Building Point to Point Video Call Application for Local Network as Socket Programming Implementation

Talitha Widya Sadina[1], Martono Dwi Atmadja[2], Muhammad Nanak Zakaria[3*], Hudiono Hudiono[4]

1,2,3,4 Digital Telecommunication Network Study Program, Department of Electrical Engineering, State Polytechnic of Malang, 65141, Indonesia.

[1]dinatalitha09@gmail.com, [2]martonoatmadja@polinema.ac.id, [3]nanak_zach@polinema.ac.id, [4]hudiono@polinema.ac.id

***Abstract*— Video call is a technology to communicate by displaying video and sound. Video calls require a network to communicate, most video call applications use cellular networks for their communication media. However, cellular networks require a server. In order for the application to be used without using a cellular network, the application creation requires a socket. Socket is an API (Application Programming Interface) used by applications to connect with each other. While socket programming is a way to use the API. Other than that, security is also needed in the video call application. Encryption is used on data, especially on the data while communicating with video calls where a communication on a video call can only be listened by the party who is communicating. Therefore, a video call application is made as a form of socket programming implementation so that the video call application does not require a cellular network for the communication process. The range of this video call application is unlimited as long as it is connected to the access point. In additional to its reach, this application is secured in the form of encryption so that when communicating later its security is guaranteed.**

***Keywords*— AES, Encryption, Python, Socket, Socket Programming, Video Call.**

I. INTRODUCTION

Nowadays, technological development is progressing rapidly across various fields, particularly in telecommunications, which plays a crucial role in enabling human interaction over distance. One of the most commonly used forms of telecommunication is voice communication due to its practicality and efficiency. However, communication becomes more effective and meaningful when voice transmission is combined with visual interaction, allowing users to observe facial expressions and gestures during conversations [1]. This need has led to the development of communication technologies capable of transmitting both audio and video data simultaneously, commonly referred to as video call communication.

Video call technology enables real-time interaction between individuals and provides significant advantages in various scenarios. One important application is for individuals with speech impairments, where video calls allow communication through sign language, which cannot be effectively supported by voice-only communication systems [2]. In addition, video call applications are widely used in educational environments, such as enabling students to consult with lecturers or supervisors who are located in different buildings or campuses without requiring physical meetings. This flexibility makes video call systems an essential communication tool in modern academic and professional activities.

From a technical perspective, video call communication requires reliable network infrastructure to support the simultaneous transmission of audio and video streams. Most existing video call applications rely on cellular networks or internet-based infrastructures as their primary communication medium. While cellular networks provide wide coverage, they typically depend on centralized servers and internet connectivity, which may introduce additional latency, higher operational costs, and dependency on external service providers. Moreover, in certain environments such as campuses, offices, or laboratories, cellular signal quality may be inconsistent or unnecessary when a stable local network is already available.

To address these limitations, the use of a Local Area Network (LAN) as a communication medium becomes a viable alternative. LAN-based communication allows devices to exchange data directly through an access point without requiring internet-based servers, thereby maximizing the utilization of existing local network resources [3]. By leveraging a LAN environment, video call applications can achieve lower latency and more controlled network conditions, which are advantageous for real-time multimedia communication.

In order to enable direct communication over a LAN without relying on cellular infrastructure, socket technology is required [4]. A socket can be described as a communication endpoint that facilitates bidirectional data exchange between applications. Technically, a socket is implemented as an Application Programming Interface (API) that allows programs to establish network connections and transmit data using specific communication protocols [5]. Socket programming refers to the method of implementing this API to

*Corresponding author

manage communication processes between client and server applications, including connection establishment, data transmission, and session termination [6].

Beyond functional communication, security is a critical requirement for video call applications. Audio and video data transmitted over a network may contain sensitive personal information and should only be accessible to authorized parties involved in the communication session [7]. Without adequate protection mechanisms, multimedia data streams are vulnerable to interception, eavesdropping, and unauthorized access. Therefore, encryption techniques are essential to ensure data confidentiality and protect user privacy during video call communication.

In this research, encryption is applied to audio and video data using the Advanced Encryption Standard (AES) algorithm, which is widely recognized for its efficiency and security in protecting digital data [8]. By encrypting multimedia streams before transmission, the original data becomes unreadable to unauthorized parties, ensuring that only intended recipients can decrypt and access the communicated information. The integration of encryption mechanisms enhances the overall security of the video call system without altering its core communication functionality.

