу којима су најбитнији:

- **Sync byte** – увек има вредност 47hex. Одговоран је за синхронизацију пакета која се на декодеру одиграва након пет примљених транспорт стреам пакета.
- **Transport error indicator** – поставља се на 1 од стране демодулатора како би означио грешку у преносу пакета транспортног тока.
- **Packet Identification(PID)** – описује садржај пакета. 13 бита овог поља приказују разлику између различитих типова пакета.
- **Continuity counter** – четворобитна вредност која се инкрементира од стране енкодера приликом слања сваког новог пакета са истим *PID*-ом и служи као одредница за изгубљене или поновљене пакете [2].

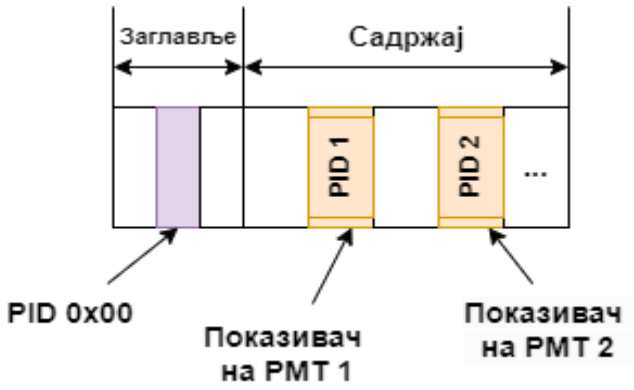


Слика 2.6 Структура TS пакета [2]

2.3.2 *PAT* и *PMT* табеле

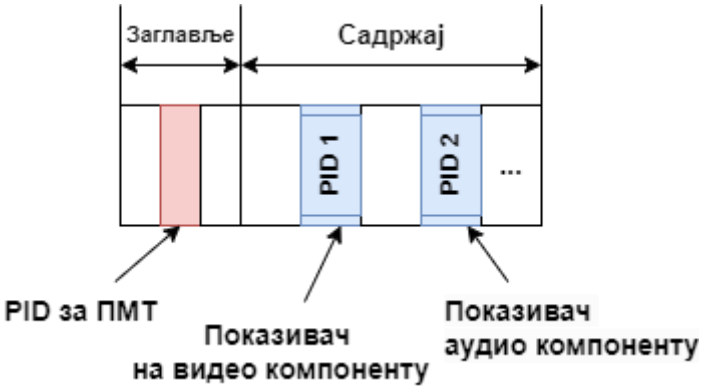
У транспортом току, сви пакети једног елементарног тока имају исти *PID*. Свака аудио, видео или телетекст компонента има свој *PID*, на основу којег демултиплексер на пријемној страни исправно спаја токове. *PSI* (енгл. **Program Specific Information**) представља метаподатке о елементарним компонентама које се синхронизују и комбинују како би настао један програм. *PSI* су табеле које се повремено шаљу у оквиру садржаја пакета.

Прва *PSI* табела коју ћемо поменути јесте *PAT* (енгл. **Program association table)**. Ова табела се појављује периодично и пружа основне информације о свим програмима који се преносе у транспортном току. Сваки унос у *PAT* садржи идентификатор програма(енгл. Program number) и *PID* за одређени сервис, чиме омогућава уређају да пронађе где се налазе остале информације о том програму. Ова табела се шаље са опште познатим *PID*-ом вредности 0x000.



Слика 2.7 Пример *PAT* tabele

Друга табела коју вреди поменути јесте *PMT*. *PMT* табела садржи детаљне информације о специфичном програму, укључујући *PID*-ове свих токова података (аудио, видео, текст итд.) повезаних са тим програмом. *PMT* табела омогућава *MPEG*-2 уређајима да правилно декодирају сваки појединачни програм који се преноси.



Слика 2.8 Пример *PMT* табеле

2.3.3 Анализа *MPEG* преносног тока

Анализа датотека *MPEG* Преносног тока је кључна за разумевање садржаја и структуре мултимедијалних токова, као што су телевизијски преноси, дигитални видео записи итд. Ове датотеке често садрже комплексне структуре података, укључујући информације о програмима, аудио и видео компонентама, као и друге релевантне метаподатке о којима је у претходним поглављима било речи. Проналажење и интерпретација ових информација може бити изазовна, што оправдава потребу за алатом који олакшава анализу *MPEG TS* датотека.

