**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**

**FACULTATEA DE INFORMATICĂ**

****

LUCRARE DE LICENȚĂ

**OpenParty**

**propusă de**

***Iulian Luca***

**Sesiunea:** *iulie, 2021*

**Coordonator științific**

Conf. Dr. Anca Vitcu

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**

**FACULTATEA DE INFORMATICĂ**

OpenParty

*Iulian Luca*

**Sesiunea:** *iulie, 2021*

**Coordonator științific**

*Conf. Dr. Anca Vitcu*

Avizat,

Îndrumător Lucrare de Licență

Conf. Dr. Anca Vitcu

Data \_\_\_\_\_\_\_\_\_\_\_\_ Semnătura \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DECLARAȚIE privind originalitatea conținutului lucrării de licență**

Subsemntatul(a) ………………………………………………………………………………………

domiciliul în …………………………………………………………………………………………………..

născut(ă) la data de ………………..…., identificat prin CNP ………….……………..………………..., absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, Facultatea de ………………………. specializarea …………………………………………………………, promoția …………………………., declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_elaborată sub îndrumarea dl. / d-na \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, pe care urmează să o susțină în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, ………………………… Semnătură student …………………………

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*OpenParty*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de testetc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, *data*

Absolvent *Iulian Luca*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original)

ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil și sursă, să aparțină autorului prezentei lucrări, *Iulian Luca.*

Încheierea acestui acord este necesară din următoarele motive:

*[Se explică de ce este necesar un acord, se descriu originile resurselor utilizate în realizarea*

*produsului-program (personal, tehnologii, fonduri) și aportul adus de fiecare resursă.]*

Iași, *data*

Decan *Prenume Nume* Absolvent *Prenume Nume*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura în original) (semnătura în original)

Cuprins

[Cuprins 6](#_Toc74703856)

[Motivație 7](#_Toc74703857)

[Introducere 8](#_Toc74703858)

[1 Tehnici de video streaming 9](#_Toc74703859)

[1.1 Compresie 9](#_Toc74703860)

[1.1.1 M-JPEG 11](#_Toc74703861)

[1.1.2 MPEG 11](#_Toc74703862)

[1.1.3 H.264 13](#_Toc74703863)

[1.2 Streaming media prin internet 13](#_Toc74703864)

[1.2.1 RTP 14](#_Toc74703865)

[1.2.2 RTCP 15](#_Toc74703866)

[1.2.3 RTSP 16](#_Toc74703867)

[1.2.4 RSVP 17](#_Toc74703868)

[2 Tehnologii utilizate 18](#_Toc74703869)

[3 Implementarea aplicației 20](#_Toc74703870)

[3.1 Modulul principal 21](#_Toc74703871)

[3.1 Modulul de redare si trimitere/primire a segmentelor audio 23](#_Toc74703872)

[3.2 Modulul de redare si trimitere/primire a cadrelor video 24](#_Toc74703873)

[3.2.1 Modulul de extragere a cadrelor video 24](#_Toc74703874)

[3.3 Metoda de sincronizare audio-video 25](#_Toc74703875)

[3.4 Modulul de TCP chatroom 27](#_Toc74703876)

[4 Concluzii 29](#_Toc74703877)

[Bibliografie 30](#_Toc74703878)

Motivație

La finalul secolului XIX frații Lumière au introdus pentru prima data în istorie societatea franceza la un nou mediu de cunoaștere, cinematografia. Fie că imaginile în mișcare deservesc un scop educațional sau de divertisment, acestea au devenit o importanta parte a societății. Încă de la o vârsta frageda, individul este introdus la o serie de imagini care sunt redate atât de rapid încât ochiul uman percepe senzația de mișcare. Dea lungul secolului XX cinematografia, care a început de la bazele unui sistem artistic, a căpătat un caracter informațional odată cu apariția televiziunii. Abilitatea ca orice persoana sa aibă acces la o serie de imagini în mișcare transmise dintr-o locație mai multor consumatori a creat posibilitatea ca mediul video sa fie valabil nu doar în cinematografe dar și în casele oamenilor. Astfel, filmele sau video-urile devin principalele surse de informație sau divertisment.

Odată cu apariția internetului la începutul anilor ’80, redarea imaginilor în mișcare nu a mai fost limitata doar la nivel de televiziune sau cinematografie. Internetul a adus odată cu el posibilitatea de a reda imagini în mișcare în mediul digital. Aceasta posibilitate are definiția de streaming. Astfel, în loc sa fie necesara stocarea completa a unui fișier, consumatorii pot primi în timp real informația, singura limitare fiind viteza conexiunii. Apar noțiuni cum ar fi comprimarea datelor pentru a folosi cat mai puțin trafic, buffering pentru a stoca o parte din datele ce urmează a fi redate, live streaming unde conținutul este înregistrat și trimis în timp real.

In 2005 apare primul site popular de streaming din lume: YouTube. La scurt timp apar alte servicii de streaming dedicate filmelor și serialelor: Netflix, hulu, vimeo.

Teleparty, Disney Party Plus – extensii care permit sincronizarea mai multor participanți pentru același stream de Netflix/Disney Plus (limitat ce oferă librăria serviciilor/fiecare persoana trebuie sa dețină un abonament plătit)

Watch2Gether - site care permite sincronizarea clipurilor de pe youtube/twitch/vimeo intr-o sesiune comuna

Syncplay - extensie pentru playere care permite sincronizarea intre fisierele locale ale participantilor (toti trebuie sa aibe acelasi fisier salvat)

(TODO: De reformulat si adaugat putin despre fiecare și de ce difera aplicatia mea, cum poate fi folosit intr-un cerc de utilizatori)

Introducere

TODO: dupa finalizarea capitolelor, rezumat despre ce urmeaza sa discut în fiecare capitol.

# Tehnici de video streaming

Exista doua modalități de transmitere a informației: descărcarea completa a unui fișier și redarea lui sau trimiterea pe bucăți a informației pentru a fi redat în timp real. Poate fi considerata o a 3-a abordare numita pseudo-streaming. Aceasta se bazează pe redarea informației la câteva secunde după ce se începe descărcarea iar, daca viteza internetului ne permite, utilizatorii pot viziona conținutul în timp ce se face descărcarea completa a fișierului în spate. Streaming-ul este echivalent cu vizionarea conținutului pe televizor. Conținutul este consumat în timp real și este pierdut la final. Pseudo-streaming poate fi înțeles ca vizionarea unui conținut pe televizor și înregistrarea lui pentru a putea fi redat în viitor.

Trei factori au influențat dezvoltarea streaming-ului: algoritmii de compresie pentru audio și video, serverele pentru streaming și îmbunătățiri aduse în rețelele de comunicare.

