Project Title

StreamSocket: A Real-Time Video Streaming Solution

Submitted by

Ayesha Islam Qadri **2020-CE-36**

Ayesha Shafique **2020-CE-39**

Submitted to

Ma'am Darkhshan Abdul Ghaffar

Course

CMPE333L Computer Networks

Semester

6th

Date

23rd February 2023

Department of Computer of Engineering

University of Engineering and Technology, Lahore

Table of Contents

Abstract

Due to the explosive growth of the Internet and increasing demand for multimedia information on the web, streaming video over the Internet has received tremendous attention from academia and industry. More and more organizations are using live video streaming as a reliable way to reach their audiences, overcome physical limitations, and help improve their operations. That's because live video streaming is not just a great alternative to in-person activities and other media, but an even better choice and more useful option in certain situations.

Problem Statement

Live video streaming has become a ubiquitous means of communication in various contexts, including remote work, education, entertainment, and social media. However, developing a live video streaming application with audio using socket programming with Python can be a daunting task. In this project, we aim to address this challenge by developing a live video streaming application with audio using socket programming with Python. Our application will leverage the client-server architecture to enable users to stream live videos with audio in real time.

Objective

The goal of this project is to create a lightweight and user-friendly solution for live video streaming that can be tailored to the needs of the user. The Python programming language and socket programming will be used to develop the application. Users will be able to stream live video from a server to multiple clients at the same time using the application. The application will also handle audio streams in real time.

Scope

The proposed project's scope is to design a live video streaming application with audio that can be used in various industries, including remote education, training, live events, meetings, and the entertainment industry. The application will be able to stream video and audio in real time, and users will be able to customize the quality settings. The project's scope is vast, and it can be tailored to meet the specific needs of different industries. It will provide a cost-effective, reliable, and user-friendly solution for organizations and individuals who require live video streaming with audio.

Features

The following are the features of the live video streaming application with audio:

1. The multithreaded server can handle multiple client requests at the same time.
2. A GUI-based application that is easy to use.
3. Support for audio and video streams in real-time.
4. Customizable video and audio quality settings.
5. Start and stop the server and client streams with ease.
6. Support for multiple clients simultaneously.
7. Low latency streaming for real-time communication.
8. Minimal system requirements.
9. User-friendly interface

Explanation

The proposed project aims to develop a live video streaming application with audio using socket programming with Python. The project will have a multithreaded server that can have multiple clients. The application will be a GUI-based Python project that allows users to start and stop the server and client options. The project will utilize socket programming with Python to establish a connection between the server and the client.

The server will receive a live video stream from the camera, which will be encoded and sent to the clients. The server will also support audio streaming to provide a better streaming experience to users. The application will support high-quality video to provide clear video to users. The GUI will be elegant and user-friendly, allowing users to start and stop the server and client options easily. The users can also customize the application to suit their needs. The application will be reliable and efficient, providing a seamless streaming experience to users.

1. Client -Server Archiecture

The proposed live video streaming application with audio will be developed using the client-server architecture. The client-server architecture is a distributed architecture that consists of two components: a server and multiple clients. The server provides services to multiple clients, and the clients request services from the server. The client-server architecture is commonly used in network-based applications and is well-suited for the proposed project.

- **Server**

The server component of the application will be responsible for handling multiple client requests simultaneously. It will be a multithreaded server, which means that it can handle multiple client requests concurrently. The server will receive live video and audio streams from the clients and then broadcast them to all connected clients. The server will also handle audio streams in real time, ensuring that the audio is synchronized with the video.

- **Client**

The client component of the application will be responsible for streaming a live video and audio from the server. Multiple clients can connect to the server simultaneously, and each client will receive the same video and audio stream from the server. The client will also be able to customize the video and audio quality settings to suit their preferences.