Постоје многи случајеви у којима је анализа *MPEG TS* датотека од виталног значаја, као што су:

- Квалитативна анализа видео записа: Проналажење и издвајање аудио и видео компоненти, као и њихова даља анализа, омогућава оцену квалитета и карактеристика мултимедијалног садржаја.
- Техничка подршка: Разумевање структуре и садржаја *TS* датотека може бити од кључног значаја за дијагностику и решавање проблема приликом репродукције или преноса садржаја.
- Форензичка анализа: Анализа *TS* датотека може бити корисна у форензичким истраживањима како би се утврдило порекло, аутентичност и интегритет дигиталних мултимедијалних садржаја.
- Развој софтвера и система: Развијање нових апликација и система који користе *MPEG TS* датотеке захтева детаљно разумевање њихове структуре и садржаја.

Стога, развој алата који омогућава ефикасну анализу *MPEG TS* датотека може имати широку примену и донети значајне користи у различитим областима, од телевизијске индустрије до рачунарске форензике и развоја софтвера [2] [6].

2.4 Python

Python је популаран програмски језик познат по својој једноставности, читљивости и моћним библиотекама које подржавају различите области као што су обрада података, вештачка интелигенција, веб развој и многе друге. Његова синтакса је елегантна и интуитивна, што олакшава писање чистог и ефикасног кода. Python такође подржава разне парадигме програмирања, укључујући процедурално, објектно-оријентисано и функционално програмирање, што га чини флексибилним језиком прилагодљивим различитим потребама и стиловима програмирања. Међутим, иако је Python једноставан за учење и коришћење, може бити мање ефикасан од неких других језика у извршавању одређених врста задатака, посебно у области брзе израде извршних програма и високо перформантних апликација. Ипак, уз правилно планирање и оптимизацију, Python може бити изузетно моћан алат за развој софтвера.

Коришћењем Python-а, једног од најпопуларнијих и најмоћнијих језика за развој софтвера, омогућено је креирање ефикасног и флексибилног алата за анализу аудио и видео садржаја. Кроз овај пројекат, истражујемо могућности Python-а у манипулацији и анализи *MPEG TS* датотека, пружајући корисницима моћан алат за истраживање и разумевање структуре и садржаја ових датотека [5].

3. Концепт решења

Приликом анализе дигиталних телевизијских сигнала у Транспорт Стреам (*TS*) формату, суочио сам се са изазовом екстракције, декодирања и интерпретације података садржаних у овим датотекама. Како бих ефикасно обрадио ове *TS* датотеке и издвојио корисне информације о њиховој структури и садржају, развио сам програмски алат који аутома*TS*ки анализира и генерише извештаје о њима. У овом поглављу описујем концепт решења који сам применио како бих ефикасно обрадио *MPEG-TS* датотеке и добио детаљан увид у њихов садржај и структуру. Ово решење комбинује теоријско разумевање формата *MPEG-TS* са практичном имплементацијом алгоритама за анализу и обраду ових датотека, чиме омогућава брзу и поуздану анализу дигиталних телевизијских сигнала.

3.1 Кораци у реализацији парсера

- Проналажење *PAT* Табеле : Први корак у анализи *MPEG-TS* датотеке је идентификација *PAT* табеле која налази на *PID*-у 0. Приликом читања *TS* пакета, проверавамо да ли је *PID* једнак 0, што указује на присуство *PAT* пакета. Када пронађемо *PAT* пакет, издвајамо информације о *PMT PID*-овима који се користе за даље мапирање.
- Проналажење *PMT* Табеле : Након што идентификујемо *PMT PID*-ове из *PAT* табеле, следећи корак је проналажење одговарајућих *PMT* табела. То постижемо читајући *TS* пакете и тражећи пакете са одговарајућим *PMT PID*-овима. Када пронађемо *PMT* пакет, анализирамо његов садржај како бисмо издвојили информације о аудио и видео *PID*-овима који су део тог програма. Ови *PID*-ови су кључни за декодирање и репродукцију одговарајућег садржаја.

