PES University Electronic City Campus

Hosur Rd, Konappana Agrahara, Electronic City, Bengaluru, Karnataka 560100

A Project Report on

"Socket Programming – Video Streaming"

Submitted by:

- **Name (SRN): Pooja Vasudev (PES2UG19EC099)**
- **Name (SRN): Shaswat Jain (PES2UG19EC126)**

Under the guidance of:

Dr. PRAMOD M S

Associate Professor,

Department of Electronics and Communication Engineering

PES UNIVERSITY

Electronic City, Bangalore-560100

2021 - 2022

ABSTRACT

Communication media has become the primary way of interaction thanks to the discovery and innovation of many new technologies. One of the most widely used communication systems today is video streaming, which is constantly evolving. Such communications are a good alternative to face-to-face meetings, and are therefore very useful for coping with many problems caused by distance.

In video streaming, clients request compressed video files, which are resident on servers. The servers can be ordinary web servers, or can be special streaming servers tailored for the video streaming application. The files on the servers can contain any type of video content, including a movies, television shows, recorded sporting events, etc.

In this project, upon client request, the server directs a video file to the client by sending the file into a socket. TCP socket connections are used in project. Before sending the video file into the network, the file is segmented, and the segments are typically encapsulated with special headers appropriate for video traffic.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Today, anyone with a fast enough Internet connection can stream high-definition movies or make a video call over the Internet. This is possible because of a technology called streaming.

Streaming is the continuous transmission of audio or video files from a server to a client. In simpler terms, streaming is what happens when consumers watch TV or listen to podcasts on Internet-connected devices. With streaming, the media file being played on the client device is stored remotely, and is transmitted a few seconds at a time over the Internet.

Sockets and the socket API are used to send messages across a network. They provide a form of inter-process communication (IPC). The network can be a logical, local network to the computer, or one that's physically connected to an external network, with its own connections to other networks. The most common example is the internet.

1.2 MOTIVATION

In the current scenario video streaming has become more popular due to the pandemic in order to socialize, overlook, connect, learn and so much more.

The most common type of socket applications are client-server applications, where one side acts as the server and waits for connections from clients, this is the type of live video streaming application we are going to build.

1.3 OBJECTIVE

To build a live video streaming application without audio using socket programming with python.

CHAPTER 2

LITERATURE SURVEY

Sl.no.	Paper Details	Description	Drawbacks	Enhancements
1.	• Metzger, Florian & Liotou, Eirini & Moldovan, Christian & Hossfeld, Tobias. (2016). TCP Video Streaming and Mobile Networks: Not A Love Story, But Better With Context. Computer Networks. 109. 10.1016/j.comnet.2016.06.033.	• TCP Video Streaming and Mobile Networks	• TCP-based video streaming and mobile networks interactions are not easy to uncover due to the sheer complexity of mobile networks and the lack of tools for proper investigations in the past. While the former issue can only be unraveled through strenuous, time-intensive research, work conducted in this paper can help improve the situation around the latter issue.	• Successive prediction of blind spot events can be tackled by a mixture of cross-layer information exchange, crowdsourcing, and Big Data.

Sl.no.	Paper Details	Description	Drawbacks	Enhancements
2.	• Bing Wang, Jim Kurose, Prashant Shenoy, and Don Towsley. 2008. Multimedia streaming via TCP: An analytic performance study. Multimedia Comput. Commun. Appl.</i> 4, 2, Article 16 (May 2008), 22 pages.	• Multimedia streaming via TCP: An analytic performance study	• The performance of TCP streaming is not solely determined by T /μ but is sensitive to the values of the various parameters in the models . For large RTTs, high loss rates and timeout values, to achieve a low fraction of late packets, either a long start up delay or a large T /μ (greater than 2) is required.	• In the paper fraction of loss rate as the performance metric is used . Performance study using more complicated and user-oriented metrics is left as future work.