Several previous studies related to socket programming and multimedia communication are used as references in this research. Prior work such as "Implementation of Socket TCP/IP to Send and Insert Text Files into Databases" discusses the use of socket communication in Java-based desktop applications for transmitting data over TCP/IP networks [9]. Another study, "Designing Voice Chat Applications with Socket Programming on Android for Local Networks," presents the development and evaluation of voice communication systems operating within LAN environments [10]. Research conducted by Egzon Salihu and Gentiana Blakaj in "Workplace Chat Application Using Socket Programming in Python" focuses on group-based communication applications implemented over local networks [11]. Additional studies, including "Realtime Chat Application using Client–Server" and "Development of Voice Call Transfer Service between Android," explore real-time communication systems using socket programming and client–server architectures [12], [13]. Furthermore, the study "Multi-User Chat Application using Client Server Architecture" highlights the implementation of LAN-based chat systems using Python and desktop platforms [14].

Although these studies demonstrate the feasibility of socket-based communication systems, most of them focus on text or voice communication and provide limited discussion on real-time video transmission combined with security mechanisms. Moreover, comprehensive evaluations that analyze both network performance and data security aspects in LAN-based video call applications remain limited. Therefore, this research aims to design and develop a desktop-based video call application that operates over a local network using socket programming. The system is evaluated based on Quality of Service (QoS) parameters, including throughput, delay, packet loss, and jitter, to assess network performance, while encryption is applied to ensure secure audio and video communication. The results of this research are expected to contribute to the development of efficient and secure real-time video communication systems in local network environments [15].

II. METHOD

A. Research Stages

The research on building a point-to-point video call application for a local network as an implementation of socket programming is conducted through several structured stages. The first stage is a literature review, which involves studying previous journals and research related to socket programming, real-time communication systems, and video or voice call applications over local networks. This stage aims to identify appropriate system architectures, communication models, and implementation approaches that have been applied in earlier studies, particularly those focusing on socket-based communication and client–server systems [9]–[14]. In addition, this stage determines the programming language and development environment used in this research by considering their suitability for real-time multimedia transmission.

After the literature review stage, the research proceeds to the system design stage. In this stage, the overall design of the video call application is developed based on the findings from previous studies on socket programming and local network communication systems [10]–[12]. The system design defines the client–server architecture, communication flow, and data exchange mechanisms for audio and video transmission using socket programming. This design stage ensures that the application can support bidirectional communication in real time over a Local Area Network.

The next stage is system implementation, where the system design is realized into a functional application. At this stage, socket programming is implemented to enable communication between the client and server applications, following approaches commonly used in previous socket-based communication systems [11], [12], [14]. The implementation includes the development of modules for audio and video capture, transmission, reception, and display, ensuring that both sides of the communication can simultaneously send and receive multimedia data.

Once the implementation stage is completed, system testing is carried out to evaluate the functionality of the application. The testing stage verifies whether the video call application can establish connections successfully, transmit audio and video data correctly, and maintain stable communication between the client and server. If errors or system malfunctions are identified, debugging and system adjustments are performed until the application operates properly, as suggested in previous real-time communication system studies [12], [13].

After the application functions correctly, the research continues with the data collection stage. In this stage, performance data is collected during video call sessions to support Quality of Service analysis. The collected data includes parameters such as throughput, delay, packet loss, and jitter, which are commonly used to evaluate the performance of real-time multimedia communication systems [15]. The collected

data is then analyzed to assess the network performance and reliability of the proposed video call application.

The final stage of the research is the conclusion and suggestion stage. At this stage, conclusions are drawn based on the analysis results to answer the research objectives and problem formulation. In addition, suggestions are provided for future research and further development of video call applications using socket programming, particularly in terms of improving performance, scalability, or security based on the findings of this study.

