Univerzitet u Zenici
Politehnički fakultet
Softversko inženjerstvo

Sistemska dokumentacija za klijent - server aplikaciju koja radi na pricipu RTSP

- Multimedijalni sistemi i aplikacije -

Studentice:

Dika Bošnjak

Hajrija Bajrić

Profesor:

Prof. dr. sc.
Nermin Goran

Zenica, decembar 2022. godine

Sadržaj

1. DOKUMENTACIJA ZAHTJEVA

Projekat BosFlix predstavlja klijent-server aplikaciju koja radi na principu RTSP-a. Kreirana je desktop aplikacija u kojoj je omogućeno da klijent u bilo kom trenutku povuče film sa servera te da kontroliše prikaz filma. Omogućeno je pokretanje i zaustavljanje filma kao i izlaz iz programa. U nastavku su navedene osnovne specifikacije projekta:

Naziv proizvoda:	BosFlix
Vizija:	Klijent - server aplikacija koja radi na principu RTSP.
Opis:	Aplikacija radi na principu RTSP. U svakom trenutku je moguće povući film sa servera i kontrolisati na strani klijenta.
Rok isporuke:	05.01.2023.
Metrika:	Mjerenje performasi proizvoda na strani klijenta i na strani servera. Analiza paketa korištenjem analizatora Wireshark.
Slučajevi upotrebe :	Učitavanje filma sa servera. Pokretanje filma. Zaustavljanje prikaza. Ponovno pokretanje. Izlaz iz programa.
Pretpostavke:	Krajnji korisnik ima PC računar na kojem je instaliran jedan od popularnih operativnih sistema Windows, Linux ili MacOS. Klijent pokreće aplikaciju i služi se kreiranim dugmadima za kontrolu prikaza videa.
Arhitektura i komponente:	Arhitektura aplikacije bazirana je na Python programskom jeziku i biblioteci Tkinter za kreiranje desktop aplikacije. Korištena su dva protokola nad kojima radi RTSP: TCP (Transmission Control Protocol) i UDP (User Datagram Protocol).

1.1. SARADNICI

Saradnici na projektu BosFlix kao i njihova zaduženja prikazani su u tabeli 1.

Uloga	Ime
Koordinator podrške za upravljanjem proizvodom	Nermin Goran
Softver inženjer	Dika Bošnjak
Tehnički arhitekt	Dika Bošnjak
Projekt menadžer	Hajrija Bajrić
Sponzor projekta	Nermin Goran
Biznis analitičar	Hajrija Bajrić
Tester softvera	Dika Bošnjak i Hajrija Bajrić

Tabela 1. Saradnici na projektu

1.2. ZAHTJEVI NA STRANI KLIJENTA

Klijent-server aplikacija bazirana na RTSP-u predstavlja desktop aplikaciju za prikaz video-sadržaja (filmova). Klijent mora moći u bilo kom trenutku povući film sa servera, pokrenuti prikaz, gledati video, imati mogućnost zaustavljanja, ponovnog pokretanja i zatvaranja programa.

Na strani klijenta, potrebno je:

- kreirati dugmad za pokretanje akcija: odabir filma, pokretanje videa, zaustavljanje prikaza i zatvaranje programa
- kreirati funkcionalnosti koje će se postići klikom na prethodnu dugmad: postaviVideo, pokreniVideo, zaustaviVideo, zatvoriProzor
- implementirati funkciju posaljiRTSPZahtjev koja će se pozivati korištenjem bilo koje ranije navedene funkcionalnosti
- implementirati funkcije procitajRTSPOdgovor koja će omogućiti procesiranje odgovora dobijenog sa servera
- kreirati funkcije za povezivanje na server i otvaranje RTP porta kako bi se omogućila komunikacija između klijenta i servera

1.3. ZAHTJEVI NA STRANI SERVERA

Na serverskoj strani ove aplikacije potrebno je omogućiti povlačenje odabranog filma na strani klijenta te korištenjem definisanih protokola omogućiti slanje paketa podataka u obliku toka (stream-a) kako bi klijent u bilo kom trenutku mogao da zaustavi isporuku paketa ili je ponovno započne. Kako bi se ovo omogućilo potrebno je:

- implementirati funkciju za procesiranje dobijenog zahtjeva od klijenta
- implementirati funkciju za enkodiranje RTP paketa
- omogućiti paketiziranje podataka koji će se slati u header-u i playload-u odgovora klijentu
- implementirati funkciju za slanje odgovora klijentu pri čemu je potrebno obratiti pažnju na način slanja bita i bajtova

1.4. ZAHTJEVI ZA METRIKU PROJEKTA

Prilikom pokretanja projekta potrebno je pratiti i analizirati poslane pakete podataka. Zadan je alat Wireshark koji je potrebno pokrenuti i pratiti slanje paketa između klijenta i servera. Posebno je potrebno obratiti pažnju na:

- korišteni port (TCP ili UDP)
- veličinu poslanih paketa
- redoslijed isporuke paketa

2. ARHITEKTURA I DIZAJN APLIKACIJE

2.1. ARHITEKTURA APLIKACIJE

Aplikacije je izrađena u Python programskom jeziku, korištenjem Tkinter biblioteke za kreiranje desktop aplikacije dok su funkcionalnosti omogućene korišenjem TCP i UDP protokola nad kojima rade RTSP i RTP protokoli i njihove funkcionalnosti. U nastavku će se nabrojane tehnologije biti detaljnije opisane.

2.1.1. RTSP PROTOKOL

Real-Time Streaming Protocol (RTSP) je mrežni protokol dizajniran za kontrolu streaming medija u zabavnim i komunikacijskim sistemima. Ovaj protokol prenosi multimedijalne podatke u realnom vremenu sa servera na krajnji uređaj. Drugim riječima, RTSP djeluje kao „mrežni daljinski upravljač“ za multimedijalne servere. On ne prenosi samu multimediju, već komunicira sa serverom koji prenosi multimedijalne podatke.

Protokol za streaming je strandardizirana metoda prijenosa videa ili audio-video sadržaja između uređaja putem mreže. Protokol za streaming videa šalje dijelove video ili audio sadržaja s jednog uređaja na drugi. Metoda pretvaranja ovih dijelova u sadržaj koji se može reproducirati na uređaju za reprodukciju naziva se metoda “ponovnog sastavljanja” (*engl. reassembling*). Za uspješan proces, krajnji uređaj mora podržavati protokol koji koristi pošiljatelj. U suprotnom, neće biti moguće pustiti prijenos.

Protokol kombinuje transkodiranje i programiranje za prijenos videa preko mreže ili na internet uz vezu koja je jednostavna za korištenje. Jedan od najčešćih slučajeva upotrebe RTSP protokola su sistemi sigurnosnih kamera, gdje je medij video stream koji može biti sa ili bez zvuka.

