# CS3001 – Computer Networks

## Final Project Report

## Client-Server / P2P Video Streaming Using Python


### Group Members

| Name | Roll Number |
|---|---|
| Arham Ali | 23K-0637 |
| Muhammad Hassan Shaikh | 23K-0830 |
| Fareed Aslam | 23I-2632 |


### 1. Introduction

Real-time video communication has become an essential part of modern digital applications. This project implements a hybrid Client-Server / P2P video streaming system using Python, TCP sockets, and OpenCV. The system allows users to stream videos in real-time, join streams as viewers, chat live, and view detailed network statistics including FPS, latency, bandwidth, and packet loss.

### 2. Project Objectives

The main objectives of the system are:

- • Develop a hybrid P2P/Client-Server streaming model.
- • Enable real-time transmission of video frames.
- • Ensure reliable communication using TCP.
- • Minimize latency and packet loss.
- • Provide smooth playback using OpenCV.
- • Support multiple viewers simultaneously.
- • Allow per-user quality selection (360p, 480p, 720p, 1080p).
- • Implement real-time chat among all connected users.

### 3. System Architecture

The system uses a hybrid Client-Server approach. The server manages connections, synchronization, and broadcast of frames. The client encodes frames into JPEG, sends them to the server, and decodes received frames for playback.

Key components:

- • TCP Sockets – for reliable byte-stream transmission.
- • OpenCV – for capturing, encoding, decoding, and resizing video frames.
- • Base64 Encoding – for safe JSON transmission.
- • Tkinter GUI – for user interface, chat, and stream controls.
- • Threading – for parallel video sending, receiving, and GUI responsiveness.

### 4. Features

- ✓ Live Video Streaming – Streamer sends JPEG frames; viewers receive in real-time.
- ✓ Live Chat – All clients can communicate during the stream.
- ✓ Resolution Control – Each viewer selects between 360p, 480p, 720p, 1080p independently.
- ✓ Stream Statistics – FPS, bandwidth, latency, and packet loss are calculated live.
- ✓ Sequence Numbers – Ensures packet ordering and packet loss detection.
- ✓ Multi-Viewer Support – Unlimited clients can join a stream.
- ✓ Streamer Lock – Only one streamer allowed at a time.
- ✓ GUI-based login screen with option to Stream or Join.

### 5. Working of the System

5.1 Starting the Application:
• User enters a username.
• Chooses either 'Start Streaming' or 'Join Stream'.

5.2 If Streamer:
• Selects video file (MP4/AVI/MKV).
• Starts stream; frames are encoded, sent to server, and broadcast.
• GUI shows real-time FPS, bandwidth, latency, packet loss.

5.3 If Viewer:
• Views the video from the active streamer.
• Adjusts quality for bandwidth management.
• Receives live stats and participates in chat.

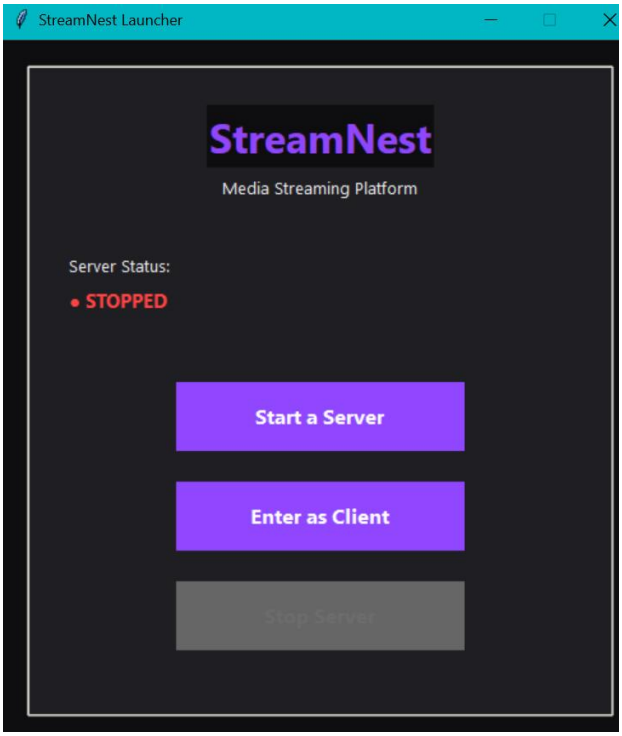
### 6. Network Statistics Implementation