B. Block Diagrams

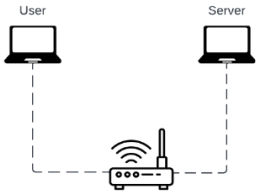

Figure 1. Block Diagram

The design that will be made to facilitate system design requires a system block diagram, in this study shown in Figure 1 which is a block diagram of the design of how the video call application work. The system design requires a block diagram to clearly illustrate the overall working mechanism of the proposed video call application. Figure 1 presents the block diagram of the system, which describes the interaction between the client and server in establishing point-to-point video call communication over a local network. In this diagram, both the client and server are equipped with input devices for capturing audio and video data, which are then transmitted through socket-based communication. The transmitted data is processed and displayed on the receiving side, enabling real-time bidirectional communication. This block diagram serves as a conceptual representation of the system architecture and provides a clear overview of the data flow and communication process implemented in the video call application.

C. Server Application Flowchart

Before presenting the detailed research procedure, it is necessary to explain how the system is implemented step by step based on the designed architecture. After defining the overall system structure through the block diagram in Figure 1, the next stage focuses on describing the sequential processes involved in developing, implementing, and evaluating the proposed video call application. To ensure that each research stage is conducted systematically and in a structured manner, a research process flowchart is used to illustrate the workflow of the study, starting from the literature review stage to the conclusion stage.

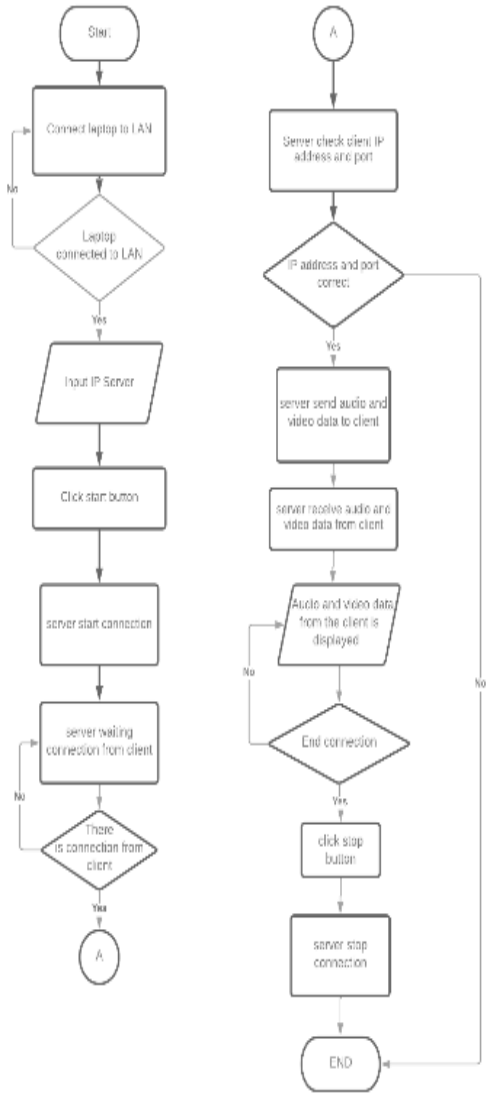

Figure 2. Server Application Flowchart

Figure 2 illustrates the workflow of the video call application on the client side, which is executed on the second laptop within the local network environment. The process begins when the client application is launched and the user inputs the server IP address to establish a connection. After the connection request is sent, the client waits for a response from the server to confirm that the communication channel has been successfully established.

Once the connection is confirmed, the client application activates the local webcam and microphone to capture video and audio data. The captured data is then encoded and transmitted to the server using socket-based communication. At the same time, the client application receives incoming audio and video data from the server, decodes the data, and displays it on the client interface in real time. This simultaneous sending and receiving process enables bidirectional communication between the client and server.

If the connection is terminated or an error occurs during the communication process, the application stops the data transmission and returns to the initial state, allowing the user to

reconnect. The flowchart in Figure 2 demonstrates that the client-side application is capable of managing connection control, multimedia data transmission, and real-time display within a point-to-point video call system.

D. Client Application Flowchart

The second diagram shows the flow of how the applicatiion work at the client application side which is run on the second laptop Figure 2 shows the flowchart. The following is a description of Figure 3.

