MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Mini Project Report
of
Computer Networks Lab (CSE 3162)

TITLE: "Enhancing Video Communication through Socket Programming: A UDP Datagram-Based Video Calling System"

SUBMITTED BY

Vinayak Joshi (Reg. no. 210905270) Roll. No. 43
Kaushal Singh (Reg. no. 210905404) Roll. No. 63
Kush Agarwal (Reg. no. 210905358) Roll. No. 58
Kushagra Saraf (Reg. no. 210905352) Roll. No. 56

Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal.

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal
16/10/2023

CERTIFICATE

This is to certify that the project titled **Enhancing Video Communication through Socket Programming: A UDP Datagram-Based Video Calling System** is a record of the bonafide work done by **Vinayak Joshi(Reg. No. 210905270), Kaushal Singh (Reg. no. 210905404), Kush Agarwal(Reg. no. 210905358) and Kushagra Saraf (Reg. no. 210905352)** submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in COMPUTER SCIENCE & ENGINEERING of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2023-2024.

Name and Signature of Examiners:

1. **Dr. Radhakrishna Bhat, Assistant Professor, CSE Dept.**

2. **Dr. Ujjwal, Assistant Professor, CSE Dept.**

TABLE OF CONTENTS

ABSTRACT:

In a world increasingly reliant on remote communication, the need for efficient and low-latency video calling systems has never been greater. This report delves into the realm of enhancing video communication through socket programming, with a specific focus on a UDP datagram-based video calling system. Our study explores the practical application of socket programming, particularly the UDP protocol, to revolutionize real-time video communication. The primary objective of this report is to elucidate the architecture, implementation, and significance of this technology. By understanding the core concepts, organizations and individuals can harness the potential to create more effective and efficient video communication systems. The advantages of a UDP-based system, such as low latency and scalability, are dissected, alongside practical guidance on addressing security and privacy concerns. Through real-world applications and case studies, this report showcases how businesses and industries can benefit from this innovation. As the report unfolds, it becomes evident that the utilization of socket programming in a UDP datagram-based video calling system is not just an enhancement but a necessity in the evolving landscape of communication technologies.

CHAPTER 1: INTRODUCTION

In today's digitally interconnected world, video communication has become an indispensable part of our personal and professional lives. Whether it's conducting virtual meetings, seeking medical advice through telemedicine, engaging in immersive online gaming experiences, or simply staying connected with friends and family, the demand for high-quality, real-time video calls is ever on the rise.

"Enhancing Video Communication through Socket Programming: A UDP Datagram-Based Video Calling System" represents a groundbreaking and forward-thinking approach to revolutionize the way we connect and communicate through video. At its core, this system leverages the power of User Datagram Protocol (UDP) and sophisticated socket programming techniques, offering an innovative solution to the challenges of video communication in an increasingly interconnected world. The core advantage of this system lies in its ability to provide low-latency, high-quality video communication.

By harnessing the speed and efficiency of UDP, it can transmit video data in real-time, ensuring that participants in a video call experience minimal delays and interruptions. This capability makes it ideal for applications where instant and uninterrupted interaction is paramount. In this exploration, we embark on a journey to uncover the immense potential of this pioneering technology. We will delve into the future prospects and potential applications of this innovative system, which promises to reshape the way we interact and transact in a digital world characterized by constant connectivity. Whether it's improving remote work experiences, revolutionizing healthcare delivery, enhancing the user experience in online gaming, or facilitating innovative social interactions, this UDP-based video calling system is poised to transform the landscape of video communication as we know it. Join us as we delve deeper into the promising possibilities of this cutting-edge technology and how it may soon become an integral part of our daily lives.

CHAPTER 2: PROBLEM STATEMENT & OBJECTIVES

Problem Statement:

In the context of modern communication technology, the existing video calling systems often face significant challenges, such as latency issues, security concerns, and scalability limitations. These limitations hinder the seamless exchange of visual information in various critical domains, including remote healthcare, virtual events, mobile applications, and IoT integration. To address these challenges and unlock the full potential of video communication, a novel approach is needed. The "Enhancing Video Communication through Socket Programming: A UDP Datagram-Based Video Calling System" seeks to provide an innovative solution to these problems..