FPS: Calculated every 0.5 seconds based on processed frames.
Latency: Each frame carries a timestamp; client computes RTT/2.
Packet Loss: Sequence numbers are used to detect skipped frames.
Bandwidth: Calculated from bits received per second using a sliding window.

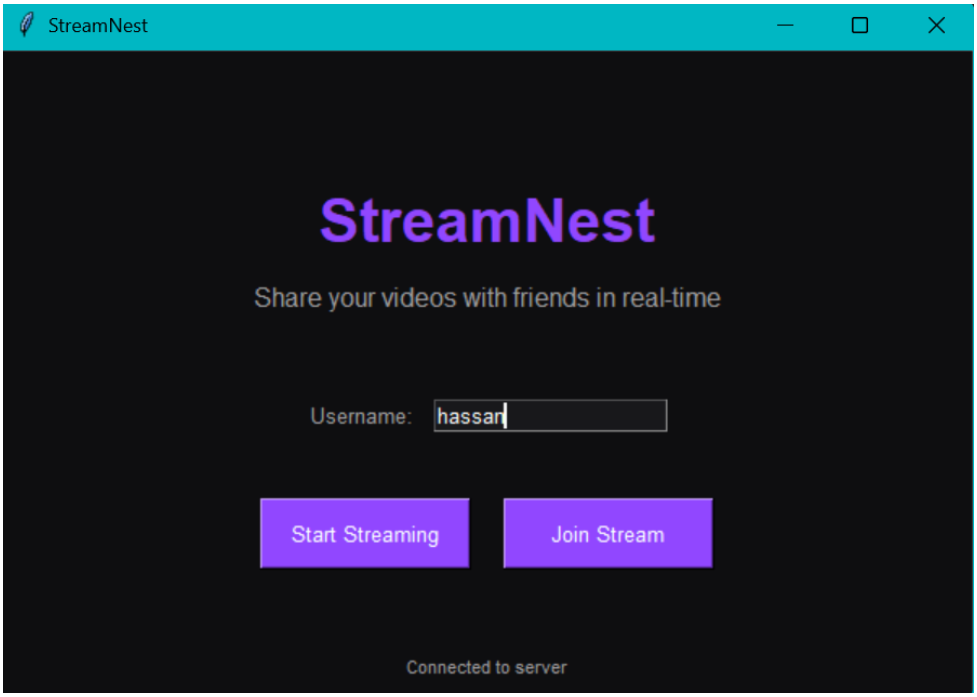
### 7. Conclusion

The project successfully demonstrates a real-time video streaming system built on core computer networks concepts such as TCP communication, packet framing, latency handling, and synchronization. The system integrates multimedia processing, GUI design, and network programming — making it a complete Computer Networks course project.

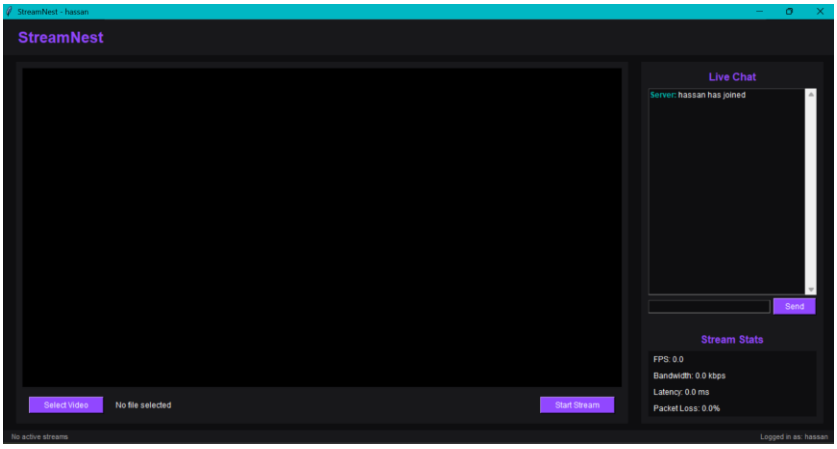
➔ Screenshots :