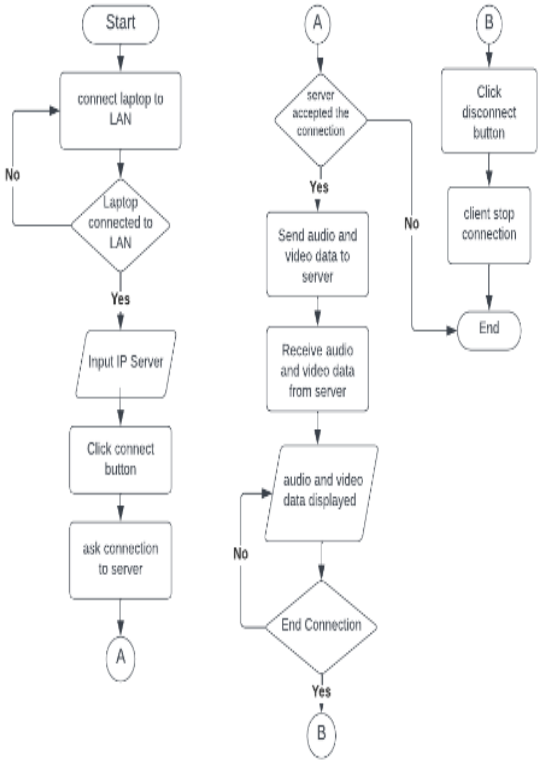

Figure 3. Client Application Flowchart

Figure 3 illustrates the workflow of the video call application on the server side within the local network environment. The process begins when the server application is launched and enters a listening state, waiting for incoming connection requests from the client application. Once a connection request is received, the server verifies and accepts the connection, establishing a communication channel using socket programming.

After the connection is established, the server activates its local webcam and microphone to capture audio and video data. The captured multimedia data is then encoded and transmitted to the client through the established socket connection. Simultaneously, the server receives incoming audio and video data sent by the client, decodes the received data, and displays it in real time on the server interface. This simultaneous transmission and reception process ensures bidirectional communication between the server and client during the video call session.

If the client disconnects or an error occurs during the communication process, the server terminates the session and returns to the initial listening state, ready to accept new connection requests. The flowchart in Figure 3 demonstrates that the server-side application is responsible for managing connection control, handling multimedia data exchange, and maintaining continuous communication during the video call operation.

E. Testing Plan

Figure 4. Testing Plan Diagram

Testing of this application will later use two laptops where the first laptop will run the server application and the second laptop will run the client application. On laptop A and laptop B, the IP will be checked which is obtained by running the command "ipconfig" at the command prompt (CMD). After checking the IP, the IP is recorded for the purposes of running the application, as shown at Figure 4.

QoS testing is done using the wireshark application by checking the data sent and received to each ip which is checked at the command prompt and the specified port. The list of ports that have been determined can be seen in the Table I.

TABLE I
PORT LIST

Description	Server port	Client port
Send Video	7000	52041
Receive Video	5000	52043
Send Audio	8000	56070
Receive Audio	6000	56072

Testing encryption data is done by printing the data so that it can be seen in the terminal when running the code. The code will be run in the pycharm application so that the data can be seen.

To evaluate the effectiveness of the implemented hardening methods, a structured security testing scenario was designed consisting of two phases: testing before hardening and testing after hardening. This comparative approach is commonly used to assess the impact of security mechanisms on system resilience [4], [13]. The evaluation includes three main stages, namely Information Gathering, Vulnerability Scanning, and cyberattack simulation.

The Information Gathering stage aims to identify publicly accessible information related to the server, such as domain configuration, exposed services, and technology stacks, using OSINT-based techniques including Whois lookup and DNS enumeration [10]. Vulnerability Scanning is performed to detect potential weaknesses in firewall configuration, open ports, and web service protection mechanisms using tools such as Nmap, SmokeWAF, and WafW00f, which are widely used for assessing network and web security [11], [14].

The cyberattack simulation stage represents common threats targeting MQTT broker servers. Denial of Service and Distributed Denial of Service attacks are conducted to evaluate broker availability and service continuity under traffic overload conditions, as previously studied in MQTT-focused DoS attack research [5]. Brute-force attacks are used to assess authentication robustness, while Reverse Shell attacks are performed to evaluate access control and privilege restriction at the operating system level [12], [15].

III. RESULTS AND DISCUSSION

A. Application Result

There are two applications: server application and client application. The server application is divided into two main pages, namely the home page and the video page. The client application is also divided into two main pages, namely the start page and the video page.

In the Figure 5 is the home page of the server application which functions to enter the server ip and start the server and activate the server application.

