Intranet based Video over Wi-Fi system

**External Project Report on Computer Networking (CSE3034)**

**Submitted by**

|  |  |
| --- | --- |
| **Saurav Kumar** | **Reg. No.: 2141011120** |
| **Ravi Ranjan Kumar** | **Reg. No.: 2141011127** |
| **Gitanjali Kumari** | **Reg. No.: 2141011179** |
| **Vaibhav Lohitashv** | **Reg. No.: 2141012011** |
|  |  |

**B. Tech. BRANCH 5th Semester (Section B )**

**INSTITUTE OF TECHNICAL EDUCATION AND RESEARCH**

**(FACULTY OF ENGINEERING)**

i

**SIKSHA ‘O’ ANUSANDHAN (DEEMED TO BE UNIVERSITY), BHUBANESWAR,**

**ODISHA**

# Declaration

We, the undersigned students of B. Tech. of **Computer Science and Information Technology** Department hereby declare that we own the full responsibility for the information, results etc. provided in this PROJECT titled “**Intranet based Video over Wi-Fi (VoWi-Fi) system**” submitted to **Siksha ‘O’ Anusandhan (Deemed to be University), Bhubaneswar** for the partial fulfillment of the subject **Computer Networking (CSE 3034)**. We have taken care in all respect to honor the intellectual property right and have acknowledged the contribution of others for using them in academic purpose and further declare that in case of any violation of intellectual property right or copyright we, as the candidate(s), will be fully responsible for the same.

|  |  |
| --- | --- |
| **Vaibhav Lohitashv**  **Registration**  **No.: 2141012011** | **Ravi Ranjan Kumar**  **Registration No.:2141011127** |
| **Saurav Kumar**  **Registration**  **No.: 2141011120** | **Gitanjali Kumari**  **Registration No.: 2141011179** |
|  | |

**DATE:**

**PLACE:**

# Abstract

This project entails the creation of an Intranet-based Video over Wi-Fi system using Java socket programming. The core objective is to establish a server-client architecture where the server captures video frames from a camera, transmits them over a local network through basic socket communication, and the client processes and displays the frames in real-time. By engaging in this project, students will gain a solid understanding of socket programming concepts in Java, the basics of streaming video frames, and the intricacies of networking applications with Java. The project emphasizes video capture, processing, and transmission over a local network, employing socket programming to facilitate seamless communication. The client side is responsible for real-time processing and display of the video frames, showcasing continuous data transmission. This endeavor provides a comprehensive learning experience, bridging the realms of networking, video processing, and real-time data communication within a single access point, making it an invaluable project for mathematics students eager to explore practical applications in Java programming.

# Contents

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Serial**  **No.** | **Chapter No.** | | **Title of the Chapter** | | **Page No.** | |
| 1. | 1 | | Introduction | | 1 | |
| 2. | 2 | | Problem Statement | | 2 | |
| 3. | 3 | | Methodology | | 3 | |
| 4. | 4 | | Implementation | | 4 | |
| 5. | 5 | | Results and interpretation | | 8 | |
| 6. | 6 | | Conclusion | | 10 | |
| 7. |  | | References | | 10 | |
|  | |  | |  | |  | |

1. **Introduction**

The introduction of this project sets the stage for the development of an Intranet-based Video over Wi-Fi (VoWi-Fi) system utilizing Java socket programming. It provides a concise overview of the project's objectives, emphasizing the establishment of a server-client architecture. The server captures video frames from a camera, and these frames are transmitted over a local network using fundamental socket communication. The client, in turn, processes and displays these frames in real-time, showcasing continuous video data transmission. The introduction highlights the project's significance as a practical exploration of socket programming concepts in Java, streaming video fundamentals, and networking applications with a focus on server setup, client connections, and data transmission. This introductory section aims to captivate the reader's interest by presenting the key features of the project, including video capture, transmission, and real-time processing, within the context of Video over Wi-Fi architecture.

