Computer Networks Lab Assignment 4

Video Streaming using UDP Socket Programming in Python

1. Aim

To implement a simple video streaming application using UDP sockets in Python, where the server reads and sends video frames and the client receives and displays them in real-time.

2. Requirements

- Python 3.x
- OpenCV (cv2)
- NumPy
- Basic understanding of UDP socket programming and client-server model

3. Theory

Socket Programming: Sockets provide the interface between the application layer and transport layer for network communication.

UDP (User Datagram Protocol):

- Connectionless protocol.
- No guarantee of packet delivery (unreliable, but faster than TCP).
- Useful in real-time applications like video streaming, gaming, and VoIP.

Video Streaming Approach in this Lab:

- **Server side:** Reads video, compresses frames, splits them into chunks, and sends using UDP.
- **Client side:** Receives chunks, reconstructs the frames, decodes them, and displays the video.

4. Algorithm

Server

1. Create a UDP socket.
2. Open the video file.

3. For each frame:
   (a) Resize and encode into JPEG.
   (b) Split frame into chunks of fixed size.
   (c) Send each chunk with a header (marker bit to indicate last packet of frame).
4. Sleep for frame interval (maintain FPS).
5. Repeat until video ends.

Client

1. Create a UDP socket and bind to listening port.
2. Receive packets continuously.
3. Append data until marker = 1 (last packet of frame).
4. Decode frame and display using OpenCV.
5. Stop when user presses “q”.

5. Result

The experiment successfully demonstrated a UDP-based video streaming system. The server was able to read and transmit video frames over the network, and the client received, decoded, and displayed them in real-time.

6. Conclusion

This lab illustrates how socket programming with UDP can be used for real-time video transmission. Although UDP does not guarantee packet delivery, its low latency makes it ideal for streaming applications such as online video, gaming, and live conferencing.