- Анализа Аудио и Видео Компоненти : Након прикупљања информација о аудио и видео *PID*-овима из *PMT* табела, вршимо анализу *TS* датотеке како бисмо идентификовали присуство и дистрибуцију аудио и видео компоненти. Ово укључује читање сваког *TS* пакета и упоређивање *PID*-ова са базом података *PID*-ова који су мапирани на одговарајуће типове садржаја, као што су *MPEG*-1, *MPEG*-2, H.264, итд. Бројимо колико пута се свака компонента појављује у току и формирамо статистике о дистрибуцији аудио и видео компоненти.
- Генерисање Извештаја : Коначно, на основу прикупљених података, генеришемо детаљан извештај о анализи *MPEG-TS* датотеке. Извештај садржи информације о укупном броју пакета, статистике о присуству и дистрибуцији аудио и видео компоненти, као и остале релевантне информације добијене током процеса анализе. Ови извештаји могу бити од великог значаја за даљу обраду и интерпретацију *TS* датотека, као и за оптимизацију и унапређење система за дистрибуцију дигиталног телевизијског садржаја.
- Графички кориснички интерфејс (енгл. Grapich user interface - GUI) омогућава корисницима управљање програмом путем интерактивних елемената као што су дугмад, поља за унос текста и прозори. Детаљније информације о GUI-у и његовој функционалности биће обрађене у наредном поглављу.

Слика 3.1 Дијаграм тока реализације парсера

4. Програмско решење

У наставку, детаљно ћемо размотрити архитектуру софтвера, главне алгоритме и технологије које су коришћене у развоју овог алата, са циљем пружања дубљег увида у процес анализе *MPEG TS* датотека и могућности Python-а у том контексту.

4.1 Основни програмски модули

Да бисмо схватили целокупну архитектуру софтвера, ажно је напоменути да се њиме поред TS датотека, могу парсирати и *PCAP*(енгл. Packet Capture) датотеке и рачунати ентропије *IP* адреса и портова, како изворишних тако и одредишних. Сходно томе архитектура целог програма се састоји из три главна модула:

- **pcap_parser.py**: Овај модул је одговоран за анализу *PCAP* датотека, који садрже податке о пакетима забележеним током преноса података преко мреже. Кроз овај модул се пролази кроз сваки пакет и врши се издвајање корисних информација као што су *IP* адресе, портови и други метаподаци.
- **mpeg_parser.py**: Овај модул се користи за анализу *MPEG TS* датотека. Основни задатак модула је да идентификује и издвоји различите компоненте у *TS* стреаму, као што су видео, аудио, и други типови података.
- **entropy.py**: Овај модул је одговоран за израчунавање ентропије података из задатих извора као што су *IP* адресе, портови и друге карактеристике пакета. Рачунање ентропије омогућава анализу колико су подаци у тим пакетима случајни или предвидљиви, што може бити корисно за разне анализе у области мрежне сигурности и оптимизације мрежних перформанси.
- **main.py**: Главни модул који управља свим функционалностима. У њему је такође направљена логика *GUI*-а.

Сваки од ових модула има своје специфичне функционалности и алгоритме за обраду одговарајућих врста података, али заједно омогућавају комплексну анализу и обраду података у мрежним окружењима. У наставку ће фокус бити на *MPEG TS* парсеру.

4.2 Главни алгоритам и процес анализе *MPEG-TS* пакета

Главни алгоритам самог програма, је имплементиран у оквиру *mpeg_parser* модула. У ту сврху направљена је класа *MpegParser*. Даље ћу навести све методе које су имплементиране у оквиру класе:

- *def __init__(self, file)* – при иницијализацији класе прослеђује се филе који желимо да парсирамо.
- *def read_ts_file(self)* – метода која се прва позива, чији је задатак отварање и читање података бинарних података који се смештају у листу. Повратна вредност ове функције су три речника и укупан број пакета, који ће представљени у крајњем извештају.
- *def parse_ts_packet(self)* – метода која издваја најбитнија поља хеадера, броји пакете и налази *PAT* и *PMT* табеле.
- *def find_pmt(self)* – метода која налази *PMT* табеле и из њих издваја и чува у речник *PID*-ове аудио и видео компоненти
- *def count_audio_and_video_components(self)* – метода чија је функција да изброји укупан број компоненти и укупан број пакета који носе те компоненте.

Знајући да сваки пакет у току има 188 бајта дужине, уграђеном *Python* функцијом *read*() читамо по 188 бајтова сваког пакета све док не дођемо до краја датотеке (Слика 6.1). Након тога следи парсирање и читање јеног по једног пакета. Библиотека ***bitstring*** омогућава манипулацију бинарним подацима на флексибилан начин у *Python-u*. Једноставна је за коришћење, ефикасна и омогућава читање, писање и манипулацију индивидуалним битовима или групама бита унутар бајтова. *Bitstring* метода коју користимо:

- *def ConstBitStream(byte = ...)* - користи за стварање битског тока који садржи низ битова на основу датог бајта или низа бајтова. Повратна вредност ове методе је објекат бит тока који садржи битове представљене датим бајтом или низом бајтова.