Objectives:

1.Minimize Latency: The primary objective of this system is to significantly reduce latency in video communication. By utilizing the User Datagram Protocol (UDP) and advanced socket programming techniques, the system aims to deliver real-time video communication with minimal delay. This is crucial for applications where instantaneous interaction is paramount, such as telemedicine and online gaming.

2.Enhance Quality: Another key objective is to enhance the quality of video communication. The system aims to provide high-definition video with improved stability, ensuring a smoother and more immersive experience for users. This objective is particularly relevant for video conferencing and virtual events.

3.Scalability: The system will be designed with scalability in mind, allowing it to accommodate a growing number of participants in a video call. The objective is to enable seamless communication in larger virtual gatherings, making it suitable for applications such as webinars and virtual conferences.

4.Security and Privacy: Security and privacy are paramount in video communication. The system aims to incorporate robust security measures, such as end-to-end encryption, to protect the confidentiality of video calls. This objective is especially critical for applications in sectors like government, healthcare, and law enforcement.

5.IoT Integration: An essential objective is to explore the integration of the system with Internet of Things (IoT) devices. By enabling video communication with IoT devices, the system can find applications in smart homes, smart cities, and industrial settings, providing a bridge between visual information and control of connected systems. By addressing these objectives, the "Enhancing Video Communication through Socket Programming: A UDP Datagram-Based Video Calling System" aims to pave the way for a new era of efficient, high-quality, and secure video communication across various domains, ultimately redefining the way we connect and collaborate in a digital world.

By addressing these objectives, our project aspires to implement internet radio broadcasting by providing a novel, efficient, and interactive platform that more closely emulates the real-life broadcasting experience, ultimately delivering a richer and more dynamic user experience.

CHAPTER 3: METHODOLOGY

The methodology for developing and implementing the "Enhancing Video Communication through Socket Programming: A UDP Datagram-Based Video Calling System" involves a series of steps and approaches to achieve the stated objectives. Below is an outline of the methodology:

1. Client Join Request:
- Clients initiate the process by sending a join request to the server, expressing their intention to join a multicast group.

2. Server Response:
- The server promptly responds by providing clients with a comprehensive station list and essential site information using TCP.

3. Station Selection and Connection:
- Clients choose their preferred station from the list and establish a connection to that station, enabling access to multimedia content.

4. Continuous Data Broadcasting:
- All stations are configured to continuously broadcast data, mirroring real-life broadcasting practices where content is transmitted regardless of receiver connections, much like traditional TV and radio.

5. Live Streaming Activation:
- When a receiver connects to a specific station, they gain immediate access to live-streaming videos from that station.

6. Media Player Utilization:
- The system employs the "ffplay" media player. Videos on the station side are converted to a streamable format using the "ffmpeg" command, ensuring smooth streaming.
- Conversion Command: `ffmpeg -i inputfile.mp4 -f mpegts streamable_output.mp4`

them to start receiving live-streaming data from the newly selected station.

Terminate:

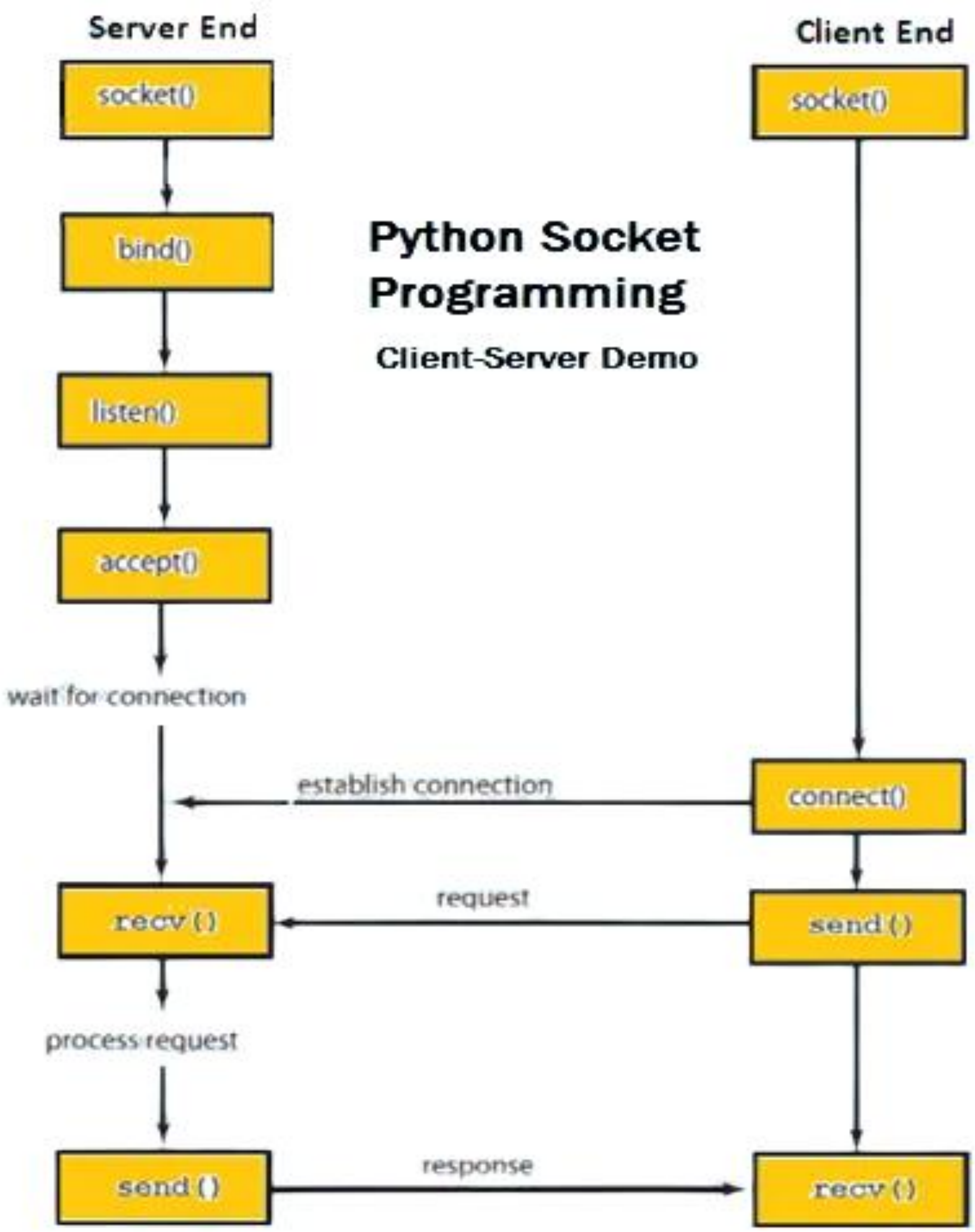
The "Terminate" function, when selected, disconnects the receiver from the station it was previously connected to. It effectively exits the station and employs the command pkill <media player> to facilitate this disconnection.

Multi-Threading:

Threads are employed to enable parallel execution of two key processes:

- Socket programming for sending and receiving data.

FLOW CHART/DESIGN OUTLINE:



CHAPTER 4: IMPLEMENTATION

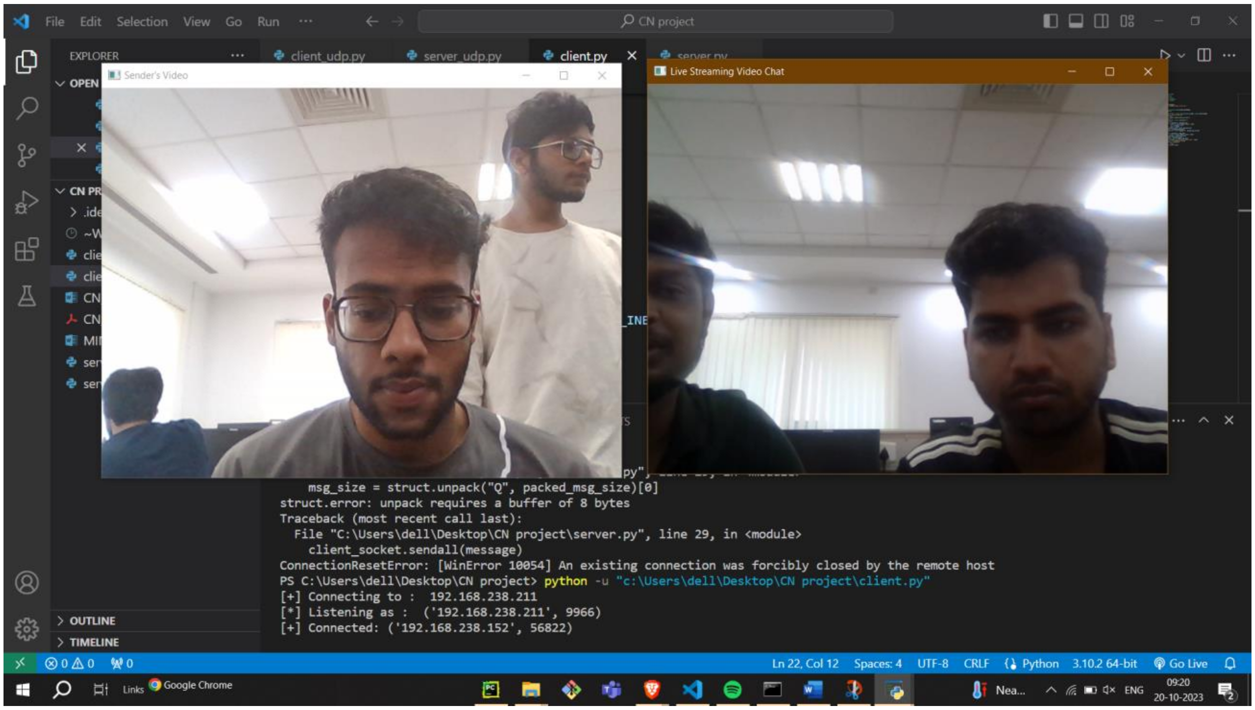
Server Code: server.py

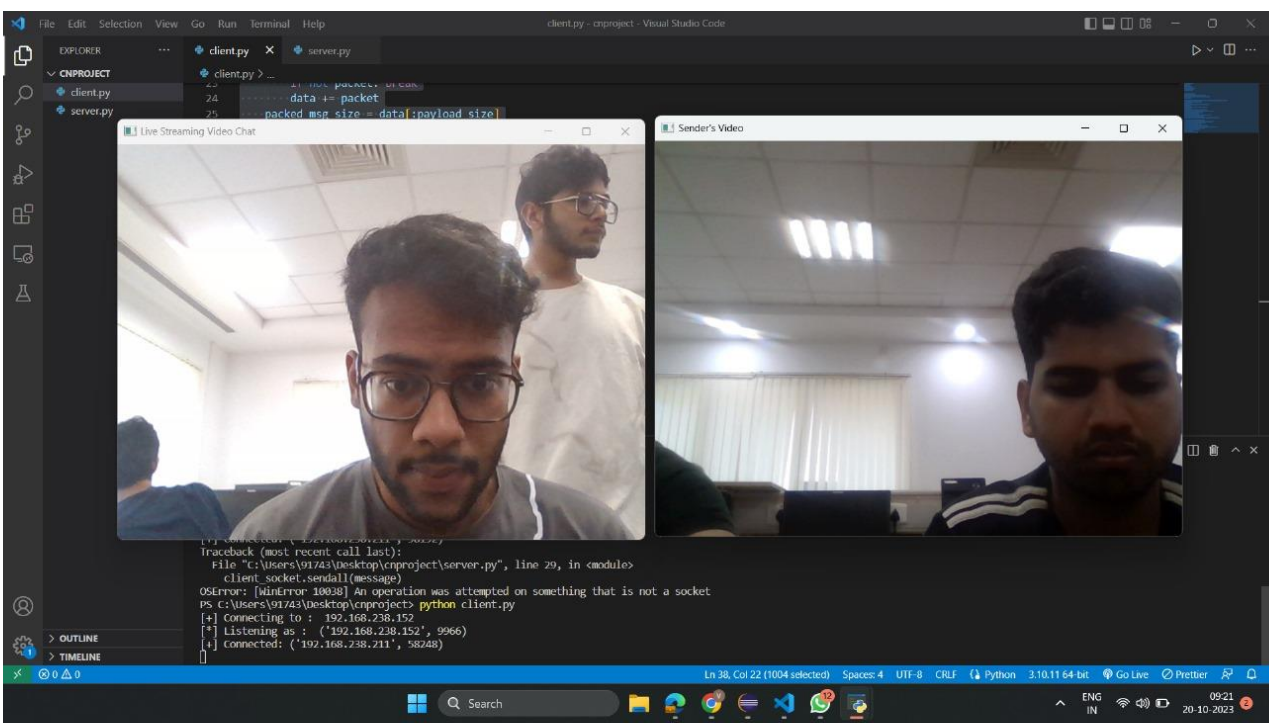
```
import socket
import cv2
import pickle
import struct

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

host_ip = '192.168.29.79'
print('[+] Connecting to : ', host_ip)
port = 9966
socket_address = (host_ip, port)

server_socket.bind(socket_address)

server_socket.listen(5)
print("[*] Listening as : ", socket_address)

while True:
    client_socket, addr = server_socket.accept()
    print('[+] Connected:', addr)
    if client_socket:
        vid = cv2.VideoCapture(0)
        vid.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
        vid.set(cv2.CAP_PROP_FRAME_HEIGHT, 500)
        while (vid.isOpened()):
            img, frame = vid.read()
            a = pickle.dumps(frame)
            message = struct.pack("Q", len(a)) + a
            client_socket.sendall(message)
            cv2.imshow("Sender's Video", frame)
            key = cv2.waitKey(1) & 0xFF
            if key == ord('q'):
                client_socket.close()
```