Prije svega, potrebno je naznačiti šta RTSP ne radi:

- RTSP ne definiše šeme kompresije za audio i video.

- RTSP ne definiše kako se audio i video inkapsuliraju u pakete za prijenos preko mreže; enkapsulacija za streaming medija može biti osigurana RTP-om ili vlasničkim protokolom. (RTP protokol je opisan u odjeljku 2.1.2) Na primjer, RealMedia G2 server i player koriste RTSP za slanje kontrolnih informacija jedan drugom. Ali sam medijski tok može biti enkapsulirani RTP paketi ili sa nekom vlasničkom RealNetworks šemom.

- RTSP ne ograničava način na koji se prenose mediji; može se prenositi preko UDP ili TCP.

- RTSP ne ograničava način na koji media player baferuje audio/video. Audio/video se može reproducirati čim počne da stiže do klijenta, može se reproducirati nakon odgode od nekoliko sekundi ili se može preuzeti u cijelosti prije reprodukcije.

Dakle, RTSP je protokol koji omogućava media playeru da kontrolše prijenos medijskog toka. Kao što je gore spomenuto, kontrolne radnje uključuju pauziranje odnosno zaustavljanje reprodukcije i pokretanje/ponovno pokretanje reprodukcije.

U konačnici, može se reći da RTSP protokol predstavlja kontrolni protokol koji zapravo upravlja načinom dostavljanja stream-a podataka ali ne izvršava sam dostavljanje podataka, za to se koristi TCP, UDP ili ova dva protokola u paru. RTSP protokol zadano koristi port 554 ali se može koristiti i bilo koji drugi slobodni port na uređaju.

2.1.1.1. NAČIN FUNKCIONISANJA RTSP PROTOKOLA

RTSP funkcioniše slično kao HTTP, opisuje se kao "mrežni daljinski upravljač" za medijske servere. Ovaj protokol je dizajniran za kontrolu video i audio tokova *bez preuzimanja* medijskih datoteka. Kada započne video stream, uređaj koji koristi RTSP šalje zahtjev medijskom serveru koji pokreće proces postavljanja.

Prvi zahtjev mora obavijestiti klijenta o dostupnim opcijama putem naredbe *Options*, kao što su pauza, reprodukcija i snimanje. Nakon toga, korisnik može pokrenuti ili isključiti stream. Pored zahtjeva za opcije, RTSP podržava nekoliko kontrolnih komandi poput reprodukcije, pauze i podešavanja. RTSP koristi TCP za održavanje end-to-end i stabilne veze bez potrebe za lokalnim preusmjeravanjem ili keširanjem.

Neke od glavnih karakteristika ovog protokola su: proširiv, jednostavan za raščlanjivanje, bezbijedan, nezavisan od transporta, nudi razdvajanje funkcionalnosti pokretanja i kontrole konferencije i mnoge druge.

Međutim, protokol ne podržava enkripciju sadržaja ili ponovni prijenos izgubljenih medijskih paketa i ne može se prenositi direktno preko HTTP-a (u pretraživaču). To je zato što je RTSP povezan s namjenskim serverom za streaming i oslanja se na RTP za prijenos stvarnih medija. Ovo veliko ograničenje i njegova nemogućnost skaliranja doveli su do pada upotrebe RTSP-a.

2.1.1.2. TERMINOLOGIJA RTSP PROTOKOLA

U nastavku je navedena terminologija potrebna za razumijevanje rada RTSP protokola:

- **Klijent**: Klijent traži kontinuirane medijske podatke od medijskog servera.

- **Veza**: Virtuelni krug transportnog sloja uspostavljen između dva programa u svrhu komunikacije.

- **Neprekidni mediji**: Neprekidni mediji mogu biti u realnom vremenu (interaktivni), gdje postoji „uska“ vremenska veza između izvora i prijemnika, ili streaming (reprodukcija), gdje je odnos manje strog.

- **Entitet**: Informacija koja se prenosi kao dio zahtjeva ili odgovora. Entitet se sastoji od metapodataka u obliku *polja zaglavlja* entiteta i *sadržaja* u obliku tijela entiteta.

- **Inicijalizacija medija**: Inicijalizacija specifična za tip podataka/kodek.. Bilo koja informacija nezavisna od transporta koju klijent zahtijeva za reprodukciju medijskog toka javlja se u fazi inicijalizacije medija pri postavljanju toka.

- **Parametar medija**: Parametar specifičan za tip medija koji se može promijeniti prije ili tokom reprodukcije toka.

- **Medijski server**: Server koji pruža usluge reprodukcije ili snimanja za jedan ili više medijskih tokova. Različiti medijski tokovi unutar prezentacije mogu poticati sa različitih medijskih servera. Medijski server može biti na istom ili drugom hostu kao i web server sa kojeg se poziva njegova prezentacija.

- **Medijski stream**: Jedna medijska instanca, npr. audio stream ili video stream. Kada se koristi RTP, tok se sastoji od svih RTP i RTCP paketa kreiranih od strane izvora unutar RTP sesije.

- **Poruka**: Osnovna jedinica RTSP komunikacije, koja se sastoji od strukturiranog niza okteta koja se prenosi putem veze ili protokola bez veze.

- **Prezentacija**: Skup od jednog ili više tokova predstavljenih klijentu kao kompletan medijski feed, koristeći opis prezentacije kako je definirano u nastavku. U većini slučajeva u RTSP kontekstu, to podrazumijeva zbirnu kontrolu tih tokova.

- **Opis prezentacije:** Opis prezentacije sadrži informacije o jednom ili više medijskih tokova unutar prezentacije, kao što su skup kodiranja, mrežne adrese i informacije o sadržaju.