Sl.no.	Paper Details	Description	Drawbacks	Enhancements
3.	• Usuff, Rahamathunnisa & Ramakrishnan, Saravanan. (2013). A survey on video streaming over multimedia networks using TCP. Journal of Theoretical and Applied Information Technology. 53. 205-209.	• A survey on video streaming over multimedia networks using TCP	• TCP has certain limitations with video streaming, the buffer management schemes and congestion control methods involved in it improves the video streaming which is still a challenging problem	• Past researchers have focused on the window size and retransmission procedures and our future direction of research will be towards improving quality of service in video streaming through TCP.

CHAPTER 3

METHODOLOGY

For Camera live-streaming app over the network, TCP (Transmission Control Protocol) or IP (Internet Protocol) could be relied upon.
This method allows a large piece of information or data let it be in the form of the image of burst images forming a video to be transmitted reliably, as it manages how a larger packet is broken into smaller packets to be transmitted and again reassembled in the right order at the destination without losing the property.

In this project, stream sockets have been used because they are:

- Connection-Oriented Sockets Transmission Control Protocol (TCP).
- Packets are sequenced with a unique flow of error-free data.
- Order and reliability are guaranteed.

Many libraries in python have been used, such as:

- **socket**: To get socket module from python. It is a way of connecting two endpoints over the network to communicate with each other.
  One Socket is a combination of IP and Port number.
- **cv2**: To import Computer Vision module from python. It is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture, and analysis including features like face detection and object detection.
- **pickle:** For serializing and de-serializing python object structures.
- **struct:** To convert native Python data types such as strings and numbers into a string of bytes and vice versa.

At the Server Side:

A server has a bind() method which binds it to a specific IP and port so that it can listen to incoming requests on that IP and port. A server has a listen() method which puts the server into listening mode. This allows the server to listen to incoming connections. The close() method closes the connection with the client. We use accept() as it blocks and waits for an incoming connection. When a client connects, it returns a new socket object representing the connection and a tuple holding the address of the client. The tuple will contain (host, port) for IPv4 connections or (host, port, flowinfo, scopeid) for IPv6. Then we have to serialize frame to bytes, pack the serialized data and send this data and print the error message in case an error occurs and finally show the video frame.

At the Client Side:

A socket object is created, connects to the server and calls client_socket.recv() to read the server's reply . In the client side we receive data frames for the video streaming but in form of data packets which we have to unpack and de-serialize and then show the actual video frame.

CHAPTER 4

IMPLEMENTATION

How will the video data transmission take place?

First, we have to create two python programming files: one Client.py and another Server.py

At the server side:

- With Open CV get video frames of webcam,
- With pickle serialize frame to byte data,
- Pack each frame data using struct module,
- Send data to client and display frame.

Python Server Module:

- Socket Creation: clientsocket= socket.socket(socket.AF_INET, socket.SOCK_STREAM)
- Socket Connect: clientsocket.connect((host_ip, port))
- Socket Send: clientsocket.sendall()

At the client side:

- Receive packets and append them to data,
- Unpack the data using struct module,
- Load the frame using pickle,
- Display the frame at client side.

Python Client Module:

- Socket Creation: s= socket.socket(socket.AF_INET, socket.SOCK_STREAM)
- Socket Bind: s.bind((host_ip, port))
- Socket Accept: conn, addr = s.accept()
- Socket Listen: s.listen(10)
- Socket Receive: data += conn.recv(4096)

CHAPTER 5

RESULTS

The results obtained under testing the system are presented and discussed in this section.
Socket programming in python language is used.
The protocol used here is TCP (Transmission control Protocol) as it is well established, provides packet ordering, retransmissions, and prevents packet loss.

*C:\Users\user\Desktop\PES2UG19EC099_WNF_PROJECT 3\server.py - Notepad++

server.py

```
# CODE FOR THE SERVER
import socket, cv2, pickle,struct # Importing libraries
# Creating Socket
server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
host_name  = socket.gethostname()
host_ip = socket.gethostbyname(host_name)
print('HOST IP:',host_ip)
port = 9999
socket_address = (host_ip,port)
# Socket Bind
server_socket.bind(socket_address)
# Socket Listen
server_socket.listen(5)
print("LISTENING AT:",socket_address)
# Socket Accept
while True:
    client_socket,addr = server_socket.accept()
    print('GOT CONNECTION FROM:',addr)
    if client_socket:
        vid = cv2.VideoCapture(0)
        while(vid.isOpened()):
            img,frame = vid.read()
            a = pickle.dumps(frame)
            message = struct.pack("Q",len(a))+a
            client_socket.sendall(message)
            cv2.imshow('TRANSMITTING VIDEO',frame)
            key = cv2.waitKey(1) & 0xFF
            if key ==ord('q'):
                client_socket.close()
```