# Problem Statement

* 1. Design and implement an Intranet-based Video over Wi-Fi (VoWi-Fi) system with a focus on single access point architecture using Java socket programming. The primary challenge is to develop a comprehensive solution that seamlessly captures video frames from a camera on the server side, efficiently transmits these frames over a local network through socket communication, and successfully displays the received video data on the client side.
  2. Constraints:
* Limited Network.
* Latency Issues.
* Hardware Dependency.
* Security Complexity.
* Device Compatibility.
* Error Resilience.
* UI Optimization.
* Resource Efficiency.

# Methodology

Following is the algorithm to connect to server and stream video. First run the server code then run client code.

Server Side:

1. Initialize server socket.

2. Initialize camera for video capture.

3. Establish socket connection with the client.

4. Loop:

a. Capture video frame from the camera.

b. Convert video frame to bytes.

c. Send video frame bytes over the socket.

d. Repeat.

Client Side:

1. Initialize client socket.

2. Establish connection to the server.

3. Loop:

a. Receive video frame bytes from the socket.

b. Convert received bytes to video frame.

c. Display the video frame.

d. Repeat.

# Implementation

The implementation is divided into **two classes**, first one is server and second one is client. The server code starts first establishing a server socket in local host with a specific port mentioned. Then it waits for a connect from client side. It will wait till client is connected.

Once the client is connected with the server it opens a GUI at the client side and starts sending video frames to client at specific refresh delay mentioned in the client code and it will stop sending frames because of negative array size exception.

Note: In this project **OpenCV** library has been added to access the webcam and capture frames and covert into video. It is a large library used for handling Visual graphic files and works.

Precaution:

* Before running the code, please download the **OpenCV jar files** that contains the necessary libraries as mentioned in the code.
* Do load the **Native library.**

***Server code:***

import org.opencv.core.Core;

import org.opencv.core.Mat;

import org.opencv.core.MatOfByte;

import org.opencv.videoio.VideoCapture;

import org.opencv.imgcodecs.Imgcodecs;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.OutputStream;

import java.net.ServerSocket;

import java.net.Socket;

public class VideoServer {

    public static void main(String[] args) {

        System.loadLibrary(Core.NATIVE\_LIBRARY\_NAME);

        try {

            try (ServerSocket serverSocket = new ServerSocket(8080)) {

                System.out.println("Server is running. Waiting for a client...");

                Socket clientSocket = serverSocket.accept();

                System.out.println("Client connected.");

                VideoCapture videoCapture = new VideoCapture(0);

                if (!videoCapture.isOpened()) {

                    System.err.println("Error: Could not open camera.");

                    return;

                }

                Mat frame = new Mat();

                MatOfByte matOfByte = new MatOfByte();

                DataOutputStream dataOutputStream = new DataOutputStream(clientSocket.getOutputStream());

                while (true) {

                    videoCapture.read(frame);

                    if (!frame.empty()) {

                        Imgcodecs.imencode(".jpg", frame, matOfByte);

                        byte[] encodedFrame = matOfByte.toArray();

                        // Send the length of the frame

                        dataOutputStream.writeInt(encodedFrame.length);

                        // Send the encoded frame to the client

                        OutputStream outputStream = clientSocket.getOutputStream();

                        outputStream.write(encodedFrame);

                        matOfByte.release();

                        outputStream.flush();

                    } else {

                        System.err.println("Error: Unable to capture frame from the camera.");

                        break;

                    }

                }

                videoCapture.release();

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

***Client code:***

import org.opencv.core.Core;

import org.opencv.core.Mat;

import org.opencv.core.MatOfByte;

import org.opencv.highgui.HighGui;

import org.opencv.imgcodecs.Imgcodecs;

import java.io.DataInputStream;

import java.io.IOException;

import java.io.InputStream;

import java.net.Socket;

public class VideoClient {

    public static void main(String[] args) {

        System.loadLibrary(Core.NATIVE\_LIBRARY\_NAME);

        try {

            try (Socket socket = new Socket("localhost", 8080)) {

                Mat frame = new Mat();

                MatOfByte matOfByte = new MatOfByte();