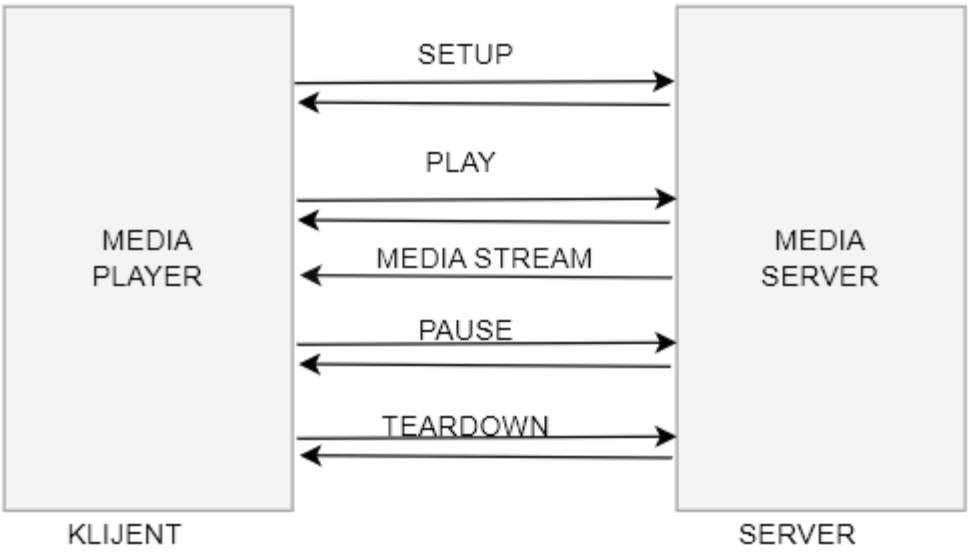
- **RTSP sesija**: Potpuna RTSP "transakcija", na primjer, gledanje filma. Sesija se obično sastoji od toga da klijent postavlja transportni mehanizam za kontinuirani medijski tok (SETUP), započinje stream sa PLAY ili RECORD i zatvara stream sa TEARDOWN.

- **Inicijalizacija transporta**: Dogovaranje informacija o transportu (npr. brojevi portova, transportni protokoli) između klijenta i servera.

2.1.1.3. ARHITEKTURA RTSP PROTOKOLA

Slika 1. prikazuje arhitekturu RTSP protokola uz navedene njegove funkcionalnosti:

- SETUP - postavljanje filma (odabir želenog videozapisa)
- PLAY - pokretanje videozapisa
- MEDIA STREAM - slanje potrebnih paketa podataka zadane veličine
- PAUSE - zaustavljanje videozapisa
- TEARDOWN - izlazak iz programa, obustavljanje RTSP protokola



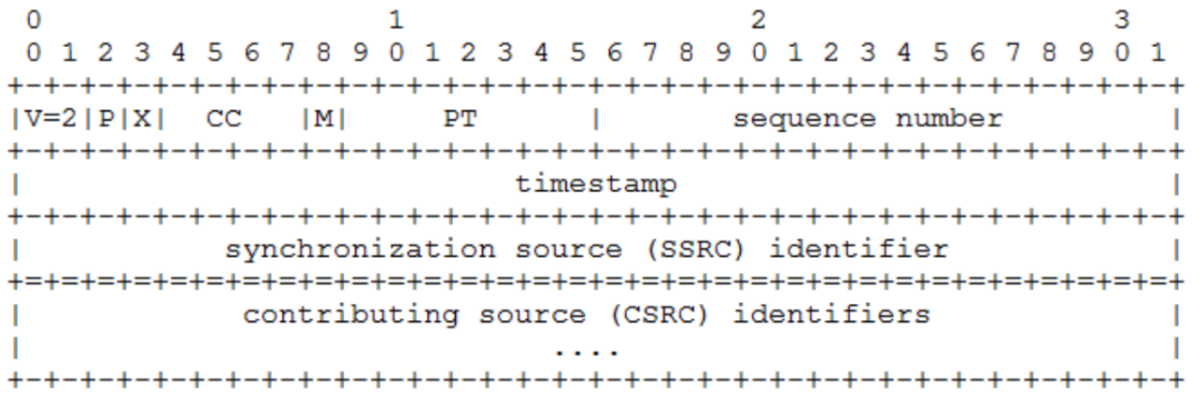
Slika 1. Interakcija između klijenta i servera koristeći RTSP

2.1.2. RTP PROTOKOL

Real Time Transport Protocol (RTP) predstavlja protokol koji radi na osiguravanju funkcija za *mrežno prenošenje podataka,* u stvarnom vremenu, a takvi podaci su primjerice video i audio podaci koji se upotrebljavaju sa svrhom simulacije putem *unicast* i *multicast* mrežnih usluga.

Ovaj protokol određuje veličinu i strukturu paketa kojim se prenose audio i zvučni zapisi, te tako pruža identifikaciju tipa sadržaja, numeraciju sekvenci paketa i vremenske oznake.

Iako dizajniran uz visok stepen neovisnosti od ostalih protokola, RTP se ipak vrlo često upotrebljava u kombinaciji s User Datagram Protocol-om (UDP) kako bi mogao iskoristiti njegove prednosti i potencijale prilikom multipleksiranja i *checksum* provjere. U nastavku je prikazan izgled header-a za RTP paket:



Slika 2. RTP paket header

Nadalje, ovaj protokol *ne sadrži mehanizme dostave paketa*, odnosno, multicasting ni brojeve portova, te mu je iz navedenog razloga prijeko potreban UDP kako bi uspješno funkcionirao. Dakle, *RTP se javlja u ulozi posrednika između UDP-a i aplikacije*.

Nadgledavanje prijenosa RTP paketa osigurava primatelju detekciju gubitaka podataka te nadoknadu nedostataka koji su uzrokovani kašnjenjem. Dakle, vrlo je osjetljiv na kašnjenje paketa i manje je osjetljiv na gubitak paketa. RTP protokol podržava formate kao što su MPEG i MJPEG.

Osnovne osobine RTP protokola su:

- **Timestamp** - svojstvo koje pamti datume i vrijeme, bez informacije o vremenskoj zoni; dozvoljava specifikaciju vremena u milijarditom dijelu sekunde

- **Numeracija sekvenci** - broj sekvenci se u ovom slučaju koristi kako bi se moglo otkriti koji su paketi izgubljeni

- **Miješanje stream-ova** RTP se primarno oslanja na protokole koji se nalaze ispod njega

Za razliku od Real Time Control Protocol (RTCP), RTP pripada transportnom sloju s razlikom što se RTP spaja na UDP. Važno je napomenuti da pri online stream-u važna je brzina samog prijenosa paketa, pri čemu zbog velikih količina paketa nerijetko dolazi do grešaka. Tako da ovaj protokol, nažalost,, nije u potpunosti pouzdan, budući da ne može garantovati isporuku u realnom vremenu.

2.1.3. TCP PROTOKOL

TCP (Transmission Control Protocol) je pouzdan konekciono orijentisan protokol koji dozvoljava da se niz bajtova sa jednog računara isporuči bez greške bilo kom drugom računaru na mreži.