*C:\Users\user\Desktop\PES2UG19EC099_WNF_PROJECT 3\client.py - Notepad++

client.py

```
while True:
    while len(data) < payload_size:
        packet = client_socket.recv(4*1024)
        if not packet: break
        data+=packet
    packed_msg_size = data[:payload_size]
    data = data[payload_size:]
    msg_size = struct.unpack("Q",packed_msg_size)[0]
    while len(data) < msg_size:
        data += client_socket.recv(4*1024)
    frame_data = data[:msg_size]
    data  = data[msg_size:]
    frame = pickle.loads(frame_data)
    cv2.imshow("RECEIVING VIDEO",frame)
    key = cv2.waitKey(1) & 0xFF
    if key  == ord('q'):
        break
client_socket.close()
```

Code for the server.py and client.py files

The server initializes and binds the data to a specific IP and port so that it can listen to incoming requests on that IP and port.

Command Prompt - python server.py

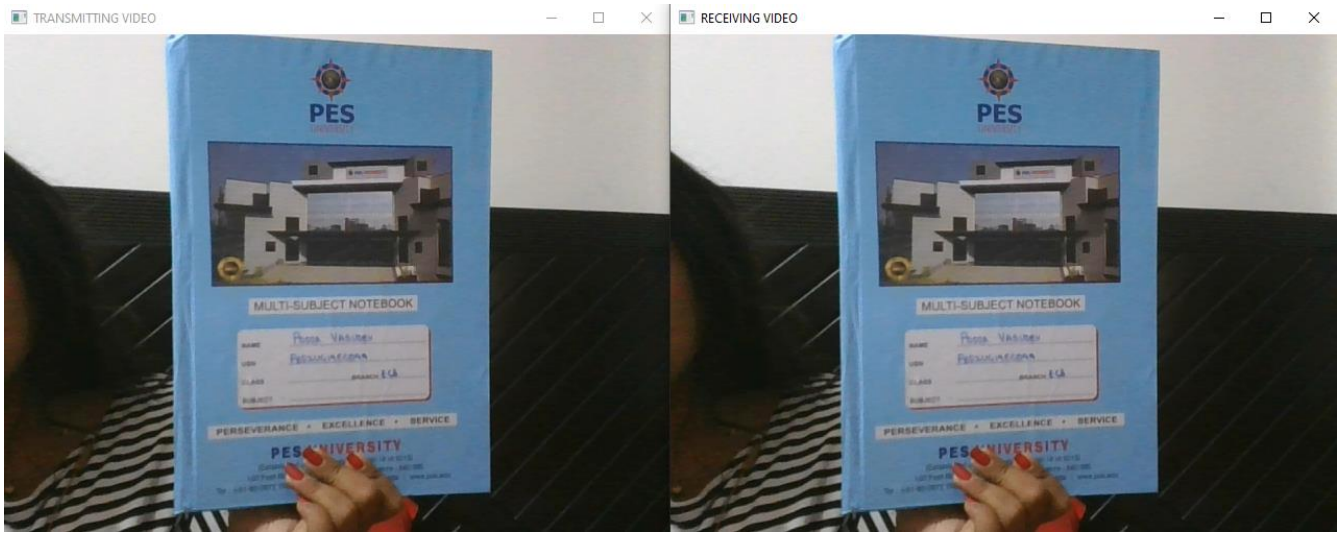
```
Microsoft Windows [Version 10.0.19043.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>cd Desktop

C:\Users\user\Desktop>cd PES2UG19EC099_WNF_PROJECT 3

C:\Users\user\Desktop\PES2UG19EC099_WNF_PROJECT 3>python server.py
HOST IP: 192.168.238.1
LISTENING AT: ('192.168.238.1', 9999)
GOT CONNECTION FROM: ('192.168.238.1', 50942)
```

Command Prompt - python client.py

```
Microsoft Windows [Version 10.0.19043.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>cd Desktop

C:\Users\user\Desktop>cd PES2UG19EC099_WNF_PROJECT 3

C:\Users\user\Desktop\PES2UG19EC099_WNF_PROJECT 3>python client.py
```

Command prompt for the server.py and client.py files (Transmitted and received signals)

One socket (node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.



