External Project Presentation of Computer Networking(CSE 3034)

on

Intranet based Video over Wi-Fi System

Supervisor

Sir Rahul Mondal

Presented by

Manshika Gerg – 2141007007

Jashmeet Panda - 2141016056

Tushar Prasad - 2141016321

Nandita Singh – 2141019333

Aman Kumar Singh - 2141019334

Department of Computer Science and Engineering

Institute of Technical Education & Research (FET)

Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar

Jan, 2024

Contents

Introduction

- In the ever-evolving landscape of corporate communication, the integration of advanced technologies has become imperative for enhancing connectivity and collaboration within organizations. One such groundbreaking solution that has revolutionized internal communication is the Intranet-based Video over Wi-Fi (VoWi-Fi) system. This innovative system leverages the power of local networks to seamlessly transmit video content, fostering a dynamic and interactive environment within the confines of an organization.
- The client-server webcam streaming application is designed to enable real-time video transmission from a server to multiple clients over a network. The primary objective of this project is to facilitate remote viewing of webcam-captured video feeds across connected devices.

Problem Statement

- Implement an **Intranet-based Video over Wi-Fi (VoWi-Fi) system** within a single access point using socket programming. Demonstrate how to capture video frames from a camera on the server side, transmit them over a local network using sockets, and display them on the client side. (use Java Programming)

Features:

Server-Client Architecture: Establish a server-client architecture, where the server captures video frames from a camera and sends them to the client for display.

Video Capture and Transmission: Perform video capture, processing, and transmission over a local network.

Socket Communication: Use basic socket programming to transmit video frame data between the client and server over a specified port.

Real-time Video Display: The client-side processes and displays the received video frames in real-time, demonstrating the continuous transmission of video data.

Methodology

VIDEO CLIENT PSEUDOCODE

1. Connect to the server using Socket and specific IP address and port
2. Initialize ObjectInputStream to receive data from the server
3. Create GUI for displaying the streamed content
4. Loop:
    5. Receive ImageIcon from the server using ObjectInputStream.readObject()
    6. Update GUI with the received ImageIcon
7. Close resources (Socket, etc.) when done

VIDEO CLIENT ALGORITHM

1. Connect to Server:
 - Create a Socket and connect to the server using the server's IP address and port.
2. Initialize ObjectInputStream:
    - Create an ObjectInputStream to receive data from the server.
3. Create GUI for Display:
    - Set up a graphical user interface (GUI) for displaying the streamed content.
4. Receiving Loop:
 - Enter an infinite loop.
        - Receive an ImageIcon from the server using ObjectInputStream.readObject().
        - Update the GUI with the received ImageIcon.
5. Close Resources:
            - Close the Socket and any other open resources when done receiving.

Methodology

VIDEO SERVER PSEUDOCODE

1. Initialize Webcam.
2. Open the Webcam and Set the View Size.
3. Initialize ServerSocket on a specific port.
4. Wait and Accept a client connection using ServerSocket.accept()
5. Initialize ObjectOutputStream to send data to the client.
6. Create and Show GUI for streaming (optional).
7. Loop:
   8. Capture frame from Webcam
   9. Convert frame to ImageIcon
   10. Send ImageIcon to client using ObjectOutputStream.writeObject()
   11. Flush the ObjectOutputStream
   12. Pause for a short duration (e.g., Thread.sleep()).
13. Close resources (Webcam, ServerSocket, etc.) when done.

the

VIDEO SERVER ALGORITHM

1.Initialize Webcam:
- Create an instance of the webcam object.

2.Open Webcam and Set View Size:
- Open the webcam.
- Set the desired view size for capturing frames.

3.Initialize ServerSocket:
- Create a ServerSocket object on a specified port.

4.Accept Client Connection:
- Wait for and accept a client connection using ServerSocket.accept().

5.Initialize ObjectOutputStream:
- Create an ObjectOutputStream to send data to the client.