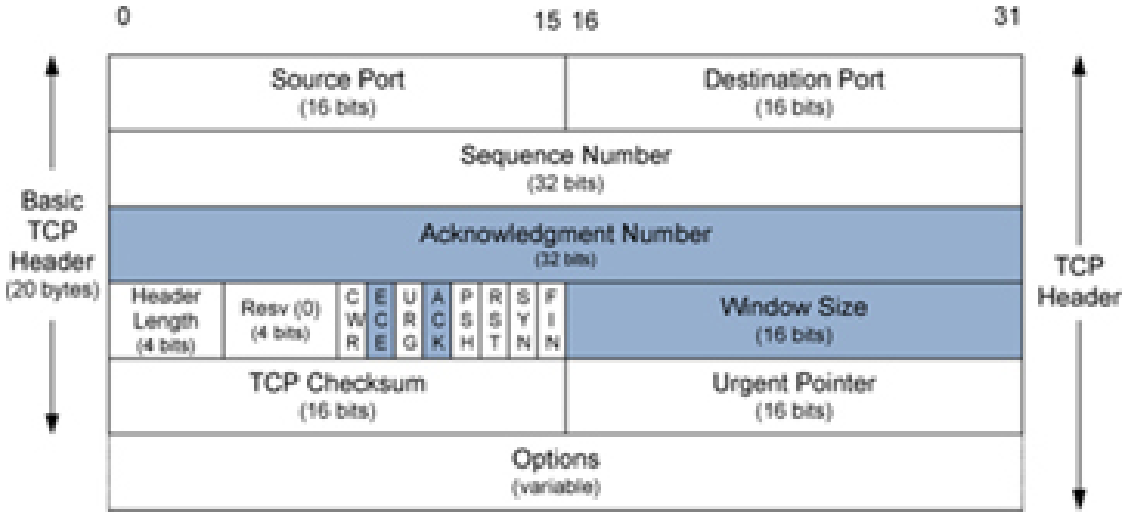
Ovaj protokol se bavi stvarima kao što su:

- Podjela podataka koji su mu proslijeđeni iz sloja aplikacije na dijelove čija veličina odgovara sloju ispod tj mrežnom sloju
- Potvrđivanje prijema paketa
- Postavljanje time out-a kako bi se osiguralo da drugi kraj potvrdi pakete koji su mu poslati

Pošto je sloj transporta zadužen za pouzdan protok podataka, nema potrebe da sloj aplikacije zna za sve ove detalje. Na odredištu prijemni TCP proces ponovo sakuplja (sjedinjuje) primljene poruke u prvobitni niz.

TCP također upravlja protokom da bi osigurao da brzi pošiljalac ne bi zagušio sporiji prijemnik većim brojem poruka od onog broja sa kojim prijemnik može da radi.

TCP protokol prilikom prenosa paketa vodi računa o portu pošiljaoca, portu primaoca, broju poslane sekvence, metapodacima, veličinom paketa, podacima u paketu i još dosta drugih. U nastavku je prikazan izgled zaglavlja TCP protokola:



Slika 2. Zaglavlje TCP protokola

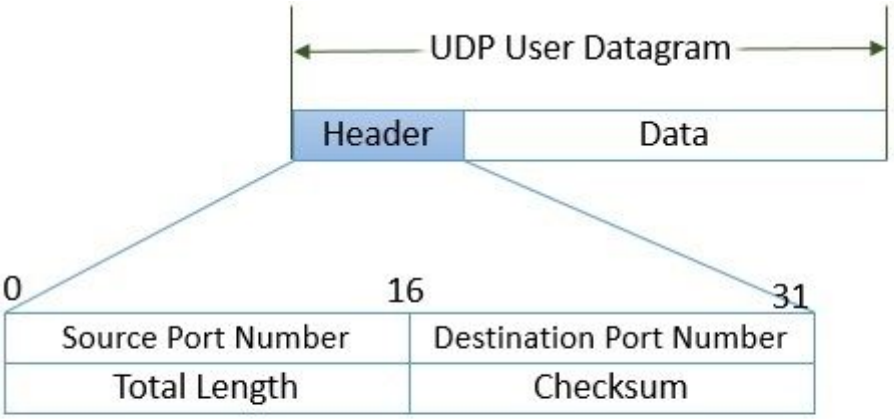
2.1.4. UDP PROTOKOL

UDP (User Datagram Protocol) protokol je protokol koji djeluje na transportnom sloju OSI modela i taj protokol efektivno omogućava aplikacijama direktno korištenje usluga mrežnog

sloja. To je, uz TCP, jedan od temeljnih internet protokola. On se umjesto TCP protokola koristi:

- kod prijenosa podataka aplikacija *koje same osiguravaju pouzdani prijenos* ili kada aplikacija dopušta manje gubitke,
- kod slanja upita jednog računara drugom, *uz mogućnost ponavljanja upita* ako odgovor ne stigne nakon isteka određenog vremenskog intervala,
- *kada je potrebno poslati manji blok podataka*, veličine jednog paketa pa je jednostavnije i brže prenositi samo podatke, bez dodatnih kontrola, a u slučaju pogrešnog prijema podatke poslati ponovno.

U odnosu na mrežnu razinu OSI modela, UDP samo dodaje funkcije multipleksiranja i provjeravanja pogreške prilikom prenošenja podataka, a nema mogućnost provjere prijema poruke jer ne čuva informaciju o stanju veze. Zbog toga se koristi kada je bitnija brzina i efikasnost od pouzdanosti.



Slika 3. UDP zaglavlje i podaci

2.1.5. PYTHON PROGRAMSKI JEZIK

Python je interpretirani, interaktivni, objektno orijentisani programski jezik visokog nivoa, koji poseduje dinamičku sematiku. Odlikuje ga visokokvalitetna struktura podataka, koja ga, u kombinaciji sa dinamičkim pisanjem i vezivanjem, čini veoma atraktivnim za brz razvoj različitih aplikacija.

Njegova fleksibilnost dozvoljava postizanje mnogih rezultata, bilo da su oni mali ili veliki. Python se tako može koristiti za pisanje jednostavnih programa, ali poseduje i snagu potrebnu za kreiranje složenih operacija koje koriste globalne, multinacionalne kompanije. Također, Python je pogodan za upotrebu „scripting" i „glue" programskih jezika, koji se koriste za vezivanje postojećih komponenti u cjelinu.

Pythonova jednostavna sintaksa lako naglašava čitljivost, smanjujući tako troškove održavanja samog programa. Python podržava pakete koji podstiču modularnost programa i

ponovno korištenje koda, dok su „The Python Interpreter" opcija i obimna biblioteka dostupne u izvornom i binarnom obliku.

2.1.5.1. TKINTER

Tkinter je standardna Python programska biblioteka koja služi kao veza s Tk open source programskim rješenjem dostupnim na različitim platformama. Tk programsko rješenje je skup grafičkih elemenata za programsku izradu korisničkog grafičkog sučelja (GUI).