                DataInputStream dataInputStream = new DataInputStream(socket.getInputStream());

                while (true) {

                    // Receive the length of the frame

                    int frameLength = dataInputStream.readInt();

                    // Receive the encoded frame from the server

                    InputStream inputStream = socket.getInputStream();

                    byte[] buffer = new byte[frameLength];

                    int bytesRead = inputStream.read(buffer);

                    if (bytesRead > 0) {

                        // Convert the byte array to a Mat object

                        matOfByte.fromArray(buffer);

                        frame = Imgcodecs.imdecode(matOfByte, Imgcodecs.IMREAD\_UNCHANGED);

                        // Check if the frame is not empty before displaying

                        if (!frame.empty()) {

                            HighGui.imshow("Video Client", frame);

                            HighGui.waitKey(50);

                        } else {

                            System.err.println("Warning: Received empty or corrupted frame from the server.");

                        }

                    } else {

                        System.err.println("Error: No data received from the server.");

                        break;

                    }

                }

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

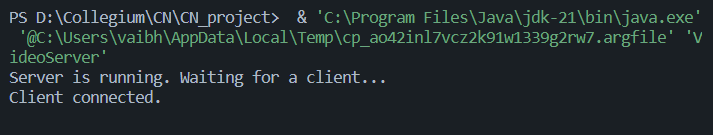
}

# Results & Interpretation

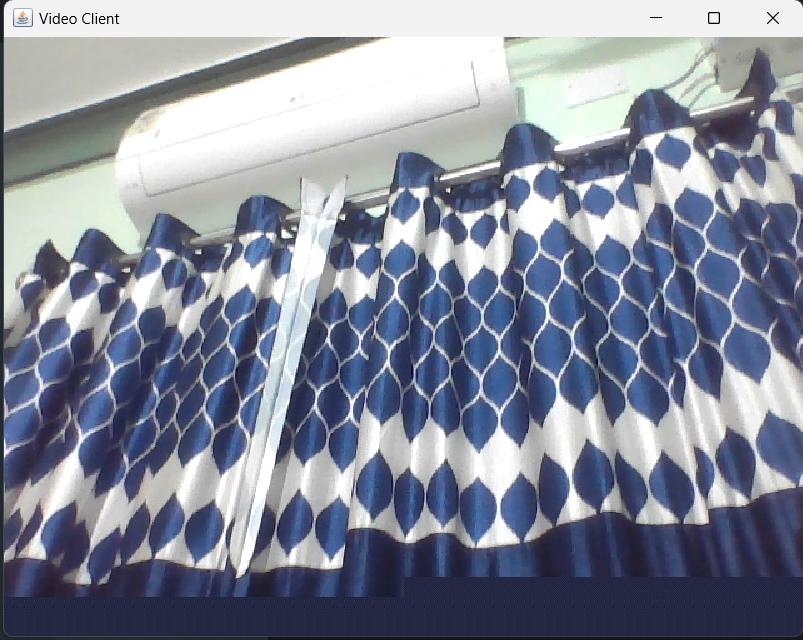
* Running server when client is not connected:



* Server is waiting for the client to connect.
* Now Client starts running:



* Client is connected to the server, Now the GUI opens and video frames starts appearing at a specific rate:



* After a certain amount of time the video freezes and client shows the following output:

Exception in thread "main" java.lang.NegativeArraySizeException:

-1283963934

* The number -1283963934 is a random number.
* And the server-side displays:

java.net.SocketException: An established connection was aborted by the software in your host machine.

* At this point the GUI starts to display corrupted frames which indicates it is time to close the GUI as it cannot capture any more frames.

# Conclusion

* To create an intranet server client video over WI-FI we need to connect using socket connection.
* To capture video frames, we need OpenCV libraries.
* Also, to create the GUI we need OpenCV libraries as it maintains a uniformity.

**References**

(as per the IEEE recommendations)

**[1] Computer Networks, Andrew S. Tannenbaum, Pearson India.**

[2] Java Network Programming by Harold, O’Reilly (Shroff Publishers).

**[3] OpenCV documentations**

**\*\*\*\*\***