За екстракцију и манипулисање битовима и бајтовима, користимо *bitwise* операције <<, >> , |, &.

```
def parse_ts_packet(self):
    enter_here_once = 0
    for packet_data in self.packet_data_list:
        # Parse the TS packet header
        ts_header = bitstring.ConstBitStream(bytes=packet_data[:4])

        sync_byte = ts_header.read('uint:8')
        transport_error_indicator = ts_header.read('bool')
        payload_start_indicator = ts_header.read('bool')
        transport_priority = ts_header.read('bool')
        pid = ts_header.read('uint:13')
        transport_scrambling_control = ts_header.read('uint:2')
        adaptation_field_control = ts_header.read('uint:2')
        continuity_counter = ts_header.read('uint:4')
```

Слика 4.1 *parse_ts_packet* метода

4.2.1 Откривање *PAT* табеле

Прво парсирамо и читамо хеадер поруке и сва поља која њему припадају, а затим на основу *PID*-а за *PAT* табелу (0x000), ишчитавамо *PAT* табелу. *PAT* табелу је довољно прочитати само једном иако се среће више пута у току.

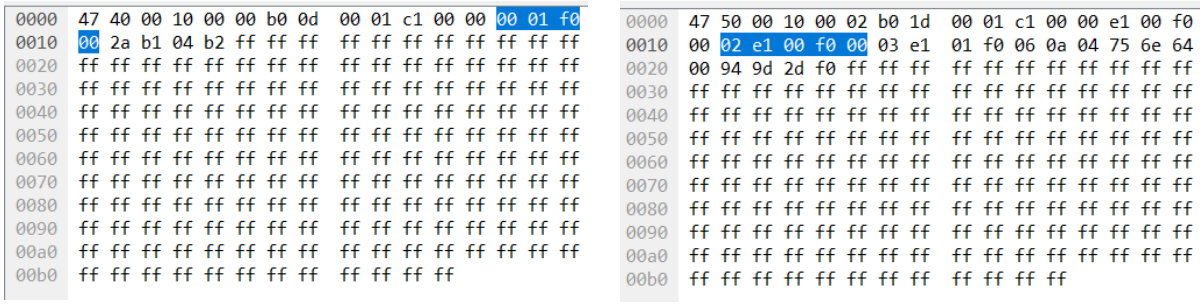
```
if (pid == 0 and enter_here_once == 0):
    # Skip the rest of the header and read the PAT payload
    pat_payload = payload_data

    payload_without_header = ((pat_payload[2] & 0x0F) << 8) | pat_payload[3]
    pmt_tables_len = payload_without_header - 1

    for i in range(11,pmt_tables_len,4):

        # Parse the PAT payload to get the PMT PID
        program_map_PID = int(pat_payload[i] & 0x1F) << 8 | int(pat_payload[i+1])

        self.program_map_PID_list.append(program_map_PID)
        self.pmt_counter += 1

    enter_here_once = 1
```

Слика 4.2 Део кода у ком се врши проналажење *PMT* табела

4.2.2 Откривање *PMT* табеле

Након откривања *PAT* табеле, следећи корак је проналажење одговарајућих *PMT* табела за сваки програм. *PMT* табела садржи информације о компонентама програма као што су аудио, видео, и друге струје података. Да бисмо пронашли *PMT* табелу, морамо прочитати одговарајући *PID* који се налази у *PAT* табели. Сваку табелу налазимо само једном и издвајамо њене компоненте које чувамо у посебном пајтон речнику. У оквиру методе парсе_тс_пацкет(), се на основу паулоада, који је раније ишчитан, налазе *PID*-ови

свих *PMT* табела које се јављају у оквиру тог транспортног тока. На слици 6.2, прва табела приказује једну *PAT* табелу на којој су плавом бојом означени бајтови који се односе на *PMT* табелу. У приказаном случају, постоји само једна *PMT* табела, а све информације везане за њу су смештене у 4 бајта означена плавом бојом. Даље, информација коју желимо да извучемо се налази на 16. и 17. бајту (12. и 13.без хеадер-а), који има вредност 0xf0 00 (0b1111 0000 0000 0000). Прва три бита су резервисана, па добијамо да је *PID* наше *PMT* табеле 0x1000.