## Compresie

Video-ul digital reprezintă o serie de imagini numite cadre, fiecare cadru fiind compus dintr-o matrice 2D de pixeli, fiecare pixel fiind reprezentat de valori R, G și B. Astfel, matricea 2D de pixeli care constituie o imagine este de fapt alcătuita din trei matrice bidimensionale, una pentru fiecare dintre componentele RGB. O rezoluție de 8 biți per componentă este de obicei suficientă pentru aplicațiile tipice pentru consumatori. [1]

Compresia se refera la procedeul prin care mărimea fișierului video se reduce considerabil prin modalități de codare și decodare înainte și respectiv după trimiterea lor. Exista doua metode de baza de comprimare: fără pierderi și cu pierderi. în timp ce compresia fără pierdere poate reproduce identic imaginile după decompresie, compresia cu pierderi reda imagini aproximate, dar nu echivalente. Compresia este aplicata folosind unul sau mai mulți algoritmi matematici pentru a elimina din date. Când videoclipul este vizionat, se aplica alți algoritmi de interpretare a datelor pentru a reconstrui imaginile și a le reda. Timpul în care se realizează acești algoritmi se numește latența de compresie. Cu cât algoritmul este mai avansat, cu atât devine mai mare latența de compresie. Atunci când se aplica compresia video și se compara cadrele adiacente în algoritm, se introduce o latența mai mare. în unele aplicări, cum ar fi filmele de studio, aceasta latența este irelevanta, din moment ce video-ul nu este vizionat în direct dar în alte aplicații care necesita transmiterea informației cat mai rapid posibil, latența redusă este esențială.

Compresia video se poate realiza prin exploatarea a patru tipuri diferite de redundanțe:

* Redundanța perceptuala
* Redundanța temporala
* Redundanța spațiala
* Redundanța statistica

Redundața perceptuala

Aceasta se refera la detaliile unei imagini care nu pot fi percepute de ochiul uman. Orice detaliu de acest gen poate fi eliminat fără sa afecteze calitatea imaginii. Sistemul vizual uman afectează modul în care sunt percepute atât detaliile spațiale cât și cele temporale într-o secvență video. [1]

Redundața temporala

Putem exploata proprietatea de persistenta selectând o frecventa mai redusa a cadrelor , dar îndeajuns de mare pentru a asigura percepția de mișcare continua. Din moment ce un video este în esența o secvență de imagini redata la o rata discreta a cadrelor, doua cadre succesive vor arata foarte asemănător. Extinderea similitudinii între două cadre succesive depinde de cât de apropiate sunt eșantioanele (intervalul de cadre) și de mișcarea obiectelor din scenă. Daca luam exemplul unui videoclip cu un prezentator de știri iar rata cadrelor este de 30 de cadre pe secunda, este foarte probabil ca doua cadre succesive sa arate asemănător. Exploatarea redundantei temporale reprezintă majoritatea câștigurilor de compresie în codarea video. [1]

Redundața spațiala

Frecvențele spațiale se referă la modificările nivelurilor dintr-o imagine. Sensibilitatea ochiului scade pe măsură ce frecvențele spațiale cresc. Pe măsură ce frecvențele spațiale cresc, abilitatea ochiului de a face diferența între nivelurile de schimbare scade. Orice detaliu care nu poate fi rezolvat este mediat. Această proprietate a ochiului se numește integrare spațială. Proprietatea ochiului poate fi exploatată pentru a elimina sau a reduce frecvențele mai mari fără a afecta calitatea percepută. Percepția vizuală umană permite astfel exploatarea redundanțelor spațiale, temporale și perceptuale. [1]

Redundața statistica

Coeficienții de transformare, vectorii de mișcare și alte date trebuie codate folosind codurile binare în ultima etapă a compresiei video. Cea mai simplă modalitate de a codifica aceste valori este prin utilizarea unor coduri de lungime fixă; de exemplu, cuvinte de 16 biți. Cu toate acestea, aceste valori nu au o distribuție uniformă și utilizarea codurilor de lungime fixă este risipă. Lungimea medie a codului poate fi redusă prin alocarea cuvintelor de cod mai scurte la valori cu probabilitate mai mare. Codificarea lungimii variabile este utilizată pentru a exploata redundanta statistica și pentru a crește în continuare eficiența compresiei. [1]

Există două ramuri mari în standardizarea compresiei: JPEG și MPEG. în timp ce standardele de baza JPEG se folosesc de comprimarea fiecărui cadru, majoritatea standardelor MPEG se bazează pe principii care reduc redundanțele din cadrele secvențiale.

### M-JPEG

M-JPEG (Motion JPEG) - reprezintă o serie de imagini JPEG. Asigura o compresie de până la 15:1 deoarece fiecare cadru este compresat independent. Din aceasta cauza nu este necesara o putere de procesare foarte mare și nu ocupa multa memorie. Principalul dezavantaj este ca raportul de compresie este foarte mic, din moment ce nu se folosește de tehnici reale de compresie video.

M-JPEG 2000 – are un raport de compresie ușor mai bun comparativ cu M-JPEG, însă este mai complex decât predecesorul sau. Conține același dezavantaj cu M-JPEG din moment ce se folosește tot de compresie pe cadre individuale.

### MPEG

MPEG (Moving Picture Experts Group) – a fost inceput în 1988 pentru a dezvolta standarde de compresie audio și video. Exista câteva standarde importante MPEG, având asemănări și diferențe notabile intre ele. Un lucru pe care toate îl au în comun este faptul ca acestea sunt standarde internaționale stabilite de ISO (Organizația Internaționala de Standardizare) și IEC (Comisia Electrotehnica Internaționala) cu contribuitori din SUA, Europa și Japonia.

* MPEG-1 dezvoltat în 1993 sub standardul ISO/IEC 11172, codarea imaginilor în mișcare și audio asociate pentru medii de stocare digitala cu un bit-rate de 1.5 Mbps. Folosește aceeași tehnica de comprimare ca JPEG, adăugând câteva tehnici de codare eficienta bazata pe secvențele video.

|  |  |
| --- | --- |
| Figura 1‑1 – Secvența de trei cadre JPEG [1] | In compresia M-JPEG fiecare cadru este comprimat și trimis individual (Figura 1‑1) |
| Figura 1‑2 – Secvența de trei cadre MPEG [1] | In compresia MPEG sunt incluse doar informații despre părțile diferite dintre cadre (Figura 1‑2) |

In timpul transmiterii, MPEG limitează consumul lățimii de banda, dar redarea secvențelor din cele 2 figuri va fi aceeași.

MPEG-1 a fost conceput pentru comprimarea video-urilor de calitate VHS fără pierderi de calitate excesive. Formatul asigura o compresie de 52:1

* MPEG-2 dezvoltat în 1995 sub standardul ISO/IEC 13818, acesta o fost utilizat ca standard de baza în televiziunea digitala și DVD, fiind capabil sa realizeze compresia semnalului video de la 280 Mbps la 2-6 Mbps (compresie aproximativa de 200:1). Semnalul audio digital este compresat de la 1.5 Mbps la 100-400 Kbps. Este compatibil cu MPEG-1, însemnând ca poate decodifica secvențe de acest tip. Se poate obține o calitate mai înalta a imaginilor folosind mai multa lățime de banda.

Un standard MPEG-3 a fost propus, destinat televiziunii de definiție înaltă (HDTV). Acesta a fost fuzionat cu standardul MPEG-2 când s-a realizat că standardul MPEG-2 îndeplinea deja cerințele HDTV folosind modificări minime.

* MPEG-4 dezvoltat sub standardul ISO/IEC 14496, caută sa ajute la suportul aplicațiilor care consuma lățime de banda putina, cum ar fi dispozitivele mobile, și în același timp, să ofere suport pentru aplicații care necesita calitate mare și dispun de lățime de banda aproape nelimitata, ca de exemplu filmele de studio. Acesta asigura o rata de compresie mai mare de 200:1.