Tk programsko rješenje implementira korisničko grafičko sučelje putem grupe standardnih dodataka, tj. elemenata (widget) grafičkog sučelja koji se spajaju s drugim elementima dok se ne dobije željena cjelina te nekoliko grafičkih organizacija za pozicioniranje standardnih elemenata na samom grafičkom sučelju.

Uz gore navedeno programsko rješenje, postoji nekoliko funkcija kojima se definira izgled osnovnog prozora u kojem se ispisuje grafičko sučelje.

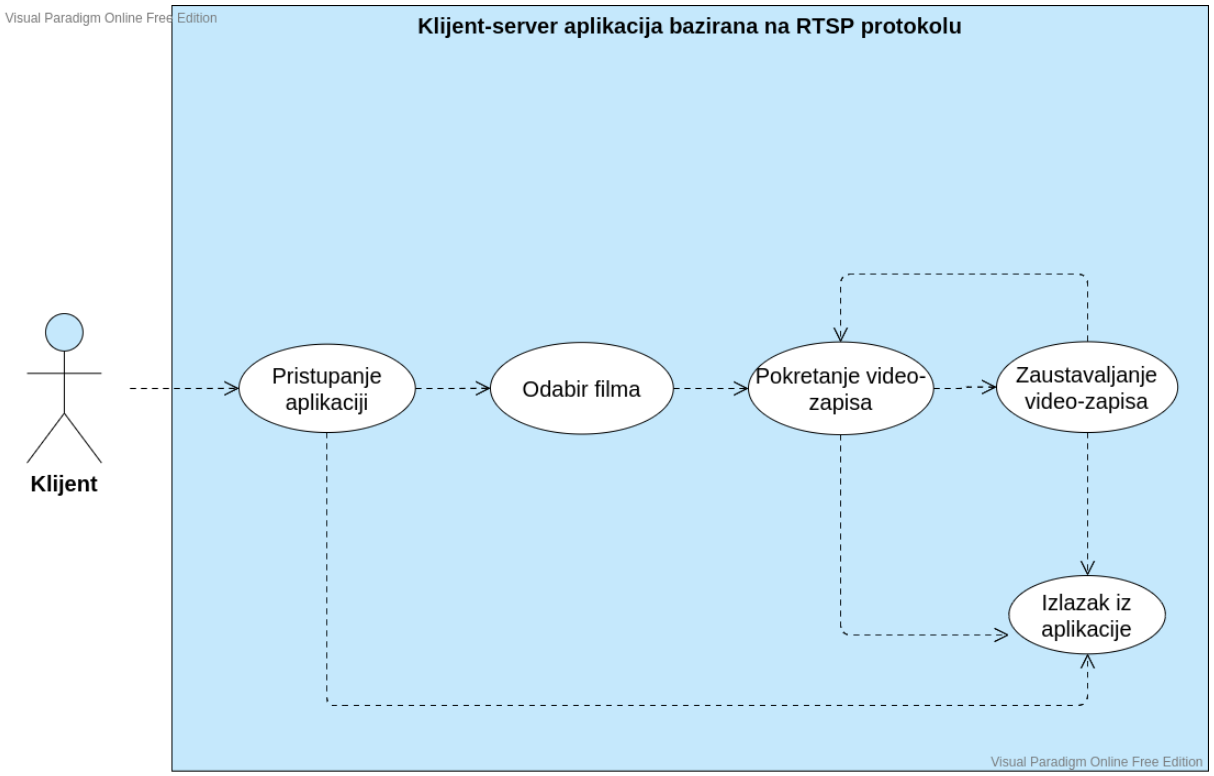
Tkinter biblioteka jednostavna je za korišenje i omogućava kreiranje desktop aplikacije uz sve potrebne komponente koje su zahtijevane u ovom projektu.

2.2. DIZAJN APLIKACIJE

Klijent-server aplikacija za prikaz filmova bazirana na RTSP protokolu, u prvi plan postavlja krajnjeg korisnika te su funkcionalnosti na strani klijenta od velikog značaja. Klijent pristupa aplikaciji, odabire željeni film i pokreće ga. Klijentu se prikazuje film u prozoru desktop aplikacije i omogućavaju funkcionalnosti zaustavljanja reprodukcije filma i ponovnog pokretanja video-zapisa. Klijent u svakom trenutku može izaći iz aplikacije pri čemu se zaustavlja slanje RTSP paketa.

2.2.1. USE CASE DIJAGRAM

Ranije navedene funkcionalnosti najbolje je vizuelno predstaviti kako bi se stekao bolji utisak o potrebnim dijelovima aplikacije. Dijagram slučaja upotrebe (use case diagram) je grafički prikaz mogućih interakcija korisnika sa sistemom. U nastavku je prikazan Use Case dijagram na strani klijenta za ovaj projekat:



Slika 4. Use Case dijagram aplikacije

2.2.2. STATE MACHINE DIJAGRAM

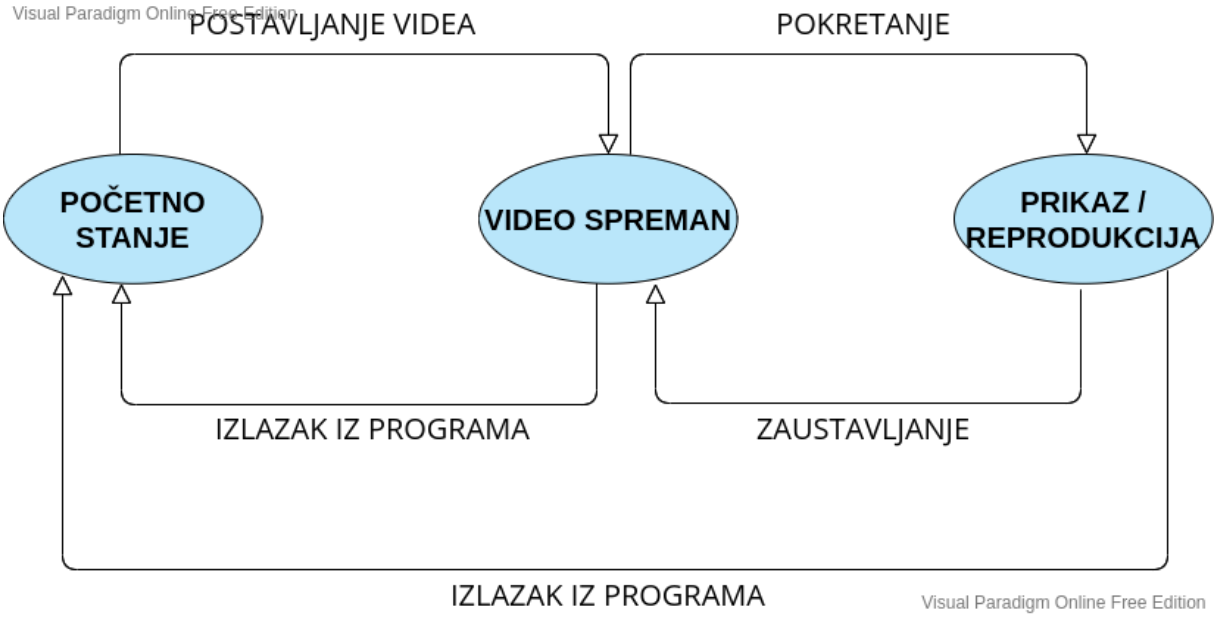
Dijagram stanja je dijagram koji se u softverskom projektovanju koristi za opisivanje sistema uzimajući u obzir sva moguća stanja objekta kada se dogodi neki događaj. Ovo je ponašanje predstavljeno i analizirano u nizu događaja koji se događaju u jednom ili više mogućih stanja. Svaki dijagram predstavlja objekte i prati različita stanja tih objekata u sistemu.