Слика 4.3 Заглавље и садржај *PAT* (лево) и *PMT* (десно)

4.2.3 Анализа *PMT* табеле

Када је *PMT* табела идентификована, следи анализа њеног садржаја. Ова анализа укључује читање информација о свакој компоненти програма, укључујући *PID*, тип компоненте (аудио, видео, итд.), и друге метаподатке. Ово омогућава идентификацију и издвајање различитих врста компоненти програма. Уколико има више од једне *PMT* табеле, оне се онда смештају у листу, из које се ваде након што буду обрађене. Обрада се врши у оквиру финд_*PMT*() методе(Слика 6.3). Опет се врши пролазак кроз све пакете, све док се не ишчитају све *PMT* табеле из листе. Прво је издвојен *PID* самог пакета, а затим се проверава да ли тај *PID* припада листи *PMT* табела. Уколико припада, додајемо га у речник како бисмо разликовали различите *PMT* табеле, налазимо дужину самог паулоада и дужину резервисаних поља за информације о програму. Након тога итерирамо кроз остатак паулоада и налазимо тип компоненте(аудио или видео) и њен *PID* и смештамо у пајтон речник.

```
#finding all PMT packets, parsing and reading Stream type from payload. ONLY ONCE for each PMT
if pid in self.program_map_PID_list:
    if pid in self.pid_dict:
        self.pid_dict["PMT "+ str(hex(pid))] = self.pid_dict.pop(pid)

    payload_without_header = ((payload_data[2] & 0x0F) << 8) | payload_data[3]
    len = payload_without_header - 1

    program_info_length = int(payload_data[11] & 0x0F) << 8 | int(payload_data[12])
    i = 12 + program_info_length + 1
    while i < len:
        es_info_length = int(payload_data[i + 3] & 0x0F) << 8 | int(payload_data[i + 4])
        audio_or_video_id = payload_data[i]
        audio_or_video_pid = int(payload_data[i + 1] & 0x1F) << 8 | int(payload_data[i + 2])
        self.a_v_id_pid_dict[audio_or_video_pid] = audio_or_video_id

        self.program_audio_or_video_component_list.append(audio_or_video_pid)
        i += 4 + es_info_length + 1

    self.pmt_counter -= 1
    self.program_map_PID_list.remove(pid)
```

Слика 4.4 Приказ дела кода у ком се врши налажење *PID*-а аудио и видео компоненти

4.2.4 Издвајање података о компонентама

На основу информација добијених из *PMT* табеле, врши се издвајање података о компонентама програма као што су аудио и видео струје.

Метода *count_audio_and_video_components()* садржи неколико корака у циљу откривања и чувања компоненти. Првенствено, потребно је било опет проћи кроз пакете у току како бисмо пронашли *PID*-ове пакета који носе компоненте које тражимо, а затим бројимо колико има тих компоненти и налазимо тачно податке које носе, аудио, видео или нешто друго. Ови подаци се могу даље анализирати или користити за разне друге сврхе, као што су репродукција или обрада садржаја.

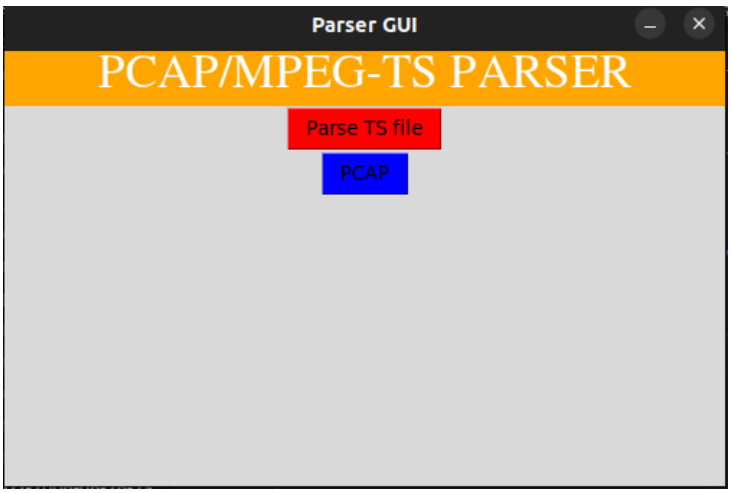
```
#finding audio and video packets based on a pid
if pid in self.a_v_id_pid_dict:
    if self.a_v_id_pid_dict[pid] in self.pid_db_2["video"]:
        self.pid_dict_1["video_component_count"] += 1
        self.pid_db_2["video"].remove(self.a_v_id_pid_dict[pid])
    elif self.a_v_id_pid_dict[pid] in self.pid_db_2["audio"]:
        self.pid_dict_1["audio_component_count"] += 1
        self.pid_db_2["audio"].remove(self.a_v_id_pid_dict[pid])

    self.pid_dict_1["components_number"].append(self.a_v_id_pid_dict[pid])

    if pid_db[self.a_v_id_pid_dict[pid]] not in self.components_dict:
        self.components_dict[pid_db[self.a_v_id_pid_dict[pid]]] = [1,hex(pid)]
    else:
        count = int(self.components_dict[pid_db[self.a_v_id_pid_dict[pid]]][0])
        count += 1
        self.components_dict[pid_db[self.a_v_id_pid_dict[pid]]][0] = count
        if hex(pid) not in self.components_dict[pid_db[self.a_v_id_pid_dict[pid]]]:
            self.components_dict[pid_db[self.a_v_id_pid_dict[pid]]].append(hex(pid))
```