În general, standardul MPEG-4 este mult mai amplu decât standardele anterioare. De asemenea, permite orice frecventa intre cadre, în timp ce MPEG-2 a fost blocat la 25 de cadre pe secundă în PAL și 30 de cadre pe secundă în NTSC. Atunci când "MPEG-4" este menționat în aplicațiile de supraveghere, este de obicei menționată MPEG-4 partea 2. Acesta este standardul "clasic" de streaming video MPEG-4, cum ar fi MPEG-4 Visual. [2]

Majoritatea diferențelor dintre MPEG-2 și MPEG-4 sunt caracteristici care nu au legătură cu codarea video și, prin urmare, nu au legătură cu aplicațiile de supraveghere. MPEG implică numai codarea completă a cadrelor cheie prin algoritmul JPEG și estimarea modificărilor de mișcare între aceste cadre cheie. Deoarece sunt transmise informații minime între fiecare patru sau cinci cadre, o reducere semnificativă a biților necesare pentru a descrie rezultatele imaginii. În consecință, rapoartele de comprimare de peste 100: 1 sunt comune. Metoda de codare MPEG este foarte complexă și plasează o sarcină computațională foarte mare pentru estimarea mișcării. Decodarea este însă mult mai simplă [1]

* MPEG-7 este despre descrierea obiectelor multimedia și nu are nimic de-a face cu compresia. Acesta oferă o bibliotecă de instrumente de descriere de bază și un limbaj de definire a descrierii (DDL) bazat pe XML pentru extinderea bibliotecii cu obiecte multimedia suplimentare. Culoarea, textura, forma și mișcarea sunt exemple de caracteristici definite de MPEG-7.

### H.264

H.264 este un standard de compresie video folosit pentru înregistrarea, comprimarea și distribuirea conținutului video online. Denumit ca și codecul cel mai utilizat pe scară largă din lume și, de asemenea, cunoscut sub numele de Advanced Video Coding (AVC) sau MPEG-4 Partea 10, H264 a fost dezvoltat în comun de ITU (Uniunea Internațională a Telecomunicațiilor). Scopul lui H.264/AVC a fost sa aducă un nou standard video digital capabil sa ofere o calitate video buna la un bitrate substanțial mai mic decât standardele anterioare, fără a complica prea mult designul, astfel încât punerea în aplicare a acestuia sa rămână practica și relativ ieftina de implementat.

Obiective principale ale standardului: [1]

1) Implementări care oferă o reducere a bitrate-ului de 50%, rezultând o calitate video fixă în comparație cu orice alt standard video.

2) Robustețea erorilor, astfel încât erorile de transmisie în diverse rețele de comunicare să fie tolerate.

3) Capabilități de latență scăzută și o calitate mai bună pentru latență mai mare.

4) Specificație simplă de sintaxă care simplifică implementările.

5) Decodarea exactă a potrivirii, care definește exact modul în care calculele numerice trebuie făcute de un codificator și un decodor pentru a evita acumularea erorilor

Următoarea generație de compresie este cunoscută sub numele de H.265 sau HEVC, oferind un alt salt în eficiență.

## Streaming media prin internet

Datorită existenței sale omniprezente, Internetul a devenit platforma majorității activităților de rețea, inclusiv a aplicațiilor media streaming, unde utilizatorii necesită integrarea serviciilor multimedia. Cu toate acestea, ca rețea de datagrame partajate, Internetul nu este potrivit în mod natural pentru traficul în timp real, cum ar fi streaming media.

În general, următoarele probleme trebuie rezolvate pentru a reda un flux date multimedia prin Internet:

* Hardware-ul de bază trebuie să ofere suficientă o lățime de bandă suficienta pentru a gestiona dimensiunea mare a datelor multimedia
* Aplicațiile trebuie să ia în considerare difuzarea multicast pentru a reduce traficul folosit
* Lățime de bandă garantată. Ar trebui să existe unele mecanisme pentru aplicațiile în timp real pentru a rezerva resurse de-a lungul căii de transmisie pentru a garanta calitatea serviciilor (QoS)
* Aplicatiile trebuie sa se ocupe de problemele de sincronizare, astfel încât datele audio și video sa poată fi redate continuu cu sincronizarea corecta. Acestea trebuie, de asemenea, sa definească operațiuni standard pentru a gestiona livrarea și a afișa datele multimedia. [3]

|  |  |
| --- | --- |
| Figura 1‑3 – Relația dintre protocoale |  |

### RTP

Transportul media în multe aplicații de streaming este implementat în principal cu RTP (Protocolul de transport în timp real, RFC 1889), care este un protocol de transport bazat pe IP pentru conferințe audio și video și alte aplicații multiparticipante în timp real. Este un protocol ușor fără funcționalitate de corectare a erorilor sau de control al fluxului. Astfel, nu garantează nici calitatea serviciilor, nici rezervarea resurselor de-a lungul căii de rețea. RTP este, de asemenea, conceput pentru a lucra împreună cu protocolul de control auxiliar, RTCP, pentru a obține feedback cu privire la calitatea transmiterii datelor și informații despre participanții la sesiunea în curs. RTP este conceput în principal pentru difuzarea multiplă a datelor în timp real, dar poate fi utilizat și în unicast. Acesta poate fi utilizat pentru transportul într-un singur sens, cum ar fi video-on-demand, precum și servicii interactive, cum ar fi telefonia prin Internet. Aplicațiile multimedia necesită o sincronizare adecvată în transmiterea și redarea datelor. Pentru a accepta transmiterea în timp real a datelor, RTP oferă timestamping, numerotarea secvențelor și alte mecanisme. Prin intermediul acestor mecanisme, RTP oferă transport end-to-end pentru date în timp real prin rețeaua de datagrame. [3]

* Timestamping-ul este cea mai importantă informație pentru aplicațiile în timp real. Expeditorul setează marcajul temporal când pachetul a fost trimis. Receptorul utilizează marcajul temporal pentru a reconstrui sincronizarea originală. De asemenea, poate fi utilizat pentru a sincroniza diferite fluxuri cu proprietăți de sincronizare, cum ar fi date audio și video în MPEG. Cu toate acestea, RTP în sine nu este responsabil pentru sincronizare. Acest lucru trebuie făcut la nivel de aplicație.
* Numerele de secvență sunt utilizate pentru a determina ordinea corectă, deoarece protocolul UDP nu livrează pachete în ordine. De asemenea, este utilizat pentru detectarea pierderilor de pachete. Observați că, în unele formate video, atunci când un cadru video este împărțit în mai multe pachete RTP, toate pot avea același marcaj temporal. Prin urmare, timestamping-ul nu este suficient pentru a pune pachetele în ordine.
* Identificatorul payload specifică formatul, precum și schemele de codificare/compresie. Folosind acest identificator, aplicația de primire știe să interpreteze și să redea datele. Exemple de specificații includ PCM, MPEG1/MPEG2 audio și video, și altele. Mai multe tipuri pot fi adăugate prin furnizarea unei specificații de profil și a formatului. Un expeditor RTP poate trimite un singur tip de sarcină utilă, deși tipul sarcinii utile se poate modifica în timpul transmisiei, de exemplu, pentru a se adapta la congestia rețelei.
* Identificarea sursei permite aplicației de primire să știe de unde provin datele. De exemplu, într-o conferință audio, din identificatorul sursă, un utilizator poate spune cine vorbește

RTP este în general folosit împreună cu UDP. Este proiectat în principal pentru difuzare de tip multicast. Din moment ce protocolul TCP este orientat pentru conexiune, acesta nu se scalează bine. Pentru datele în timp real, fiabilitatea nu este la fel de importantă ca livrarea în timp util. Chiar mai mult, transmisia fiabilă furnizată de retransmiterea datelor ca în TCP nu este de dorit. Pachetele RTP și RTCP sunt de obicei transmise utilizând serviciul UDP/IP.