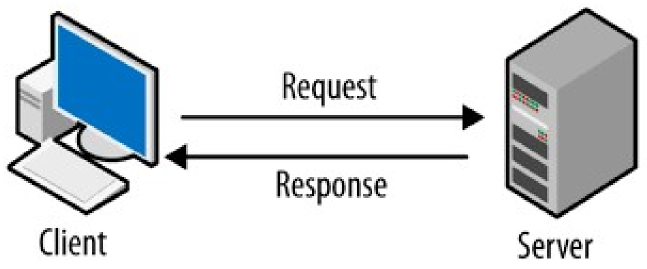

Figure 1 Client-Server Architecture

2. Socket Programming

The application will be developed using the Python programming language and socket programming. Socket programming is a programming interface used for network communication between computers over the internet or a local network. It is a low-level networking API that allows software developers to build network applications that can communicate with other programs over a network. In socket programming, a socket is a software component that serves as the endpoint for a two-way communication link between two programs running over a network.

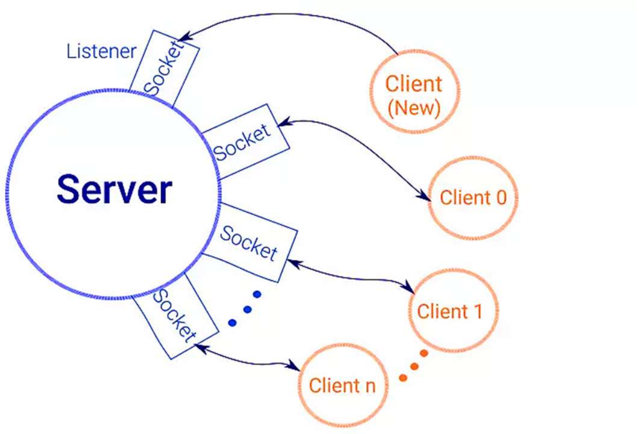

Figure 2 Sockets with Client & Server

3. Socket API Flow Diagram

The below flow diagram explains the Socket API and how socket components bind, listen and accept the connection request between server and client.