Client code: client.py

```
import socket
import os
import cv2
import pickle
import threading
import struct


def sendImage():
    os.system('python server.py')


t1 = threading.Thread(target=sendImage)
t1.start()

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host_ip = '192.168.29.32'
port = 9966
client_socket.connect((host_ip, port))
data = b""
payload_size = struct.calcsize("Q")
while True:
    while len(data) < payload_size:
        packet = client_socket.recv(8 * 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(8 * 1024)
    frame_data = data[:msg_size]
    data = data[msg_size:]
    frame = pickle.loads(frame_data)
    cv2.imshow("Live Streaming Video Chat", frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord('q'):
        break
client_socket.close()
```

SCREENSHOTS





CHAPTER 5: CONCLUSION

In this Computer Networking project, we set out to explore the intricacies of video streaming, aiming to provide a seamless and efficient user experience. Our project encompassed the design and implementation of a video streaming system, along with the study of network protocols, data transmission techniques, and video encoding technologies. We can draw several key conclusions from our endeavors:

Understanding Network Protocols: Through this project, we delved into the world of network protocols, specifically focusing on real-time multimedia data transmission. We gained valuable insights into the differences between TCP and UDP, each with its own advantages and trade-offs. We learned that while TCP ensures reliability through data retransmission, UDP is more suitable for low-latency, real-time streaming applications, allowing us to make informed protocol choices for our video streaming system.

Latency and Quality Trade-offs: Balancing latency and video quality is a constant challenge in video streaming. Through experimentation, we discovered that lower-latency streaming is essential for real-time applications like video conferencing, while higher-latency streaming can be acceptable for on-demand video playback services.

Quality of Service (QoS): We recognized the significance of Quality of Service parameters, such as bandwidth, jitter, and packet loss, in ensuring a smooth video streaming experience. Network monitoring tools allowed us to assess and optimize QoS parameters to enhance the reliability and quality of our video streaming system.

Scalability and Load Balancing: In a real-world scenario, video streaming platforms must handle an increasing number of users and traffic. We explored load balancing techniques to distribute incoming connections across multiple servers, ensuring scalability and redundancy.

Future Directions: Video streaming technology is ever-evolving, with emerging standards like WebRTC and low-latency streaming protocols. As a continuation of this project, we would explore these technologies and investigate adaptive streaming techniques for optimizing video quality based on network conditions.

In conclusion, our project provided us with an in-depth understanding of video streaming and its associated challenges. We believe that the knowledge and experiences gained through this project will serve as a solid foundation for future work in the field of computer networking and multimedia technologies.

CHAPTER 6: DIVISION OF WORK

Server code: Vinayak, Kushagra

Client code: Kush, Kaushal

Debugging: Vinayak, Kush

Report: Kaushal, Kushagra

CHAPTER 7: REFERENCE

The resources that have been used in the creation of this project are:

Websites:

- **https://pypi.org/project/opencv-python/**
  (open-cv , cv2 documentation)
- **https://docs.python.org/3/library/pickle.html**
  (pickle library documentation)
- **https://docs.python.org/3/library/os.html**
  (os library documentation)
- **https://www.youtube.com/**
- **https://www.geeksforgeeks.org/python-network-programming/**

We also referred to the book (Computer Networks - A Top-Down Approach (Firouz Mosharraf Behrouz A Forouzan)) to resolve queries