Output of the video being streamed

Upon receiving a connection from the client, the transmitting and receiving window opens up. As seen the transmitting and receiving videos are in complete synchronization and there is no packet loss that occurs.

CONCLUSION

The conclusions drawn from this project is:

- TCP is used where transferring every frame/packet is important. Netflix, Youtube, etc. video streaming services all use TCP and simply buffer a few seconds of content, instead of using UDP since the delay is not crucial and TCP transfers can be easily accomplished over HTTP and web browsers without the need for additional plugins and software.
- Using socket programming in TCP guarantees the delivery of data to the destination router, thus making it reliable.
- The speed of transmission is due to reordering and retransmission.
- If there's a missing packet which causes a glitch in the video, TCP will let the packet to be retransmitted. YouTube/Netflix/Hulu etc, adjusts video quality based on network congestion, and this can be detected by TCP.
- It also performs error checking and error recovery. It offers extensive error-checking mechanisms using flow control and acknowledgment of data.

But, on the other hand:

- If the client demands a bandwidth more than the sender could provide and there is a data loss, then latency is unavoidable. Now if we think of a live video streaming, when packet loss occurs, the performance of the live streaming will be heavily deteriorated because the protocol will be busy re-transmitting the lost packets. This will result in the viewer lagging behind than what is actually being streamed on the video live.
- TCP buffers the unacknowledged segments for every client. In some cases this is undesirable, such as TCP streaming for very popular live events because the list of simultaneous clients (and buffering requirements) are large in this case. Pre-recorded video-casts typically don't have as much of a problem with this because viewers tend to stagger their replay activity.
- IP multicast significantly reduces video bandwidth requirements for large audiences; multicast requires UDP (and is incompatible with TCP).
- For real time videos, using UDP has the merit that it's faster and has lower overhead, a few packets dropped in between doesn’t matter.

REFERENCES

[1] Zebari, Ibrahim & Zeebaree, Subhi & Yasin, Hajar. (2019). Real Time Video Streaming From Multi-Source Using Client-Server for Video Distribution. 109-114. 10.1109/SICN47020.2019.9019347.

[2] Usuff, Rahamathunnisa & Ramakrishnan, Saravanan. (2013). A survey on video streaming over multimedia networks using TCP. Journal of Theoretical and Applied Information Technology. 53. 205-209.

[3] Metzger, Florian & Liotou, Eirini & Moldovan, Christian & Hossfeld, Tobias. (2016). TCP Video Streaming and Mobile Networks: Not A Love Story, But Better With Context. Computer Networks. 109. 10.1016/j.comnet.2016.06.033.

[4] Santos-González I, Rivero-García A, Molina-Gil J, Caballero-Gil P. Implementation and Analysis of Real-Time Streaming Protocols. Sensors (Basel). 2017 Apr 12;17(4):846. doi: 10.3390/s17040846. PMID: 28417949; PMCID: PMC5424723.

[5] Bing Wang, Jim Kurose, Prashant Shenoy, and Don Towsley. 2008. Multimedia streaming via TCP: An analytic performance study. ACM Trans. Multimedia Comput. Commun. Appl. 4, 2, Article 16 (May 2008), 22 pages. DOI:https://doi.org/10.1145/1352012.1352020

[6] Krasic, C., Li, K., & Walpole, J. (2001). The Case for Streaming Multimedia with TCP. *IDMS*.

[7] Website: https://pyshine.com/Socket-programming-and-openc/

[8] Website: http://www2.ic.uff.br/~michael/kr1999/6-multimedia/6_02-audioVideoOverWeb.html

DEMO VIDEO WITH EXPLANATION LINK

https://drive.google.com/file/d/1bzHGrCBngOlK_92iQT-NP-eV6xjGpPGD/view?usp=sharing