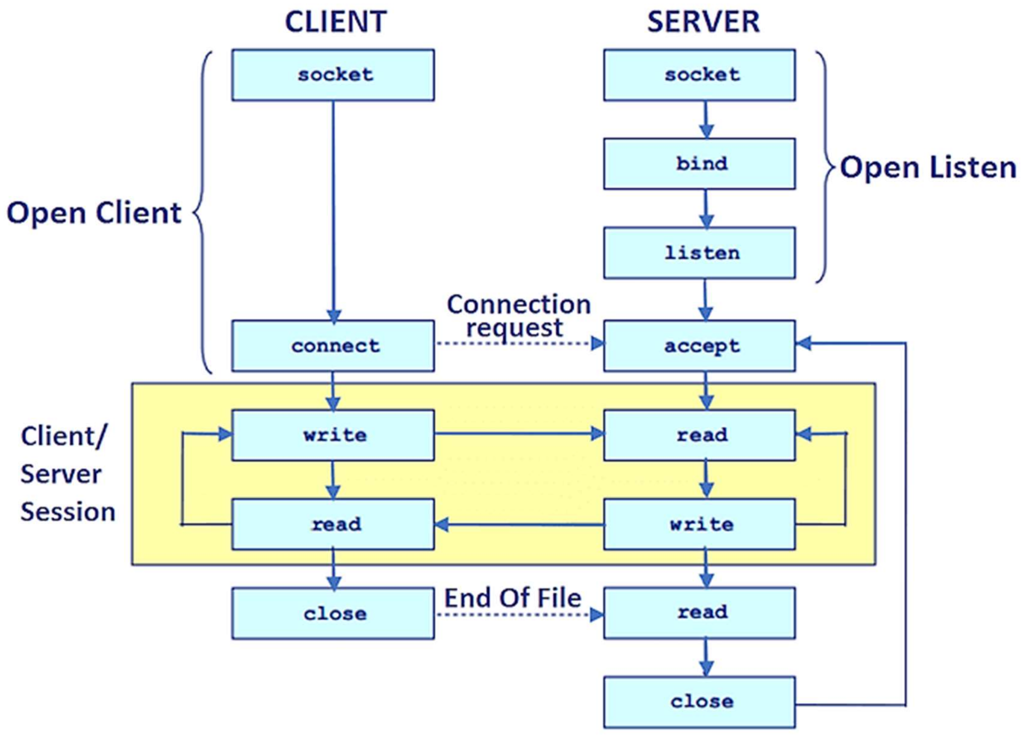

Figure 3 Socket API Flow Diagram

4. Architecture Diagram for Video Streaming

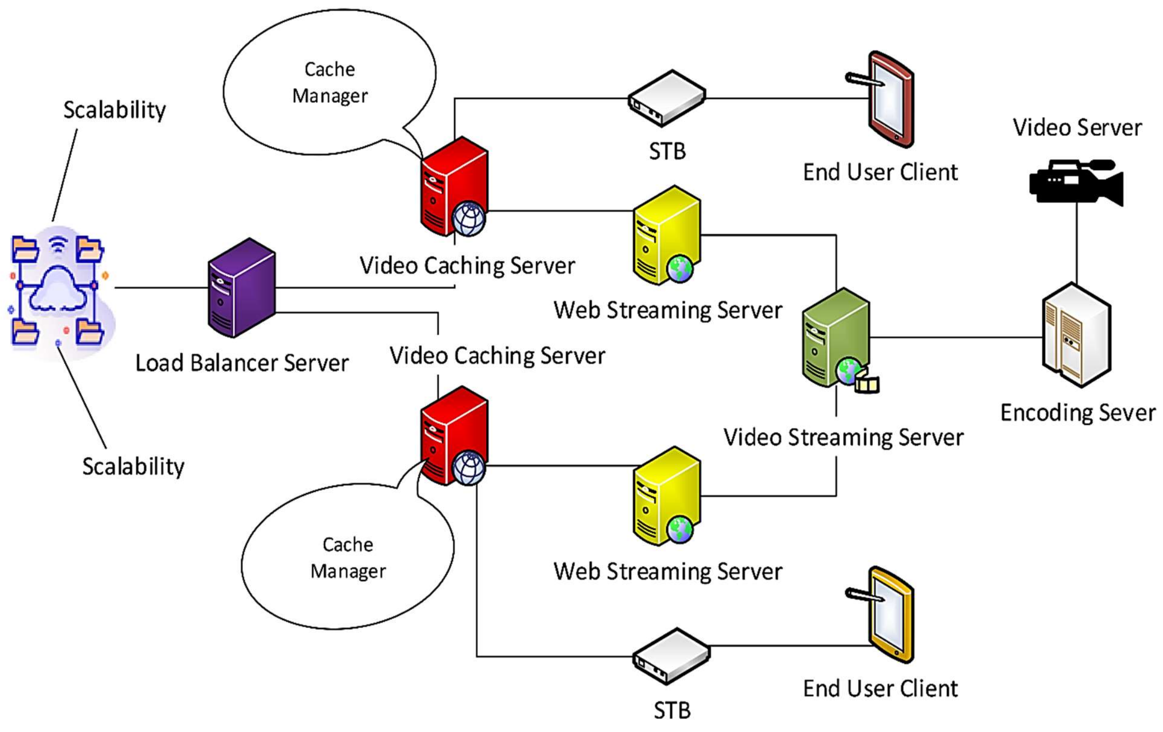

Figure 4 Architecture Diagram for Video Streaming

Here is an architecture diagram for a video streaming system, with an explanation of architecture:

When a user requests to watch a video, the request is first received by the Load Balance Server, which distributes the request to an available Web Streaming Server. The Web Streaming Server serves the initial web page to the user, which includes the video player and controls. When the user clicks the play button, a request is sent to the Video Streaming Server, which begins streaming the video content to the user. The Cache Server stores frequently accessed content to reduce the load on the Video Streaming Server and improve response times.

Before the video content can be streamed, it needs to be encoded into a suitable format for streaming. This is done by the Encoding Server, which optimizes the video quality for different devices and screen sizes. The original video files are stored on the Video Server, which provides access to them for the encoding and streaming servers.

Finally, the STB or end-user client receives the streamed video content and displays it on the user's television or device. The Video Streaming Server handles requests for specific segments of the video to ensure that the video quality is optimized for the user's internet connection. The entire system works together to provide a smooth and seamless video streaming experience for the user.