U nastavku je predstavljen dijagram stanja za klijent-server aplikaciju za prikaz video sadržaja na principut RTSP protokola uključujući sljedeća stanja:

- POČETNO STANJE APLIKACIJE
- STANJE SPREMAN VIDEO
- STANJE REPRODUKCIJE VIDEO ZAPISA

U ovom dijagramu predstavljeni su sljedeći događaji:

- POSTAVLJANJE VIDEA (ODABIR)
- PLAY (POKRETANJE VIDEA)
- PAUSE (ZAUSTAVLJANJE VIDEA)
- TEARDOWN (PREKIDANJE PROGRAMA)



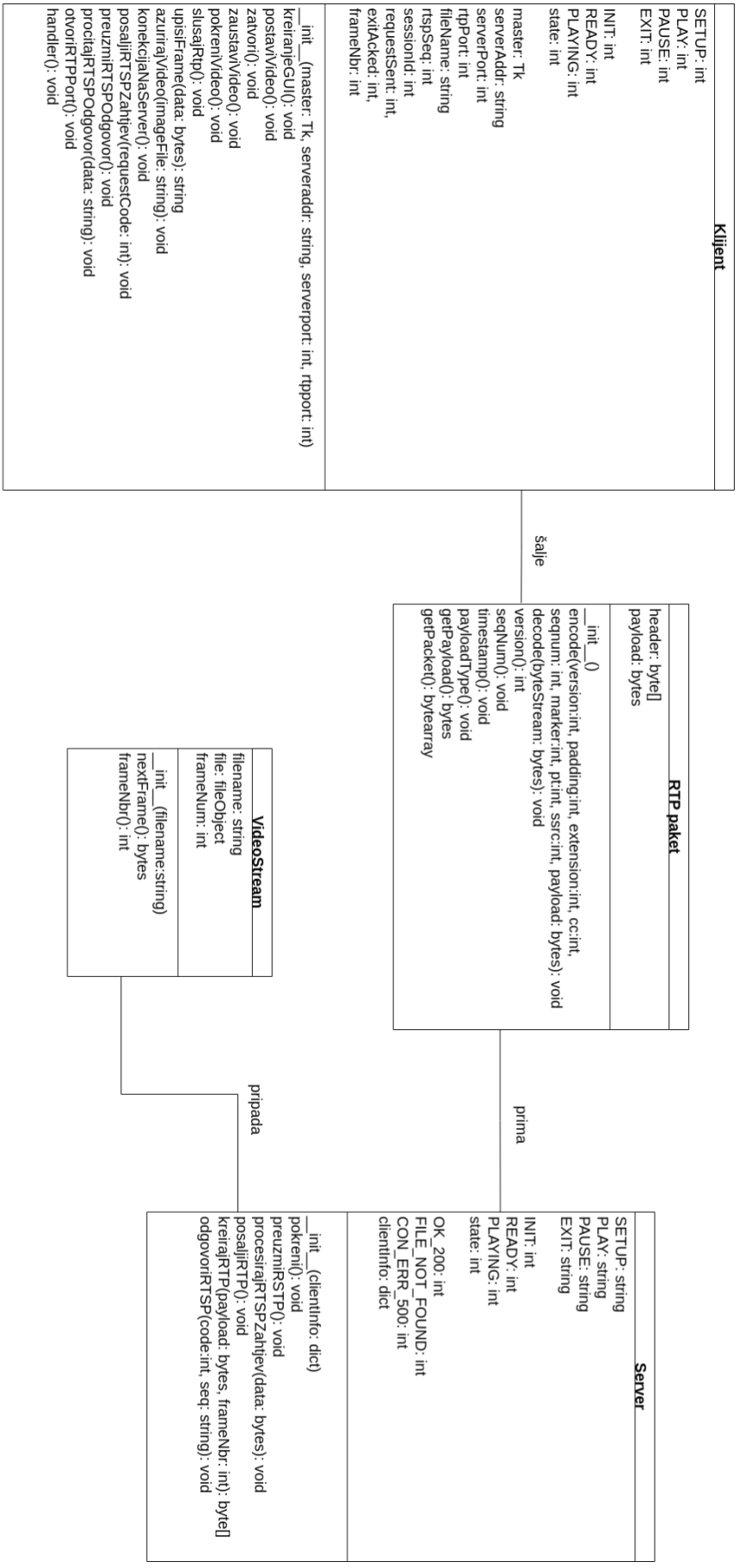
Slika 5. State machine dijagram

2.2.3. DIJAGRAM KLASA

Dijagram klasa (dio UML-a) jest vrsta strukturnog dijagrama u softverskom inženjeringu, koji opisuje strukturu sistema objašnjavajući klase unutar sistema, njihove atribute i odnose. Dijagram klasa pokazuje postojanje klasa i njihovih međusobnih odnosa prilikom logičkog oblikovanja sistema.

U nastavku je prikazan klasni dijagram klijent/server aplikacije za reprodukciju video sadržaja. Predstavljene su klase:

- Server - klasa koja ima potrebna svojstva i obezbjeđuje funkcionalnosti na serverskoj strani aplikacije
- Klijent - klasa koja ima potrebna svojstva i obezbjeđuje funkcionalnosti na serverskoj strani aplikacije
- Video stream - klasa koja upravlja kreiranjem frame-ova podataka iz određenog fajla
- RTP paket - klasa koja omogućava kreiranje RTP paketa koji se šalje između servera i klijenta



Slika 6. Dijagram klasa

3. APLIKACIJA

Na osnovu kreiranih dijagrama: dijagrama slučaja upotrebe, dijagrama stanja i dijagrama klasa kreirana je desktop aplikacija koristeći programski jezik Python. U nastavku će biti predstavljene i opisane glavne funkcionalnosti koje su implementirane.