Слика 4.5 Приказ дела кода у ком се врши бројање компоненти и укупан број пакета у којима се оне налазе

4.2.5 Графички кориснички интерфејс

Tkinter(енг. Tk interface) је стандардна библиотека за израду графичких корисничких интерфејса у *Python-u*. Омогућава развој разноврсних апликација са интерактивним корисничким интерфејсима, укључујући прозоре, дугмад, оквире, мените, дијалоге и многе друге компоненте. *Tkinter* је једноставан за коришћење, лако се учи и подржава различите оперативне системе, што га чини популарним избором за развој *GUI* апликација у *Python-u*. Осим основних компоненти, *Tkinter* такође пружа могућности за прилагођавање изгледа и понашања елемената корисничког интерфејса, као и за обраду корисничких акција и догађаја.

GUI апликација омогућава корисницима интерактивну интеракцију са програмом путем графичког корисничког интерфејса. Главни прозор апликације има назив "*PCAP/MPEG-TS PARSER*" истакнут великим насловом. Два главна дугмета у овом *GUI*-у омогућавају корисницима да одаберу жељену врсту анализе: "Parse TS file" за анализу *MPEG-TS* датотека и "*PCAP*" за анализу *PCAP* датотека. Дугмад су стилизована и означена бојама како би корисницима било лакше препознати њихову функционалност: црвена за анализу *MPEG-TS* датотека и плава за анализу *PCAP* датотека(Слика 4.6).



Слика 4.6 Приказ почетне стране апликације

Када корисник кликне на дугме "Parse TS file", отвара се нови прозор у којем корисник може унети путању до *MPEG-TS* датотеке. Након одабира датотеке, корисник може започети процес парсирања притиском на дугме "Start"(Слика 4.7) .

Слика 4.7 Приказ прозора у којем је могуће изабрати датотеку за парсирање

Слично томе, кликом на дугме "*PCAP*" отвара се прозор у којем корисник може одабрати *PCAP* датотеку. Додатно, корисник може конфигурисати параметре анализе, као што су избор извора (Source IP, Destination IP, итд.) и тип ентропије. Након подешавања, корисник може започети анализу притиском на дугме "Start".
Основне функције које су коришћене у сврху прављења *GUI*-а су:

- Tk(): Ова функција креира главни прозор апликације.
- geometry(): Поставља почетну величину прозора.
- title(): Поставља наслов прозора.
- Label(): Користи се за приказивање текста или слике на прозору.
- Entry(): Прави поље за унос текста у које корисник може да унесе податке.
- Button(command=…): Креира дугме које корисник може да притисне да покрене одређену акцију.Parametar *command* дефинише функцију која ће се извршити приликом притиска на дугме.
- pack(): Поставља елемент у прозор.
- grid(): Поставља елементе у мрежу (редове и колоне) у прозору.

5. Испитивање

У овој секцији представљени су резултати парсирања MPEG транспортних токова које је генерисао наш програм за анализу. Ови резултати ће затим бити упоређени и верификовани помоћу *DVB inspectora*, што ће нам омогућити да проверимо тачност и потпуност наше анализе и програмске имплементације у односу на стандарде и спецификације за MPEG транспортне токове.