6.Create GUI for Streaming (Optional):
- Optionally, set up a graphical user interface (GUI) for streaming the webcam content.

7.Streaming Loop:
- Enter an infinite loop.
  *Capture a frame from the webcam.
  *Convert the frame to an ImageIcon.
  *Send the ImageIcon to the client using ObjectOutputStream.writeObject().
  *Flush the ObjectOutputStream.
  *Pause for a short duration (e.g., Thread.sleep()).

8.Close Resources:
- Close the webcam, ServerSocket, and any other open resources when done streaming.

Implementation

CLIENT CODE

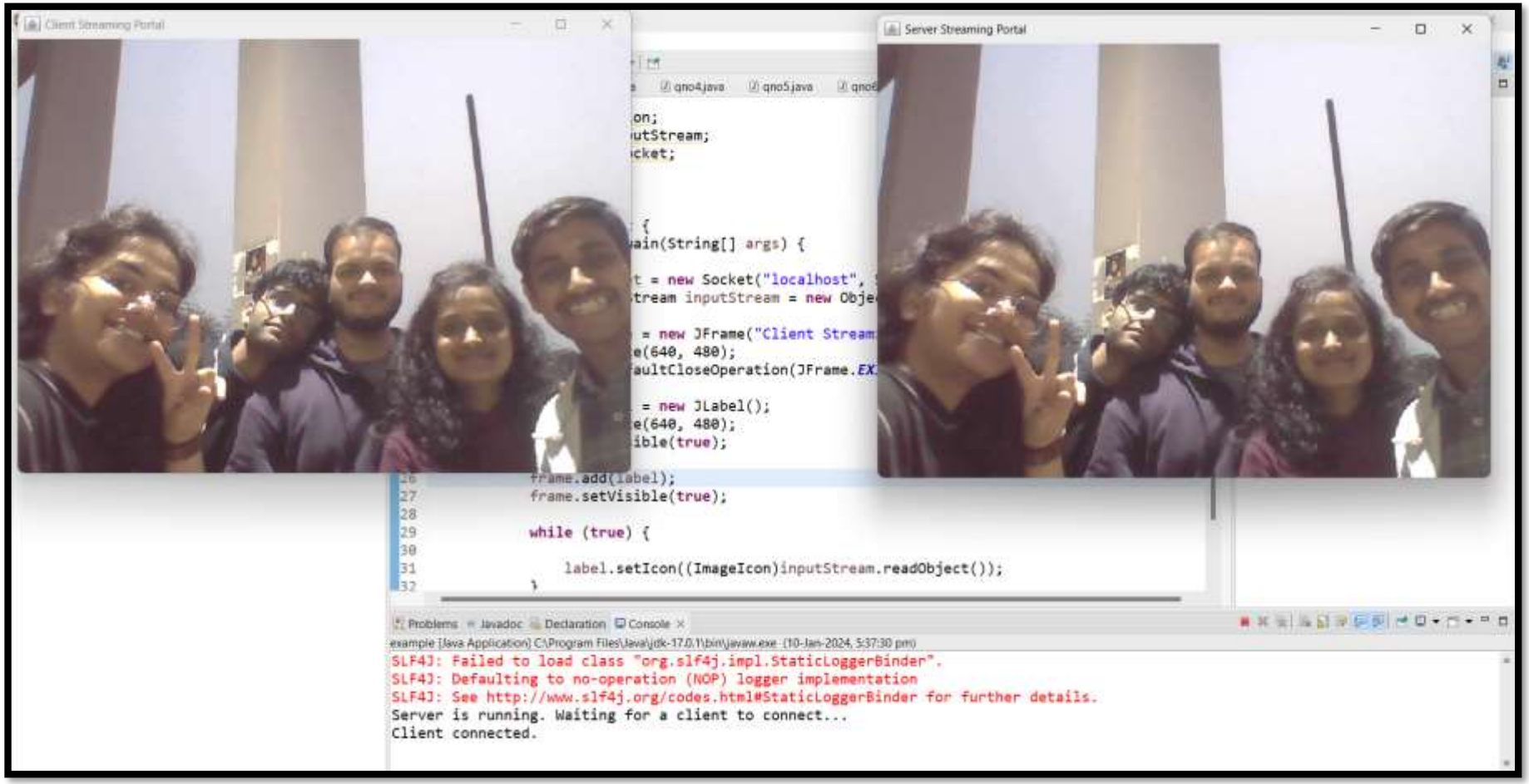
```
import javax.swing.*;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.net.Socket;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.net.ServerSocket;
import java.net.Socket;


public class client2 {
   public static void main(String[] args) {
      try {
         Socket socket = new Socket("localhost", 5555);
         ObjectInputStream inputStream = new
ObjectInputStream(socket.getInputStream());

         JFrame frame = new JFrame("Client Streaming
Portal");
         frame.setSize(640, 480);
         frame.setDefaultCloseOperation(JFrame.EXIT_ON_
CLOSE);
JLabel label = new JLabel();
         label.setSize(640, 480);
         label.setVisible(true);

         frame.add(label);
         frame.setVisible(true);

         while (true) {

          label.setIcon((ImageIcon)inputStream.readObject());
         }
      } catch (IOException | ClassNotFoundException e) {
         e.printStackTrace();
      }
   }}
```