5. Flow Diagram For Video Streaming

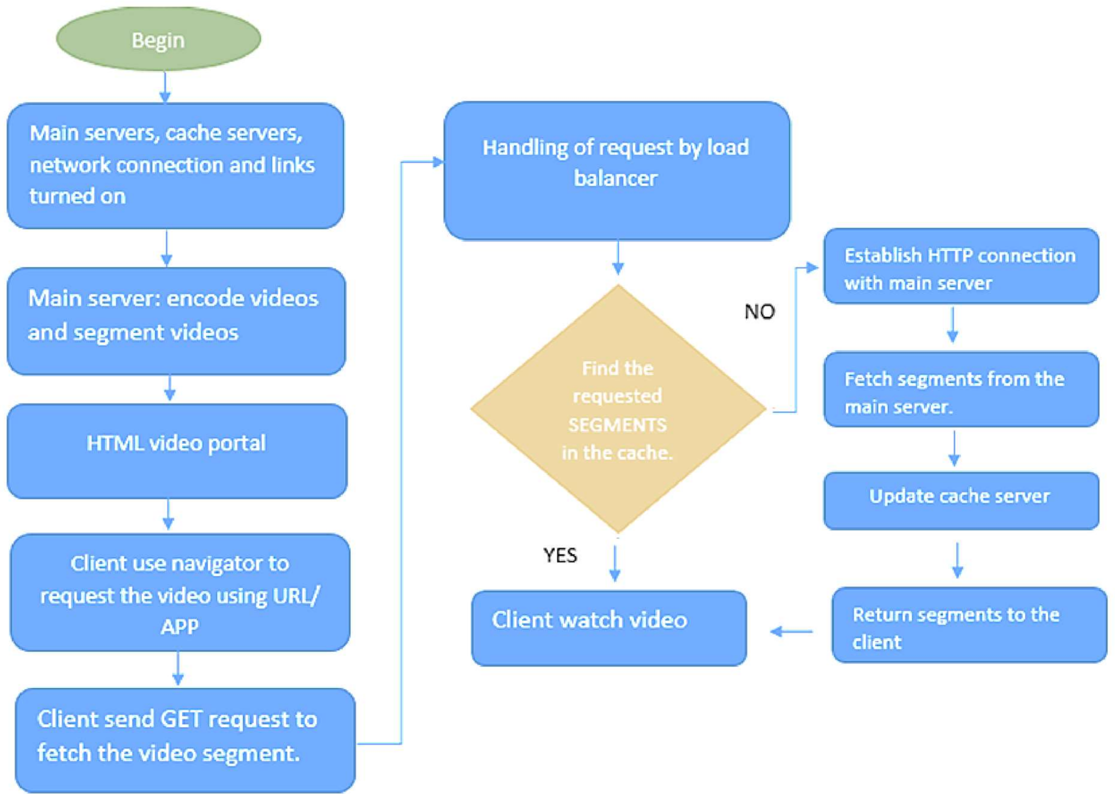
Begin
Main servers, cache servers, network connection and links turned on
Main server: encode videos and segment videos
HTML video portal
Client use navigator to request the video using URL/ APP
Client send GET request to fetch the video segment.
Handling of request by load balancer
Find the requested SEGMENTS in the cache.
NO
YES
Establish HTTP connection with main server
Fetch segments from the main server.
Update cache server
Return segments to the client
Client watch video

Figure 5 Flow Diagram explaining the working of Video Streaming Architecture

This flow diagram outlines the entire process of the proposed video streaming architecture, including the main servers, cache servers, network connections, load balancers, and request handling. Firstly, all servers are powered on and the central server holds encoded video chunks in its master manifest file, which it uploads to the HTML video portal. Users can navigate the portal and request video files.

When a user makes a request, the load balancers handle the initial request and determine which cache server will provide the requested content most efficiently. The cache server checks its cache for the desired content, and if it is not available, it fetches it from the central server via HTTP connection, provides the content to the end user, and saves a copy in its cache for future requests.

References

[1] https://www.mdpi.com/2076-3417/10/21/7691#metrics

[2] https://www.mdpi.com/2076-3417/11/21/9820?type=check_update&version=2