➔ Can either join a stream or start a stream:

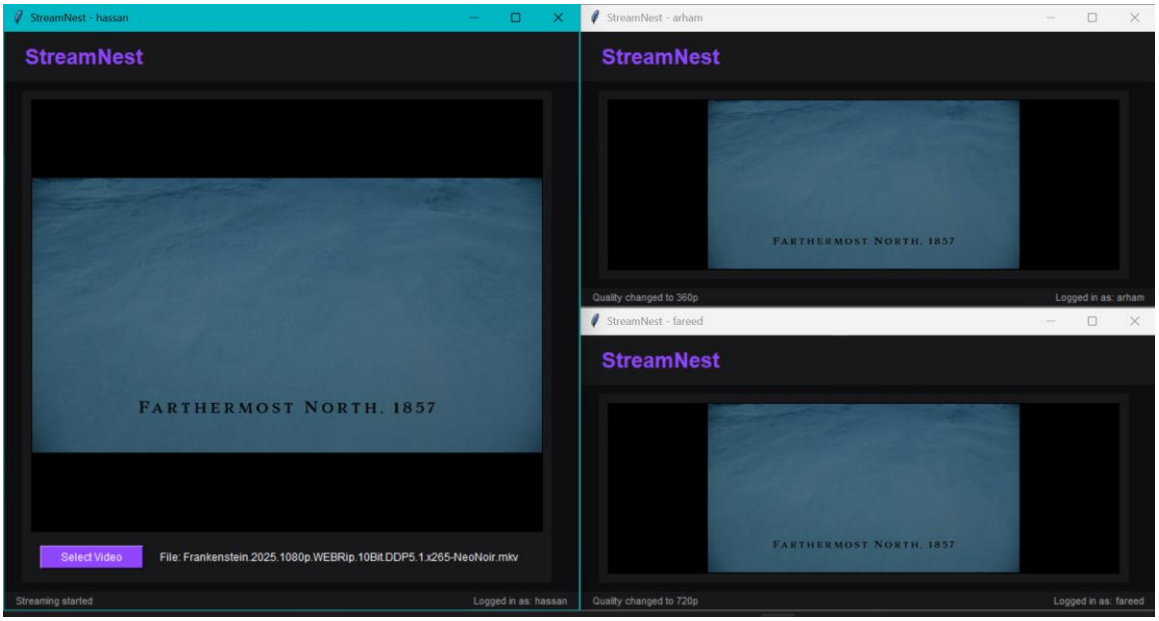



->Initial screen(as streamer) :

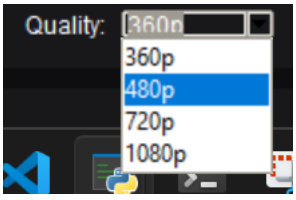



Select video to stream
And starts stream initiates the stream

->N numbers of viewers can join to a Stream:

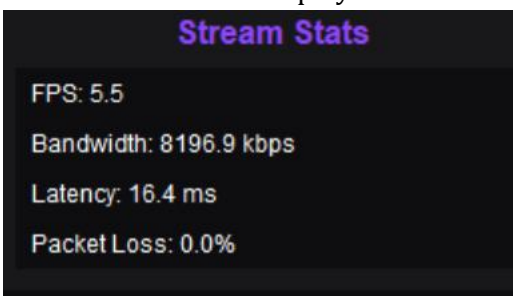



And viewers can change their resolutions based on their need




Here 3 people have joined hassan streams in 1080
Arham views in 360p
Fareed views in 480p

➔ Stream stats are displayed to each client




➔ Clients connected to the server can live chat