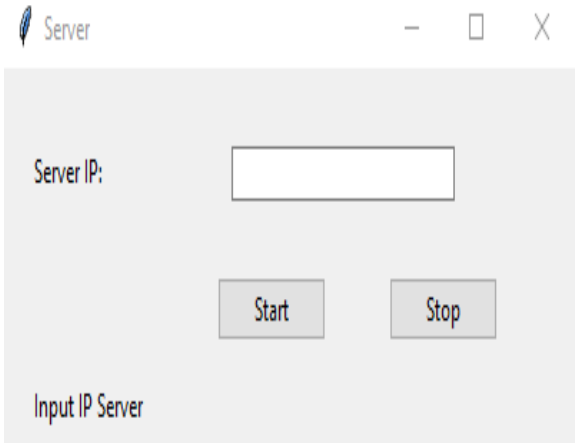

Figure 5. Server Application Main page

Figure 6. Server Application Video Page

Figure 6 is a video page. This page shows two videos, namely videos obtained from the client application and videos from the server application.

Figure 7 is the main page of the application which functions to enter the server IP and start the server and activate the client application.

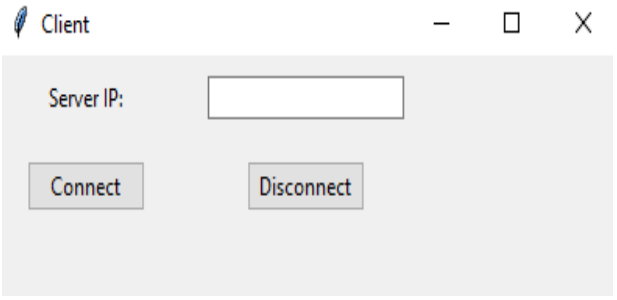

Figure 7. Client Application Main Page

Figure 8 is a video page. On this page, two videos are shown, namely the video obtained from the server application and the video from the client application.

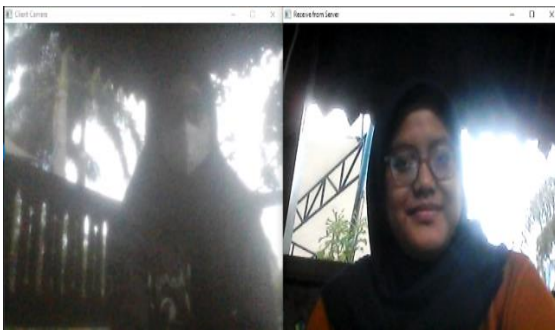

Figure 8. Client Application Video Page

Figure 6 and Figure 8 present the video call interface from two different perspectives, namely the server side and the client side of the application. Figure 6 shows the video page of the server application, where two video streams are displayed simultaneously. The first video stream represents the live video captured from the server device's webcam, while the second video stream displays the video received from the client application. This interface confirms that the server is capable of receiving, decoding, and displaying video data transmitted by the client in real time.

In contrast, Figure 8 illustrates the video page of the client application, which also displays two video streams. On this page, the first video stream shows the live video captured from the client device's webcam, and the second video stream displays the video received from the server application. The successful display of both video streams on the client side indicates that bidirectional video communication between server and client has been established properly.

The comparison between Figure 6 and Figure 8 demonstrates that the video call application operates symmetrically on both ends of the communication. Each side is able to transmit local video data while simultaneously receiving and displaying video data from the opposite side.

This confirms that the socket programming implementation supports real-time, point-to-point video communication over a local network, ensuring that both server and client applications function correctly and consistently during the video call session.

B. Quality of System (QoS) Result

The QoS test results of video delivery from server to client and from client to server are presented in Table II. The Table shows the value of throughput, packet loss, delay and jitter in the video delivery process.

In the process of sending video from server to client, the throughput value obtained reaches 100%, which according to TIPHON standardization shows index 4 or can be called a very good value. Packet loss obtained shows 0% where no packets are lost and according to TIPHON standardization this Figure shows an index value of 4 or can be called a very good value. The delay value obtained in the process of sending video from the server to the client shows 0.000376 and in the TIPHON standardization.