### RTCP

RTCP (RTP Control Protocol) este protocolul de control proiectat să funcționeze împreună cu RTP. Într-o sesiune RTP, participanții trimit periodic pachete RTCP pentru a transmite feedback cu privire la calitatea livrării datelor și informațiile fiecărui membru. Acesta este reprezentat de următoarele tipuri de mesaje: [3]

* RR (raport receptor): Receptorul raportează un feedback de calitate a recepției datelor, cum ar fi cel mai mare număr de pachete primite, numărul de pachete pierdute, decalajul inter-sosire și timestamp-uri pentru a calcula întârzierea dus-întors dintre expeditor și destinatar
* SR (raport expeditor): Expeditorul raportează o secțiune de informații despre expeditor, furnizând informații despre sincronizarea inter-media, contoarele cumulative de pachete și numărul de octeți trimiși
* SDES (elemente de descriere ale sursei): Informații pentru a descrie sursele
* APP (funcții specifice aplicației): Este destinat utilizării experimentale pe măsură ce sunt dezvoltate noi aplicații și noi caracteristici

Folosind mesajele de mai sus, RTCP oferă următoarele servicii pentru a controla transmiterea datelor cu RTP: [3]

* Monitorizarea calității serviciilor și controlul congestiei. RTCP oferă feedback unei aplicații cu privire la calitatea distribuției datelor. Expeditorul își poate ajusta transmisia pe baza feedback-ului raportului receptorului. Receptorii pot ști dacă o congestie este locală, regională sau globală. Managerii de rețea pot evalua performanța rețelei pentru distribuția multicast
* Identificarea sursei. Pachetele RTCP SDES (descrierea sursei) conțin informații textuale, reprezentând identificatori unici la nivel global ai participanților la sesiune. Acestea pot include numele unui utilizator, numărul de telefon, adresa de e-mail și alte informații
* Sincronizare inter-media. Expeditorul RTCP raportează un indicator care specifica timpul real, împreună cu timestamp-ul pachetului RTP. Acesta este folosit pentru sincronizare inter-media
* Pachetele RTCP sunt trimite period către participanți. Pentru a putea fi scalat la grupuri mari de multicast, RTCP trebuie sa împiedice controlul traficului din a suprasolicita resursele rețelei. RTP limitează controlul traficului la cel mult 5% din traficul total al sesiunii. Acesta este impus prin ajustarea ratei în funcție de numărul de participanți

### RTSP

RTSP (Real Time Streaming Protocol, RFC 2326) este un protocol client-server de prezentare multimedia ce permite livrarea controlată a datelor multimedia transmise prin rețeaua IP. RTSP oferă metode de realizare a comenzilor (redare, derulare, pauză, oprire) similare cu funcționalitatea furnizată de CD playere sau VCR-uri. RTSP este un protocol la nivel de aplicație conceput pentru a funcționa cu protocoale de nivel inferior, cum ar fi RTP și RSVP, pentru a oferi un serviciu complet de streaming prin Internet. Poate acționa ca o telecomandă de rețea pentru servere multimedia și poate rula prin TCP sau UDP. RTSP poate controla fie un singur stream, sau mai multe stream-uri sincronizate. RTSP oferă următoarele operațiuni: [3]

* Cerere media de la server. Clientul poate solicita detalii ale prezentării și cere server-ului sa seteze o sesiune separata pentru a trimite datele
* Invitația unui server media la o conferință. Serverul media poate fi invitat la conferință pentru a reda fișiere media sau pentru a înregistra o prezentare
* Adăugarea de fișiere media la o prezentare existentă. Serverul și clientul se pot notifica reciproc cu privire la orice suport suplimentar care devine disponibil

În RTSP, fiecare prezentare și stream media este identificat printr-un URL de tip RTSP. Prezentarea generală și proprietățile suportului sunt definite într-un fișier de descriere a prezentării, care poate include codificarea, limba, URL-urile RTSP, adresa de destinație, portul și alți parametri. Fișierul de descriere a prezentării poate fi obținut de client utilizând HTTP, e-mail sau alte mijloace. RTSP își propune să ofere aceleași servicii pentru audio și video transmise la fel ca și HTTP pentru text și grafică. Dar RTSP diferă de HTTP prin mai multe aspecte. În primul rând, în timp ce HTTP este un protocol stateless, un server RTSP trebuie să mențină "stări de sesiune" pentru a corela solicitările RTSP cu un stream. În al doilea rând, HTTP este practic un protocol asimetric în cazul în care clientul emite cereri și serverul răspunde, dar în RTSP atât serverul media, cât și clientul pot emite solicitări. De exemplu, serverul poate emite o solicitare pentru a seta parametrii de redare ai unui stream. [3]

### RSVP

RSVP (Resource reSerVation Protocol) este protocolul de control al rețelei care permite receptorului de date să solicite o calitate specială end-to-end a serviciului pentru fluxurile sale de date. Astfel, aplicațiile în timp real pot utiliza RSVP pentru a rezerva resursele necesare la routere de-a lungul căilor de transmisie, astfel încât lățimea de bandă solicitată să poată fi disponibilă atunci când transmisia are loc efectiv. RSVP este o componentă principală a viitorului IntServ[[1]](#footnote-1), care poate oferi atât servicii cu cel mai bun efort, cât și servicii în timp real.

RSVP este utilizat pentru a seta parametrii de rezervare a resurselor retelei. Atunci când o aplicație gazdă (receptorul stream-ului de date) solicită o anumită calitate a serviciului (QoS) pentru stream-ul său de date, se utilizează RSVP pentru a livra solicitarea sa routerelor de-a lungul căilor fluxului de date. RSVP este responsabil pentru negocierea parametrilor de conectare cu aceste routere. Fiecare nod capabil de rezervarea resurselor are mai multe proceduri locale pentru configurarea și executarea rezervării. Controlul politicii determină dacă utilizatorul are permisiunea administrativă de a face rezervarea (autentificare, control acces și contabilitate pentru rezervare). Controlul admiterii ține evidența resurselor sistemului și determină dacă nodul are suficiente resurse pentru a furniza QoS-ul solicitat. [3]

# Tehnologii utilizate

Dezvoltarea aplicației s-a realizat folosind limbajul Python 3.6.0 împreună cu diverse librarii ce au ajutat la implementarea diverselor funcționalități necesare.

|  |  |
| --- | --- |
| Quick PyQt5 : 1] Signal and Slot Example in PyQt5 | by Manash Kumar Mandal  | Manash&#39;s blog | PyQt5 este o legătură către framework-ul Qt v5, un set de librarii C++ și instrumente de dezvoltare care includ abstractizări independente de platformă pentru interfețe grafice cu utilizatorul (GUI), precum și rețele, fire de executie, expresii regulate, baze de date SQL, SVG, OpenGL, XML și multe alte caracteristici puternice. Dintre acestea au fost selectate funcționalitățile pentru generarea interfeței grafice (împreună cu tool-ul Qt Designer - Figura 2‑1), stabilirea legăturii dintre elementele interfeței și codul rulat în spate, elemente care afișează informația video extrasa din fișierele multimedia folosite de către aplicație și inițializează diferite fire de execuție și asigura comunicarea intre acestea. |