5.1 *DVB Inspector*

DVB Inspector је отворени *DVB* анализатор, написан у Јави. Може приказати логичку структуру *DVB SI* и *PSI* података. Такође приказује податке о коришћењу битова. *DVB Inspector* може се користити за анализу садржаја; структуру *MPEG* видео записа, телетекста, *DVB* титлова, итд.

5.2 Тестирање кода

Тестирање самог софтвера генерално представља изазован посао. Резултати парсирања су представљени у облику *TXT* датотеке под називом *report.txt*. Да бисмо верификовали тачност овог извештаја (слика 5.1) упоредићемо је са извештајем који добијемо од стране *DVB Inspectora* споменутог раније. Извештај нашег програма садржи:

- Укупан број пакета у току
- Број аудио и видео компоненти
- Тачан назив аудио и видео компоненти
- Све *PID*-ове који се јављају у току
- Укупан број пакета са одређеним *PID*-ом

```
Report for sample_960x400_ocean_with_audio.ts:
Number of packets:48193

components_number: 2
audio_component_count: 1
video_component_count: 1

Components:
ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG-2 higher rate interlaced video): 34703 PID: 0x100
ISO/IEC 11172-3 (MPEG-1 audio): 12630 PID: 0x101

PIDs:
0x11:94
0x100:34703
0x101:12630
PAT :383
PMT 0x1000:383S
```

Слика 5.1 Изгенерисани извештај

5.3 Резултати тестирања

У наставку су дати резултати тестирања тј. упоређивања генерисаног извештаја са излазом из *DVB Inspector*-а. Резултати су представљени за *MPEG-TS* токове пронађене на интернету , као и оне које сам добио од стране ментора.

Назив тестиране датотеке	**Резултат**
sample_640x360.ts	Успешно
sample_960x540.ts	Успешно
sample_960x400_ocean_with_audio.ts	Успешно
ABC.ts	Успешно
channel0.ts	Успешно

Табела 1 Резултати поређења генерисаног извештаја и *DVB Inspector*-а

Поред горе наведених тестова, рађени су такође стрес тестови и тестови издржљивости програма.

6. Закључак

У овом истраживању, развили смо и имплементирали MPEG парсер који нам омогућава ефикасно и тачно анализирање транспортних токова у оквиру MPEG стандарда. Користећи овај парсер, можемо дубље разумети структуру и садржај MPEG транспортних токова, што има значајне примене у областима као што су дигитална телевизија, мултимедијални садржаји и мрежне комуникације. Овај рад пружа допринос у области обраде и анализе видео и аудио садржаја у дигиталном формату, отварајући нове могућности за дубље разумевање и ефикасно управљање медијским садржајима.

Обзиром да је овај пројекат базиран на откривању и представљању само једне могућности реализације парсера, има много места за напредак у различитим областима. Неке од могућности за надоградњу и оптимизацију укључују додавање подршке за додатне *MPEG* стандарде, побољшање алгоритама за парсирање, имплементација додатних функција за анализу и откривање додатних информација које пакети у оквиру једног транспортног тока носе, оптимизација кода,.. Додавање нових функционалности могу значајно повећати корисност и употребљивост парсера у различитим сценаријима и индустријама.

7. Литература

[1] Милошевић, Ивана и Слободан Здравковић (2018), "*Дигитална телевизија*", Висока школа електротехнике и рачунарства струковних студија, електронска књига.

[2] Tektronix, " *A Guide to MPEG Fundamentals and Protocol Analysis*", електронска књига.

[3] T.Sikora, (1997)., "*MPEG video compression*", Institute of Electrical and Electronics Engineers, 1053-5888, електронски магазин

[4] "MPEG". Доступно на: https://sr.wikipedia.org/sr-ec/MPEG/, (Коришћено 17.03.2024.)

[5] "MPEG-2 Transmission". Доступно на: https://web.archive.org/web/20170813122328/http://www.abdn.ac.uk/erg/research/future-net/digital-video/mpeg2-trans.html, (Коришћено 17.03.2024.)

[6] „Python документација". Доступно на: https://www.python.org/doc/, (Коришћено 17.03.2024.)

[7] „Компресија са и без губитака". Доступно на: https://www.pcmag.com/encyclopedia/term/lossless-compression, (Коришћено 15.03.2024.)

[8] „*RGB* и *YUV* простор боја ". Доступно на: https://dexonsystems.com/blog/rgb-yuv-color-spaces, (Коришћено 15.03.2024.)