TABLE II
SENDING VIDEO QUALITY OF SYSTEM

	Server (192.168.0.105)	**Client (192.168.0.103)**
Throughput	100%	100%
Packet Loss	0%	0%
Delay	0,000376	0,001038
Jitter	0,000376	0,002912

In the process of sending video from server to client, the throughput value obtained reaches 100%, which according to TIPHON standardization shows index 4 or can be called a very good value. Packet loss obtained shows 0% where no packets are lost and according to TIPHON standardization this Figure shows an index value of 4 or can be called a very good value. The delay value obtained in the process of sending video from the server to the client shows 0.000376 and in the TIPHON standardization.

The QoS test results of video reception from server to client and from client to server are presented in Table III. The Table shows the value of throughput, packet loss, delay and jitter in the video reception process.

In the process of receiving video from server to client, the throughput value reached 100%, which according to TIPHON standardization shows index 4 or can be called a very good value. Packet loss obtained shows 0% where no packets are lost and according to TIPHON standardization this Figure shows an index value of 4 or can be called a very good value.

The delay value obtained in the process of sending video from server to client shows a number 1.34025 and in the TIPHON standardization for the delay value gets an index number 4, which is very good. For the jitter value, a value of 2.6805 is obtained, which in the TIPHON standardization is included in index 3, which is good.

While in the process of receiving video from client to server, the throughput value obtained is 100%, which according to the QoS standardization by TIPHON, this value gets index number 4 or very good value. The packet loss value shows 0%, which means that no packets are lost and in the TIPHON standardization the value gets an index number 4 which means very good. The delay value obtained is 0.000766 where in the TIPHON standardization this number gets an index value of 4, which is a very good value. For the jitter value, a value of 0.000763 is obtained, which in the TIPHON standardization is included in index 3, which is good.

TABLE III
RECEIVE VIDEO QUALITY OF SYSTEM

	Server (192.168.0.105)	**Client (192.168.0.103)**
Throughput	100%	100%
Packet Loss	0%	0%
Delay	1,34025	0,000766
Jitter	2,6805	0,000763

The QoS test results of sending audio from server to client and from client to server are presented in Table IV. The Table shows the value of throughput, packet loss, delay and jitter in the video reception process.

In the process of sending audio from server to client, the throughput value obtained reaches 100%, which according to TIPHON standardization shows index 4 or can be called a very good value. Packet loss obtained shows 0% where no packets are lost and according to TIPHON standardization this Figure shows an index value of 4 or can be called a very good value. The delay value obtained in the process of sending video from server to client shows 0.008061 and in the TIPHON standardization for the delay value gets an index number 4, which is very good. For the jitter value, a value of 0.022506 is obtained, which in the TIPHON standardization is included in index 3, which is good.

While in the process of sending audio from client to server the throughput value obtained is 100% where according to the QoS standardization by TIPHON this value gets index number 4 or very good value. The packet loss value shows 0%, which means that no packets are lost and in the TIPHON standardization the value gets an index number 4 which means very good. The delay value obtained is 0.008875 where in the TIPHON standardization this Figure gets an index value of 4, which is a very good value. For the jitter value, a value of 0.00838 is obtained, which in the TIPHON standardization is included in index 3, which is good.

TABLE IV
SEND AUDIO QUALITY OF SYSTEM

	Server (192.168.0.105)	**Client (192.168.0.103)**
Throughput	100%	100%
Packet Loss	0%	0%
Delay	0,008061	0,008875
Jitter	0,022506	0,00838

The results of QoS testing of receiving audio from server to client and from client to server are presented in Table V. The Table shows the value of throughput, packet loss, delay and jitter in the video reception process.

In the process of receiving audio from server to client, the throughput value obtained reaches 100%, which according to

TIPHON standardization shows index 4 or can be called a very good value. Packet loss obtained shows 0% where no packets are lost and according to TIPHON standardization this Figure shows an index value of 4 or can be called a very good value. The delay value obtained in the process of sending video from server to client shows 0.009187 and in the TIPHON standardization for the delay value gets an index number 4, which is very good. For the jitter value, a value of 0.025165 is obtained, which in the TIPHON standardization is included in index 3, which is good.

While in the process of receiving audio from client to server, the throughput value obtained is 100%, which according to the QoS standardization by TIPHON, this value gets index number 4 or very good value. The packet loss value shows 0%, which means that no packets are lost and in the TIPHON standardization the value gets an index number 4 which means very good. The delay value obtained is 0.008035 where in the TIPHON standardization this Figure gets an index value of 4, which is a very good value. For the jitter value, a value of 0.007358 is obtained, which in the TIPHON standardization is included in index 3, which is good.