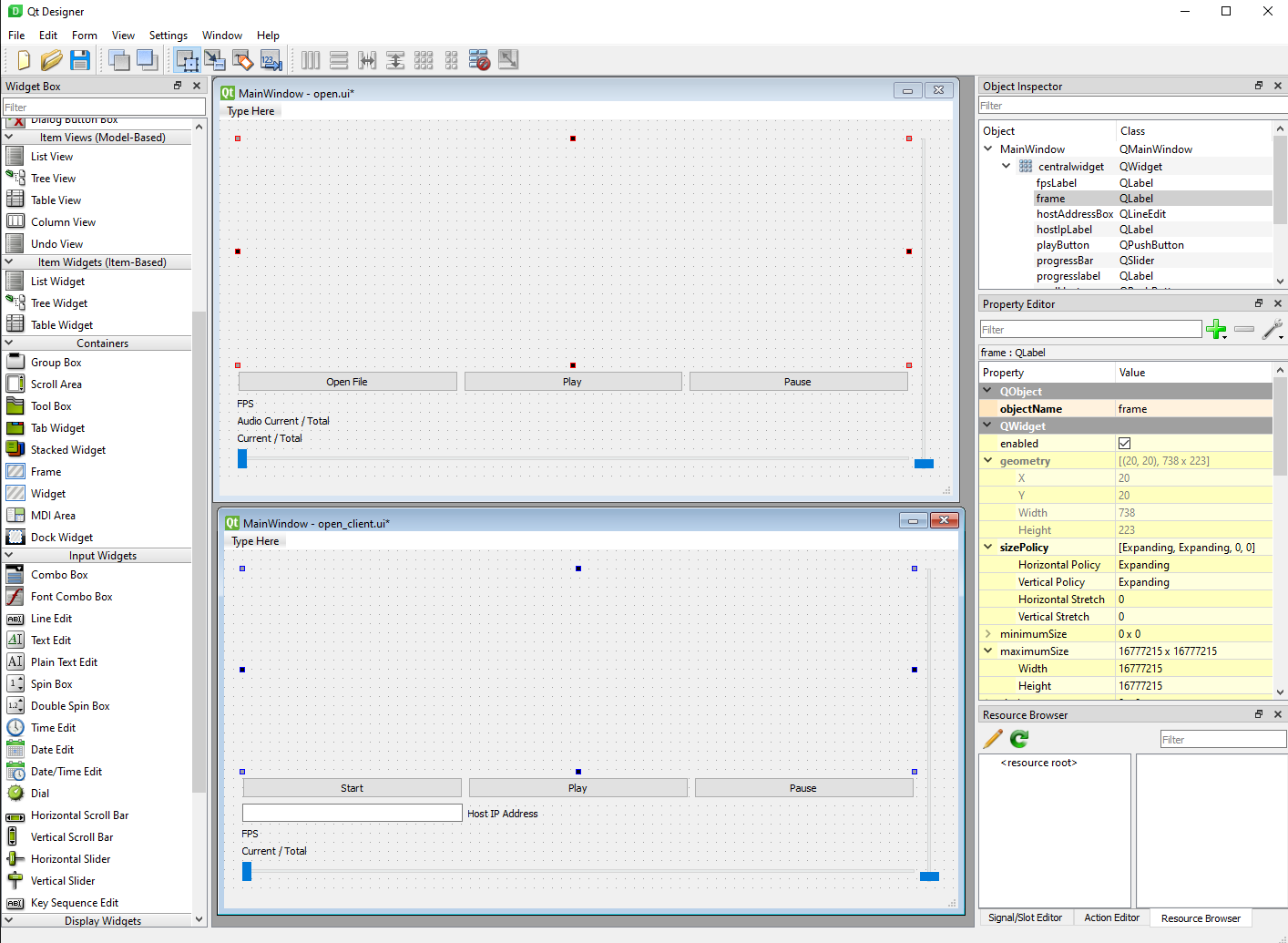


Figura 2‑1- Qt Designer pentru generarea interfeței

OpenCV (Open Source Computer Vision Library) este o librărie ce a fost folosită pentru extragerea și procesarea imaginilor din fișierele multimedia.

FFmpeg este o colecție de librarii și tool-uri folosite la nivel de linie de comanda pentru manipularea informațiilor multimedia cum ar fi audio, video, subtitrărilor și a metadatelor. Acesta a fost utilizat în dezvoltarea aplicației cu scopul de a citi și extrage informația audio și generarea unui fișier de tip *wave* compatibil cu librăria de procesare audio folosit în aplicație.

PyAudio este o legătură către PortAudio, o librărie open-source ce se ocupa cu redarea informațiilor audio primite de către fișierul generat de librăria descrisa mai sus.

pycaw (Python Core Audio Windows Library) este o librărie ce permite obținerea și setarea valorilor de volum ale aplicatiei la nivel de sistem de operare. Aceasta librărie ne limitează, din păcate, la un singur tip de sistem de operare: Microsoft Windows.

PyInstaller este o librărie ce compilează codul aplicației, împreună cu toate dependentele sale într-un singur pachet. Acest pachet va putea fi rulat pe sisteme ce nu conțin un interpretor Python instalat sau module folosite în codul sursa. în momentul de fata, aceasta librărie este capabilă să compileze cod compatibil cu Windows, Mac OS X și Linux. Totuși, nu este cross-platform. O aplicație care este menită să ruleze pe Windows trebuie generată de asemenea din Windows.

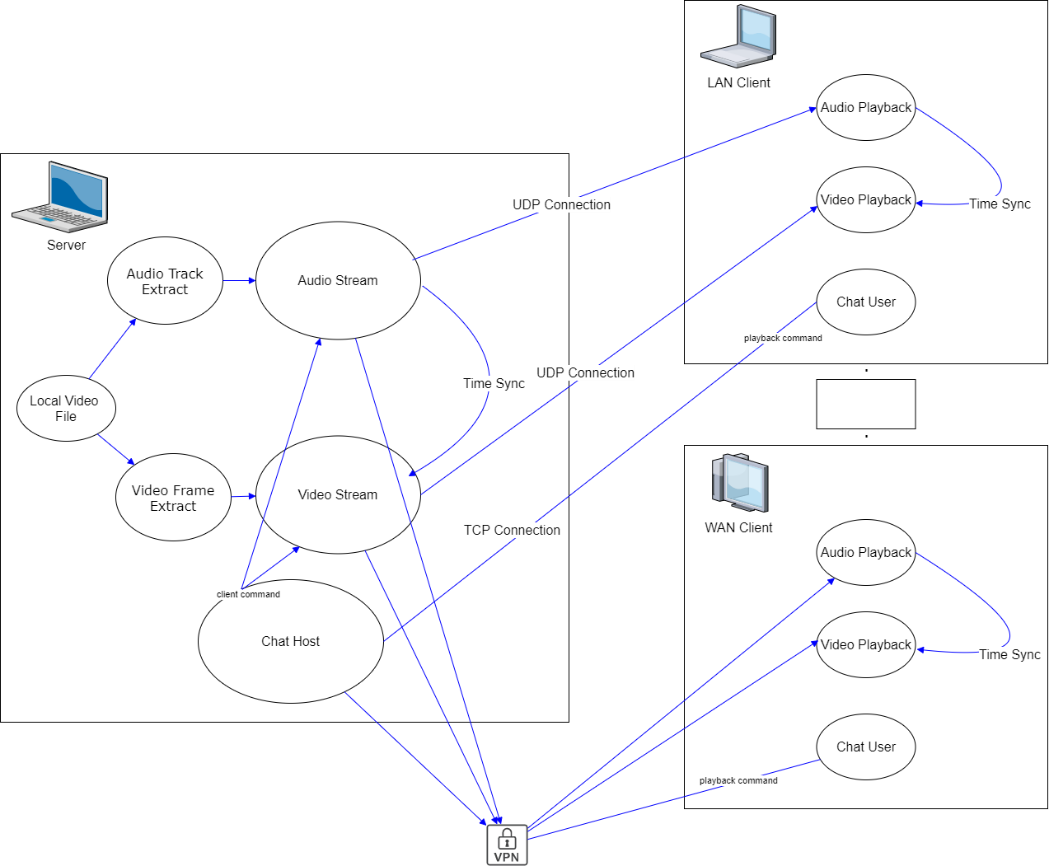
TODO: Radmin VPN pentru a stabili o conexiune intre utilizatori la distanta