Implementation

SERVER CODE

```
import com.github.sarxos.webcam.Webcam;

import java.io.IOException;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.awt.Dimension;
import java.awt.image.BufferedImage;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class server2 {
   public static void main(String[] args) {
      try {
         Webcam webcam = Webcam.getDefault();
         webcam.setViewSize(new Dimension(640, 480));
         webcam.open();

         ServerSocket serverSocket = new ServerSocket(5555);
         System.out.println("Server is running. Waiting for a client
to connect...");
Socket clientSocket = serverSocket.accept();
         System.out.println("Client connected.");

                              ObjectOutputStream outputStream = new
                        ObjectOutputStream(clientSocket.getOutputStream());
         JFrame frame = new JFrame("Server Streaming Portal");
         frame.setSize(640, 480);
         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

         JLabel label = new JLabel();
         label.setSize(640, 480);
         label.setVisible(true);

         frame.add(label);
         frame.setVisible(true);
         while (true) {
            BufferedImage image = webcam.getImage();
            ImageIcon imageIcon = new ImageIcon(image);
            label.setIcon(imageIcon);
            outputStream.writeObject(imageIcon);
            outputStream.flush();


            Thread.sleep(100);
         }
      } catch (IOException | InterruptedException e) {
         e.printStackTrace();
      }
   }}
```

Result and Interpretation



Sample Real-Time Video Screenshot (LEFT – Client and RIGHT – Server)

Conclusion

- The Intranet-based Video over Wi-Fi (VoWi-Fi) system is an impressive technological advancement that can help organizations communicate better. With VoWi-Fi, team members can collaborate effectively and efficiently over existing Wi-Fi infrastructure without relying on external communication platforms. This system is user-friendly and adaptable, making it easy for organizations to streamline their workflow and enhance productivity.

- VoWi-Fi is also cost-effective and provides secure and reliable video transmission. Its scalability and flexibility make it suitable for diverse organizational structures. In short, implementing VoWi-Fi can help organizations optimize internal communication and create a more connected and collaborative work environment.

- As we continue to navigate the ever-evolving landscape of digital communication, the Intranet-based VoWi-Fi system emerges as a cutting-edge solution that aligns seamlessly with the demands of modern workplaces. Successfully executing this project signifies a step forward in harnessing technology to optimize internal communication.

References

- **[1] Computer Networks, Andrew S. Tannenbaum, Pearson India.**
- [2] Java Network Programming by Harold, O'Reilly (Shroff Publishers).
- **[3] GeeksforGeeks**


Any Questions ?

Thank You