TABLE V
RECEIVE AUDIO QUALITY OF SYSTEM

	Server (192.168.0.105)	Client (192.168.0.103)
Throughput	100%	100%
Packet Loss	0%	0%
Delay	0,00918726	0,008035
Jitter	0,0025165	0,007358

The results of testing data from server to client and from client to server are presented in Table VI. The Table shows what the data looks like if the data is encrypted and what if the data is not encrypted. In the Table, the video data is successfully encrypted as evidenced by the video data that is seen to be in the form of b'\xc5\x10\x0e\x00 data. The data is the result of the frame of the encrypted video. The form of the data is a series of bytes in hexadecimal form where each pair of hexadecimal characters represents one byte. So the data is a representation of four bytes in hexadecimal form.

TABLE VI
ENCRYPTION DATA

Data Type	Data Form
Video data from server to client	b'\xc5\x10\x0e\x00' b'\xc5\x10\x0e\x00' b'\xc5\x10\x0e\x00' b'\xc5\x10\x0e\x00' b'\xc5\x10\x0e\x00' b'\xc5\x10\x0e\x00' b'\xc5\x10\x0e\x00'
Video data from client to server	b'\xc5\x10\x0e\x00' b'\xc5\x10\x0e\x00' b'\xc5\x10\x0e\x00' b'\xc5\x10\x0e\x00' b'\xc5\x10\x0e\x00' b'\xc5\x10\x0e\x00' b'\xc5\x10\x0e\x00'
Audio data from server to client	(\x09\x1e\x80o\x00@\x80u\x08,\x00D\x08a\x08K\x00p\x08 \x08\x22\x08M\x00v\x08D\x00>\x00\x91\x00\xa4P\x00c\x08r\x00S\x08\x05\x00\xec\ \xff\x04\x08\x84\x08\x03\x08\xaf\x08\x09\x08ö\x00V\x08C\x08\x1c\x08\xd9\xff\xcd\xff\xe4\xff\xfe\xff\xf8\xff\xf6\xff\n\x08S\x00Z\x0 \x0f\x00\xf1\xff\xac\xff\x02\xff?\x08V\x08@\x08\xad\xff\xab\xff\xd9\xff\xc3\xff\xe8\xff\xd7\xff\xcd\xff\x11\x08\xf4\xff\xa7\xffo \xff\xff\xea\xff\x0e\xff\xd2\xff\x18\x0c:\x08\x07\x08\x22\x08\x1b\x08\n\x08\xfd\xff\xdc\xff\xa0\xff\x0a\xff\x13\x09\x08\n\x08 \x0b\x08\x0e\x08\xe0\xff\xc9\xff\xeb\xff\x04\x08\x0e\x08\xeb\xff\x08\x08A\x08\t\x08W\xff8\xff\x07\x08\xe5\xff' \xff+\xff\x1b\xffh\xff\x89\xff\x8b\xff\x9a\xff\xcd\xff\x11\x08\xf7\xff\xd1\xff\xff\xff\x19\x08\r\x08\x02\x08\xfc\xff\; \x0f\x08\xea\xff\x1e\x08O4\x08\x0b\x08\x14\x08\x03\x08\xd7\xff*\x08}\x08\x08\x1b\x08(\x08\x14\x08\x11\x08\x12\x08\xe2' \x08"\x08\xfa\xff\x1d\x08\xd5\xff1\x08e\x08p\x08\x08\x15\x08M\x08\x0me\x08W\x08\xfb\xff\xfe\xff5\x08&\x08<\; \x08\x1a\x08+\x08'
Audio data from client to server	

C. Discussion

he results obtained in this research demonstrate that the proposed point-to-point video call application based on socket programming is able to operate effectively over a local network environment. From the system implementation perspective, the block diagram and flowcharts presented earlier confirm that both client and server applications can establish a direct connection, manage session control, and perform simultaneous transmission and reception of audio and video data. The successful display of local and remote video streams on both sides indicates that bidirectional communication is achieved as designed.