TODO: adaugat explicatii pentru cateva obiecte Qt ca sa fie intelese in capitolul 3 de implementare

..Protocol UDP folosit pentru transmiterea datelor audio și video catre clienti

..Protocol TCP folosit pentru a crea o conexiune de tip full-duplex unde clientii isi inregistreaza cate un username și pot comunica prin server intre ei. Serverul citeste mesaje de la clienti pentru a primi comenzi de rulare, pauza sau derulare de la clienti.

# Implementarea aplicației



Figură 23‑1 Arhitectura aplicației

Aplicația este împărțită in 5 module pentru server, respectiv 4 module pentru client. Fiecare componenta are un rol si un scop bine definit, unele comunicând intre ele pentru a accesa informații necesare in scopuri precum sincronizări sau emiteri de comenzi si semnale. Modulele au fost împărțite in:

* Modulul principal (server /client)
* Modulul de extragere a cadrelor video din fișier (server)
* Modulul de redare si trimitere/primire a cadrelor video (server /client)
* Modulul de redare si trimitere/primire a segmentelor audio (server /client)
* Modulul de TCP chatroom (server /client)

## Modulul principal

Acesta, in prima faza, are rolul de a inițializa interfața grafica (Figura 3‑1) dintr-un fișier .ui generat cu ajutorul lui Qt Designer (Figura 2‑1) si face legătură dintre elementele sale cu codul de procesare ale celorlalte module.

|  |  |
| --- | --- |
| Figura 3‑1- Interfața grafică si elementele sale |  |

**1.** Elementul de tip QPushButton conține următoarele funcționalități si caracteristici pentru server:

* Marcat cu textul „Open File”.
* După selectarea unui fișier, acesta se asociază unei instanțe de tip VideoCapture din OpenCV folosite in operațiile video si extragerea FPS[[2]](#footnote-2)-ul luat din metadata fișierului.
* Se generează un nou fișier audio cu numele „temp.wav” ce va fi folosit in transmiterea si afișarea segmentelor audio. Aceasta generare are loc cu o instrucțiune la nivelul liniei de comanda, folosind FFmpeg. Sunt incluși câțiva parametrii care setează fișierul generat cu bitstream/număr de canale/rata de redare prestabilite si opțiunea de suprascriere activa.
* Se inițializează o variabila de tip coada unde putem seta o mărime maxima. Aceasta variabila are rolul de a funcționa ca un intermediar intre producător – consumator pentru extragerea si redarea cadrelor video in doua fire de execuție diferite.
* Se inițializează firul de execuție pentru extragerea cadrelor video.
* Se inițializează firele de execuție pentru trimiterea si afișarea cadrelor video si audio. Daca au fost deja inițializate, firele de execuție sunt distruse si pornite altele pentru a actualiza metadatele noului fișier selectat.
* Se inițializează firul de execuție pentru găzduirea chatroom-ului si așteptarea clienților. Daca a fost deja inițializat, atunci el va face legătura cu noile fire audio/video create si le transmite lista curenta a adreselor clienților
* Se inițializează o variabila de tip pycaw care face legătura cu mixerul audio al sistemului de operare (valabil doar cu Windows)

Pentru client:

* Marcat cu textul „Start”
* Se inițializează firele de execuție pentru primirea si afișarea cadrelor video si audio.
* Se inițializează firul de execuție pentru conectarea la chatroom-ul ținut de server. Interfața client-ului va conține un text box in care se introduce IP-ul server-ului
* Se inițializează o variabila de tip pycaw care face legătura cu mixerul audio al sistemului de operare (valabil doar cu Windows)

**2.** Element de tip QLabel este transmis către firul afișării cadrelor video.

**3.** Doua elemente de tip QPushButton trimise in initializarea firelor de redare audio/video pentru a opri/reporni derularea.

**4.** Element de tip QSlider trimis in initializarea firului de redare video pentru a afișa sau seta poziția in timp a conținutului.

**5.** Element de tip QLabel ce afiseaza timestamp-ul in format hh:mm:ss.ms in care se afiseaza timpul curent/timpul total. Pe partea de server vor fi doua etichete: una pentru timestamp-ul video si cealaltă pentru timestamp-ul audio, folosite pentru debug.

**6.** Element de tip QSlider ce este folosit pentru reglarea volumului aplicației

**7.** Element de tip QLineEdit folosit in interfata client pentru a introduce adresa serverului

## Modulul de redare si trimitere/primire a segmentelor audio

In inițializarea modulului se creează legăturile cu elementele necesare din interfața, se setează socketul UDP si se creează obiectul de tip *PyAudio* responsabil cu redarea sunetului.

Din modulul principal, fișierul deschis va fi folosit împreuna cu librăria *fmmpeg* pentru a genera un fișier wave folosit in stream. Fișierul va fi reconstruit cu o frecventa de 44100 Hz, un bitrate de 160 kbps si 2 canale pentru redare stereo. Reconstruind fișierul cu aceste valori cunoscute, vom știi ce parametrii de intrare vor fi folosiți pentru obiectul redării audio pe partea de client (Tabel 6).

|  |
| --- |
| self.p = pyaudio.PyAudio()  self.CHUNK = 1024  self.stream = self.p.open(format=self.p.get\_format\_from\_width(2),  channels=2,  rate=44100,  output=True,  frames\_per\_buffer=self.CHUNK) |

Tabel 6 - Initializarea obiectului de stream din client cu valorile folosite in generarea fisierului wave

Pe partea de server, fișierul generat se poate deschide folosind librăria nativa *wave* din python cu scopul transmiterii datelor către clienți. In funcția principala a modulului se citesc datele din obiectul wave pe bucati de cate 1024 de octeți. Actualizăm timestamp-ul audio in interfața, trimitem datele către adresele clienților prin socket-ul UDP si le scriem in obiectul pyaudio pentru redarea sunetului in aplicația server-ului. Această funcție va fi repetată la o anumita frecvență, folosind un obiect QTimer atașat de funcțiile legate de butoanele Play si Pause, similar cu timerul din Modulul de redare si trimitere/primire a cadrelor video. Încă o funcție va mai fi creată si legata de slider-ul din interfață. Folosind funcția wave.setpos() setăm poziția aleasă din slider.