3.1. IZVORNI KOD

U **ServerStart.py** se nalazi socket.socket() kreira socket objekat. Argumenti proslijeđeni socket() su konstante koje se koriste za specificiranje porodice adresa i tipa socketa. AF_INET je porodica Internet adresa za IPv4. SOCK_STREAM je tip socketa za TCP, protokol koji će se koristiti za transport poruka u mreži.

Vrijednosti proslijeđene .bind() zavise od porodice adresa socketa. U ovom primjeru koristite socket.AF_INET (IPv4). Dakle, očekuje host i port. Host može biti ime hosta, IP adresa ili prazan niz. Ako se koristi IP adresa, host bi trebao biti IPv4 formatiran adresni niz. IP adresa 127.0.0.1 je standardna IPv4 adresa za loopback interfejs, tako da će samo procesi na hostu moći da se povežu sa serverom. Ako se proslijedi prazan string (kao što je u ovom slučaju), server će prihvatiti veze na svim dostupnim IPv4 interfejsima.

Port predstavlja broj TCP porta za prihvatanje konekcija od klijenata. Trebao bi biti cijeli broj od 1 do 65535, jer je 0 rezervisana. Neki sistemi mogu zahtijevati privilegije superkorisnika ako je broj porta manji od 1024.

```
rtspSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
rtspSocket.bind(('', SERVER_PORT))
```

Također, u ServerStart.py se dobivaju informacije o klijentu; adresa i port kroz RTSP/TCP sesiju:

```
clientInfo = {}
clientInfo['rtspSocket'] = rtspSocket.accept()
Server(clientInfo).pokreni()
```

U **Server.py** nalaze se moguće akcije na strani servera:

- Konstruktor __init__ koristi se za inicijalizaciju propertija koji su pohranjeni u objekat ključ-vrijednost parova i imenovan je kao clientInfo

- Funkcija pokreni() - pokreće server na posebnoj niti na osnovu poslanog RTSP zahtjeva

- Funkcija preuzmiRTSPZahtjev() - pokreće se u zasebnoj niti i osluškuje port za dobavljanje novog RTSP zahtjeva te ga nakon toga dekodira u utf-8 format.

- Funkcija procesirajRTPZahtjev() - parsira RTSP zahtjev koristeći funkciju razdvajanja (split()) i ažurira stanje, te vraća odgovor klijentu. Funkcija treba da obradi sljedeće zahtjeve:
    - SETUP - stanje se postavlja na READY i generiše se nova sesija
    - PLAY - stanje se postavlja na PLAYING, UDP port se otvara za strimanje RTP paketa u novoj niti
    - PAUSE - stanje se postavlja na READY i zaustavlja se nit za strimanje
    - EXIT - prekida se nit za strimovanje i zatvara se RTP socket

- Funckija posaljiRtp() - pokreće se u zasebnoj niti i kontinuirano prikazuje video klijentu koristeći pomoćne metode iz VideoStream klase. Nakon toga, validan RTP paket se šalje klijentu.

- Funkcija odgovoriRTSP() - kreira paket sa odgovorom i vraća jedan od 3 moguća status koda: 200 u slučaju ispravnog zahtjeva, 404 za fajl koji nije pronađen i 500 u slučaju errora vezanog za konekciju / probleme na serveru.

U **KlijentStart.py** nalazi se inicijalizacija adresa i portova, kao i naziva fajle:

```
serverAddr = sys.argv[1]
serverPort = sys.argv[2]
rtpPort = sys.argv[3]
fileName = sys.argv[4]
```

Sys.argv je automatski lista stringova koji predstavljaju argumente (odvojene razmacima) na komandnoj liniji. Ime dolazi iz konvencije C programiranja u kojoj argv i argc predstavljaju argumente komandne linije.

Također, u KlijentStart.py kreira se i novi klijent:

```
app = Klijent(root, serverAddr, serverPort, rtpPort)
app.master.title("BOSFLIX-RTPClient")
root.mainloop()
```

U **Klijent.py** nalaze se moguće akcije na strani klijenta:

- Konstruktor __init__ koristi se za inicijalizaciju propertija koji su pohranjeni u objekat ključ-vrijednost parova

- Funkcija kreiranjeGUI() - kreira Tkinter desktop aplikaciju sa zadanim izgledom

- Funkcija postaviVideo() - šalje naziv željenog videa na server

- Funkcija zatvori() - šalje zahtjev EXIT i automatski zatvara aplikaciju

- Funkcija zaustaviVideo() - šalje zahtjev PAUSE

- Funkcija pokreniVideo() - kreira novu nit i šalje novi zahtjev PLAY

- Funkcija slusajRtp() - služi za osluškivanje dolazećih RTP paketa,dekodira ih i prikazuje kao okvire podataka. Obrađuje sve vrste zahtjeva.

- Funkcija upisiFrame() i azurirajVideo() - 2 pomoćne metode koje prikazuju payload u aplikaciji kao sliku na GUI-ju

- Funkcija posaljiRTSPZahtjev() - provjerava stanje i šalje zahtjev na server. Paketi zahtjeva su identični te im se samo razlikuje statusni kod

- Funkcija preuzmiRTSPOdgovor() - preuzima RTSP odgovor na novu nit. Vrši dekodiranje i poziva metodu procitajRTSPOdgovor().

- Funkcija procitajRTSPOdgovor() - parsira poruku koju dobavi sa servera i ažurira stanja na osnovu odgovora:
    - SETUP - ažurira stanje na READY i otvara RTP port
    - PLAY - ažurira stanje na PLAYING
    - PAUSE - ažurira stanje na READY
    - EXIT - ažurira stanje na INIT

- Funkcija otvoriRTPPort() - otvara UDP port i postavlja timeout

- Funkcija handler() - zadužena je za zaustavljanje neočekivanog izlaska iz programa

U RtpPacket.py nalaze se pomoćne funkcije za rad sa RTP paketom:

- Funckija encode() - postavlja header prema zadanom obliku kako treba izgledati u RTP paketu (2 bita za verziju, 1 za padding, 1 za ekstenziju, 4 za contribution source bit, 1 marker bit i 7 bita za payload type. Nakon toga je tu broj sekvence i timestamp i SSRC. Payload se dodaje kao paket.

- Funkcija decode() - razdvaja header od payload-a.

U VideoStream.py nalaze se pomoćne funkcije za čitanje okvira (frame-ova) po 5 bita.

3.2. POKRETANJE APLIKACIJE

Prilikom pokretanja aplikacije, u terminalu je potrebno upisati sljedeću liniju koda kako bi se pokrenuo server:

```
python3 ServerStart.py 1700
```

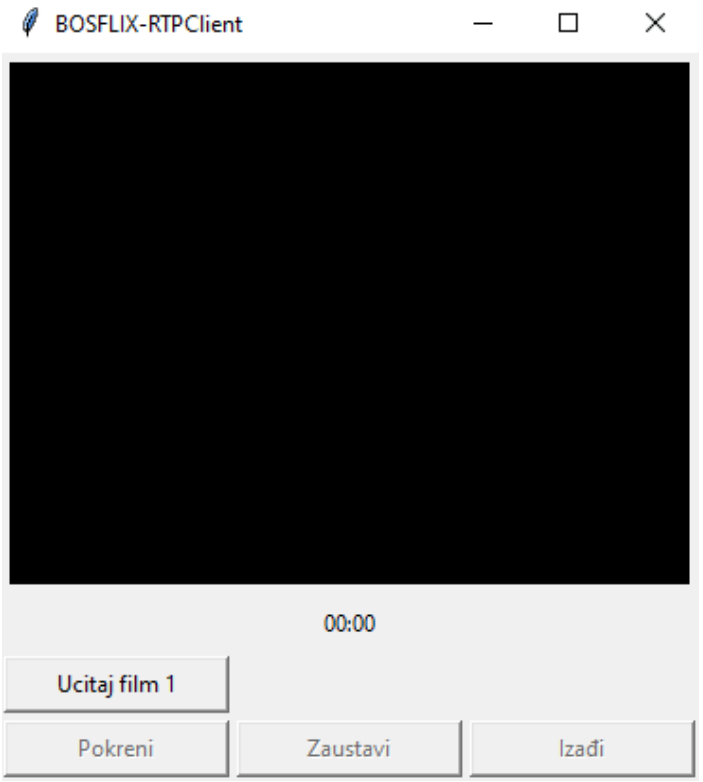
Korisniku se nakon toga prikazuje sljedeća poruka:

```
RTSP ceka na sljedeci zahtjev...
```

Zatim je potrebno upisati sljedeću komandu kako bi se pokrenula klijentska aplikacija:

```
python3 KlijentStart.py localhost 1700 1025
```

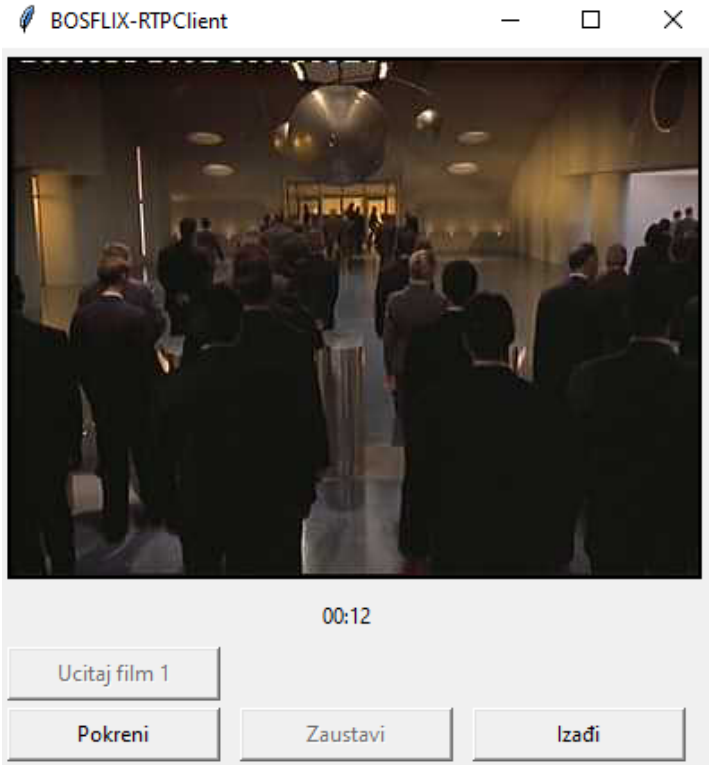
Nakon toga korisniku se prikazuje aplikacija, kao što je prikazano na Slici 7:



Slika 7. Početni prikaz aplikacije BosFlix na strani klijenta

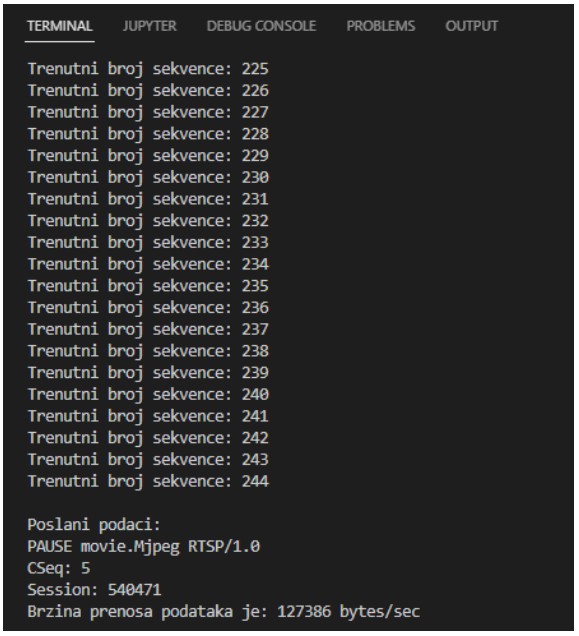
Korisniku su prikazane sljedeće opcije: Ucitaj film 1, Pokreni, Zaustavi i Izađi.

Klikom na opciju Ucitaj film 1, definiše se željeni film te klikom na gumb za pokretanje filma, film se pokreće. Također, korisnik ima opciju za zaustavljanje filma, kao što je prikazano na slici 8:



Slika 8. Zaustavljanje filma od strane korisnika

Sve vrijeme tijekom učitavanja, pokretanja i zaustavljanja filma, u terminalu razvojnog okruženja, ispisuju se podaci tip poslanog podatka, broj sekvenci, sesija i brzina prenosa podataka, prikazano na Slici 9:



Slika 9. Prikaz u terminalu razvojnog okruženja

LITERATURA

1. Overview of Realtime Streaming Protocols, PubNub
2. Real Time Streaming Protocol (RTSP), Brien Posey
3. Real Time Streaming Protocol, H. Schulzrinne, A. Rao, R. Lanphier
4. How to send video using UDP socket in Python, (https://pyshine.com/Send-video-over-UDP-socket-in-Python/)
5. Socket Programming Assignment 6: Video Streaming with RTSP and RTP (https://gaia.cs.umass.edu/kurose_ross/programming/Python_code_only/VideoStreaming_programming_lab_only.pdf)