Pe partea de client deschidem un fir de execuție separat care așteaptă datele de la server prin socket si le introduce într-o coada de așteptare. Funcția de redare audio citește aceste date din coada si le scrie in obiectul de stream audio.

Coada de așteptare nu trebuie sa aibă o mărime prea mare, deoarece redarea sunetului va fi continuata pana la golirea cozii in aplicația clienților, chiar si după ce aplicația server pune pauza.

## Modulul de redare si trimitere/primire a cadrelor video

In prima faza, se conectează elementele 2, 3, 4 si 5 ale interfeței grafice (Figura 3‑1) si se primesc informații din Modulul principal. Un socket de tip UDP va fi inițializat cu scopul de a trimite cadrele video către lista de IP a clienților, respectiv primi cadrele de la server. Lista din server este actualizata de către Modulul de TCP chatroom la fiecare prima inițializare sau la fiecare conectare a unui nou participant in chat.

Funcționalitatea primara a modulului pe partea de server este descrisa astfel:

Se ia un cadru cu numărul sau de secvență din coada comuna cu Modulul de extragere a cadrelor video. Folosind apoi următoarea formula, se calculează timestamp-ul curent, împreuna cu timestamp-ul total al derulării video:

|  |  |
| --- | --- |
|  | (actualizat in eticheta de timestamp a video-ului la fiecare cadru cu formatarea hh:mm:ss.ms) |

Tabel 1 - Calcul timestamp in secunde pentru video

TODO: impactul codificării in JPEG cu librăria OpenCV in cod si serializarea datelor din punct de vedere al mărimii datelor

Se stochează cadrul împreună cu numărul sau de secvență si timestamp-ul într-un dicționar, acestea urmând a fi serializate si trimise către lista ce conține IP-urile clienților, actualizata de firul de execuție a chatroom-ului in modul descris mai jos.

Se va afișa cadrul video in elementul 2 din interfața (Figura 3‑1). Modulul repeta aceste operații pana când va fi semnalata oprirea sau mutarea poziției in timp folosind bara de progres.

### Modulul de extragere a cadrelor video

Valabil doar pe partea de server. Acesta este inițializat in Modulul principal si are rolul de producător pentru redarea si trimiterea cadrelor video, împreună cu numărul lor de secvență, folosind o coada de stocare comuna cu Modulul de redare si trimitere/primire a cadrelor video. Funcționează ca un buffer pentru redarea parții video.

Aici putem modifica cadrele. Singura modificare prezenta la momentul actual este micșorarea mărimii cadrelor pentru a ne permite transmiterea lor prin UDP. Tot in acest modul tratam posibile erori, cum ar fi lipsa anumitor cadre din fișier.

## Metoda de sincronizare audio-video

Din cauza faptului ca viteza de procesare si afișare a fiecărui cadru video variază, trebuie sa aplicam o metoda care ajustează funcția de afișare la o anumita frecventa reprezentata de variabila FPS extrasa din metadata fișierului. Folosim următorul pseudocod:

|  |
| --- |
| timp\_de\_asteptare = 1 / fps\_metadata  tpf[[3]](#footnote-3) = timpul\_curent()  repeta cat timp avem cadre si status = redare:  proceseaza\_si\_afiseaza\_cadru()  fps\_actual = 1 / (timpul\_curent() - tpf)  tpf = timpul\_curent()  daca fps\_actual > fps\_metadata:  mareste timp\_de\_asteptare  altfel:  micsoreaza timp\_de\_asteptare  asteapta\_in\_secunde(timp\_de\_asteptare) |

Tabel 2 - Pseudocod de sincronizare cu FPS-ul din metadata

Pentru a aplica timpul de așteptare, folosim o noțiune din librăria PyQt numita QTimer, care funcționează ca un cronometru cu un timp de așteptare stabilit. Se face legătura cu o funcție si, odată ce a fost pornit timer-ul, așteaptă cu timpul stabilit si aplica funcția. Timpul este apoi resetat la valoarea inițială si se repeta cat timp timer-ul rămâne in statusul activ. Se poate actualiza oricând in timpul execuției un timp de așteptare nou ce va fi aplicat la următoarea resetare. Statusul timer-ului se poate modifica in activ/inactiv in interiorul firului de execuție in care a fost creat.

Timer-ul nu poate fi oprit sau pornit de către alt fir de execuție. Pentru a comunica acestuia oprirea/pornirea din exteriorul firului de execuție, se folosește o alta noțiune din librăria PyQt numita pyqtSignal. Se creează un atribut al clasei de acest tip si se face legătura acestuia cu o funcție definita in interior. Atâta timp cat avem acces la instanța acelei clase, putem folosi atributul din exterior pentru a semnala firului sa execute funcția (in cazul de fata, semnalam firului de execuție sa oprească/pornească timer-ul din interior). Astfel, un modul exterior poate porni/opri acel timer de frecventa atunci când îndeplinește anumite condiții, cum ar fi o cerere de la un client conectat.

Aplicând sincronizarea descrisa in Tabel 2, redarea cadrelor va da o senzație de mișcare normala. Totuși acesta nu asigura sincronizarea cu redarea audio.

Rulând segmentele din Modulul de redare si trimitere/primire a segmentelor audio, s-a observat o redare naturala a sunetului, fără sa fie nevoie de reglări. Astfel, ne putem folosi de timestamp-ul audio pentru a aplica reglări in frecventa de afișare a cadrelor video. Putem include instanța firului de execuție audio pentru a accesa momentul curent in care se afla redarea.

Înlocuind in Figura 3‑2 variabilele de FPS cu timestamp-ul audio, respectiv video calculate in (TODO: figurile timestamp audio si video), obținem următorul cod:

|  |
| --- |
| timp\_de\_asteptare = 1 / fps\_metadata  repeta cat timp avem cadre si status = redare:  proceseaza\_si\_afiseaza\_cadru()  timestamp\_video = nr\_secventa\_video / FPS\_metadata  timestamp\_audio = TODO:  daca timestamp\_video > timestamp\_audio:  mareste timp\_de\_asteptare  altfel:  micsoreaza timp\_de\_asteptare  asteapta\_in\_secunde(timp\_de\_asteptare) |

Tabel 3 - Pseudocod de sincronizare cu timestamp audio

Aplicând numai pseudocodul descris in Tabel 3, timestamp-urile video si audio vor fi sincronizate, insa redarea video face schimbari prea bruste la frecventa de afisare.

Aplicând cele doua metode simultan (Tabel 4), reușim sa sincronizam redarea audio cu video, iar frecventa cadrelor video va fi stabila:

|  |
| --- |
| timp\_de\_asteptare = 1 / fps\_metadata  tpf = timpul\_curent()  repeta cat timp avem cadre si status = redare:  proceseaza\_si\_afiseaza\_cadru()  timestamp\_video = nr\_secventa\_video / FPS\_metadata  timestamp\_audio = TODO:  daca timestamp\_video > timestamp\_audio:  mareste timp\_de\_asteptare  altfel:  micsoreaza timp\_de\_asteptare    fps\_actual = 1 / (timpul\_curent() - tpf)  tpv = timpul\_curent()  daca fps\_actual > fps\_metadata:  mareste timp\_de\_asteptare  altfel:  micsoreaza timp\_de\_asteptare  asteapta\_in\_secunde(timp\_de\_asteptare) |

Tabel 4 - Pseudocod de sincronizare cu timstamp audio si FPS-ul din metadata

Folosind un fișier audio-video, putem analiza schimbările in FPS-ul actual in timpul rulării aplicației. Reamintim ca FPS-ul actual a fost calculat cu formula:

FPS\_actual = 1 / (timpul\_curent() – timp\_cadru\_precedent())

Astfel, dintr-un set de date de 1500 de cadre (1 minut de redare cu 25 FPS) am extras FPS-ul actual calculat pentru fiecare cadru cu toate cele 3 metode de sincronizare:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Sync cu FPS metadata | Sync cu audio timestamp | Sync cu ambele metode |
| Medie FPS | 25.985 | 31.96343 | 27.18817 |
| Deviație Standard | 6.79591 | 35.19043 | 8.419155 |

Tabel 5 - Comparații intre metodele de sincronizare audio și video

In concluzie, pseudocodul din Figura 3‑5 poate fi folosit ca sa ținem redarea ambelor module sincronizata si, in același timp, redarea cadrelor video nu va fi deranjanta ochiului uman.

## Modulul de TCP chatroom

Modulul va fi inițializat pe partea de server cu un socket TCP care așteaptă conexiuni si o lista ce va fi folosita in transmiterea adreselor clienților către firele de execuție audio si video. Va fi executata o funcție ce îndeplinește următoarele roluri:

* așteaptă si accepta conexiuni noi si confirma către client printr-un mesaj
* trimite un mesaj prin care cere clientului conectat sa introducă un username
* anunță participanții ca s-a conectat un utilizator nou
* adaugă adresele IP ale conexiunilor noi într-o lista
* trimite lista firelor de redare audio si video
* pornește cate un fir de execuție pentru fiecare client care executa o functie numita *handle*

Funcția *handle* așteaptă alte mesaje si verifica daca acestea sunt comenzi playback. Astfel, au fost create 3 comenzi:

* „/pause” trimite cate un semnal firelor de execuție pentru redare audio si video sa oprească execuția
* „/play” trimite cate un semnal firelor de execuție pentru redare audio si video sa reînceapă execuția
* „/skipto <număr>” accesează firele de redare audio/video si le trimite numarul la care sa se repoziționeze. Acest număr reprezintă numărul de secvența al cadrelor video

In caz de eroare, clientul este considerat scos din sesiune. Se actualizează lista de adrese a clienților in firele de redare audio/video si serverul anunța restul participanților cine a ieșit.

Pe partea de client, la inițializare, se face legătura cu socketul creat in Modulul principal si se așteaptă un input de username de la tastatura. Se încerca apoi realizarea conexiunii cu serverul prin adresa introdusa in elementul 7 din interfață (Figura 3‑1- Interfața grafică si elementele sale).

In funcția principala se deschid 2 fire de executie:

* un fir așteaptă input-uri de la tastatura, atașează numele si mesajul într-o structura de tipul {„user”: user, „msg”: message}, o serializează si o trimite prin socket-ul TCP către server
* un fir așteaptă mesaje de la server, acestea fiind de fapt mesaje trimise de alți utilizatori către server si transmise către toți participanții sesiunii

# Concluzii

TODO:

La final discut despre probleme pe care le-am întâlnit incercand sa implementez partea video doar cu librăria opencv – suportul pentru interfața nu e utilizabil pentru un video player sau nu contine tot ce am nevoie. Implementarea barei de progres putea fi facuta doar cu setarea lui, actualizarea barei blocheaza complet procesarea frame-urilor și nu putea fi despartita pentru a putea interactiona cu ea pe un fir de executie separat. Lipsa de suport pentru adaugarea altor elemente în interfata: butoane de play/pauza, timestamp sau FPS counter puteau fi adaugat doar în frame-ul video ceea ce nu este ideal. PyQt rezolva toate problemele mentionate mai sus

Despre ce as adauga la aplicate: suport pentru subtitrari, controlul calitatii video-ului din interfata, interfata dedicata pentru chat în loc de command line, impartirea frame-urilor video pe bucati și reconstruirea lor la client pentru a avea calitate video mai mare atata timp cat ne permite viteza conexiunii.

Transmiterea frame-urilor video se realizeaza în clasa de playback din server. Din cauza asta queue-ul video de la clienti se incarca cu maxim un frame. Queue-ul audio functioneaza ca un buffer pentru clienti dar daca setam o marime mare atunci avem lag intre video și audio, daca setam o marime mica atunci calitatea audio-ului nu este foarte buna (se aude cu mici intreruperi intre frame-uri audio). Pentru a seta un queue mai mare la audio putem sincroniza pe partea clientului audio și video cum am facut la server. De asemenea trebuie trimise frame-urile video din clasa de Video Generation de la server, nu de la video playback, pentru a putea umple buffer-ul clientilor.

Contributii: construirea și setarea interfetei, repornirea thread-urilor necesare pentru a avea noile metadate, despre cum reglez rezolutia video-ului pentru a incape în datagrame, folosirea queue-ului în video gen pentru ca cv2.read() consuma timp de procesare și ar fi fost inutil daca il adaugam în partea de playback, am folosit timere pentru ca aveam nevoie de functii repetate la o anumita frecventa (detaliat conceptul de timer), cum am folosit semnale ca sa pot opri/porni timerele din exterior (mai exact cand un client trimite o comanda catre server), cum libraria audio garanteaza playback-ul la o viteza ideala asa ca am folosit timestam-ul de la audio ca sa sincronizez frecventa afisarii video-ului, impreuna cu sincronizarea fps-ului din metadata pentru o derulare stabila a videoului (sa nu aibe schimbari prea bruste în framerate, mecanism de corectare a erorilor pentru a isi pastra integritatea), aveam nevoie de format .wav ca sa poata fi integrata cu libraria pyaudio ce permite difuzarea și transmiterea pe segmente, ordinea de activare a thread-urilor ca sa poata fi realizata comunicarea intre ele: thread-urile audio/video pornite primele ca sa putem folosi semnalele de timer în constructorul thread-ului tcp chat / setarea adreselor clientilor în thread-urile audio video chiar și dupa reconstruirea acestora, thread-ul audio pornit inaintea thread-ului video ca sa putem accesa timestamp-ul audio-ului folosit pentru sincronizarea video

Bibliografie

|  |  |
| --- | --- |
| [1] | S. Ponlatha și R. S. Sabeenian, „Comparison of Video Compression Standards,” *International Journal of Computer and Electrical Engineering,* vol. 5, nr. 6, 2013. |
| [2] | Axis Communications, „An explanation of video compression techniques,” 2008. |
| [3] | A. Dashti, H. K. Seon și C. Shahabi, „Fundamentals of streaming media systems,” în *Streaming Media Server Design*, Prentice Hall, 2003. |

Nume litera-prenume autor (an) – titlu, editura

1. Integrated Services – arhitectura care specifica elementele ce garantează calitatea serviciilor (QoS) [↑](#footnote-ref-1)
2. FPS – cadre pe secunda (frames per second) [↑](#footnote-ref-2)
3. tpf = momentul in care a fost afișat cadrul precedent (time previous frame) [↑](#footnote-ref-3)