Based on the application testing results, the client and server workflows operate consistently with the designed flowcharts. The client is able to initiate a connection by providing the server IP address, while the server successfully listens and responds to incoming requests. Once the connection is established, both sides can capture multimedia data from local devices, transmit it through socket communication, and display received data in real time. This confirms that socket programming provides a reliable mechanism for point-to-point multimedia communication without relying on cellular networks or external servers.

The Quality of Service (QoS) analysis further supports the effectiveness of the proposed system. The measurement results show that throughput reaches 100% and packet loss remains at 0% for both audio and video transmission. According to the TIPHON standard, these values fall into the “very good” category, indicating that the local network environment provides stable data delivery during video call sessions. This result confirms that LAN-based communication can support real-time multimedia transmission with minimal performance degradation when network conditions are controlled.

In terms of delay and jitter, the results show that audio transmission experiences low delay and acceptable jitter values, which ensures that voice communication remains clear and synchronized. Video transmission shows slightly higher delay and jitter on certain communication directions, particularly during video reception, which can be attributed to factors such as frame decoding processes, buffering mechanisms, and differences in device processing capabilities. Nevertheless, the measured delay and jitter values still fall within acceptable limits based on the TIPHON classification, indicating that the

communication quality remains suitable for real-time video calls.

From the security aspect, the implementation of encryption on audio and video data significantly enhances communication confidentiality. The results presented in the encryption testing table show that transmitted data appears in encrypted byte form, making it unreadable without proper decryption. This confirms that the applied encryption mechanism successfully protects multimedia data from unauthorized access during transmission. The combination of socket programming and encryption ensures that the proposed video call application not only performs well but also maintains data security.

Overall, the discussion of results confirms that the proposed point-to-point video call application is capable of delivering stable, real-time, and secure multimedia communication over a local network. The integration of socket programming enables efficient client–server communication, QoS evaluation verifies acceptable network performance, and encryption mechanisms ensure data confidentiality. These findings indicate that the developed system is suitable for use in environments such as campuses, offices, or laboratories where reliable local network infrastructure is available.

IV. CONCLUSION

Through a series of testing processes and data collection, it can be concluded that the development of a video call application utilizing a local network is feasible through the implementation of socket programming, which enables direct communication between devices via network sockets. The application operates using a client–server architecture, where the server creates a socket, binds it to a specific IP address and port, and listens for incoming connection requests to process data and send responses, while the client socket establishes a connection to the server to facilitate bidirectional data exchange. In terms of performance, the application achieves a Quality of Service (QoS) index value of 4 (very good) for throughput, packet loss, and delay based on TIPHON standardization, while jitter in both audio and video transmission achieves an index value of 3 (good). From the security perspective, the implementation of encryption is evidenced by the video data being transmitted in a constant encrypted byte format, such as b'\xc5\x10\x0e\x00, which is distinctly different from the audio data that appears as structured hexadecimal raw data. These results indicate that the proposed application is capable of providing reliable performance and adequate data security for real-time video call communication over a local network.

REFERENCES

[1] S. W. Hati, "Rancang Bangun Aplikasi Video Conference Berbasis Python," *Prosiding Seminar Nasional Teknologi Informasi dan Komunikasi (SENATIK)*, Politeknik Negeri Malang, pp. 102-108, 2022.

[2] N. I. Pratiwi, "Penggunaan Media Video Call Dalam Teknologi Komunikasi," *Jurnal Ilmiah Dinamika Sosial*, p. 202, 2017.

[3] B. P. Adhi, "Optimasi Jaringan Lokal (LAN) Untuk Layanan Multimedia," *Jurnal Eltek*, Politeknik Negeri Malang, vol. 19, no. 1, 2021.

[4] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 6th ed. Pearson, 2021.

[5] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 8th ed. Pearson, 2021.

[6] M. O. F. Sarker, *Python Network Programming*, Packt Publishing, 2019.

[7] R. T. Prasetio, "Peningkatan Keamanan Transmisi Video Real-time Menggunakan Algoritma AES," *Jurnal Informatika Polinema (JIP)*, Politeknik Negeri Malang, vol. 6, no. 4, 2020.

[8] A. Prameshwari and N. P. Sastra, "Implementasi Algoritma Advanced Encryption Standard (AES)," *JURNAL EKSPLORA INFORMATIKA*, pp. 52 - 58, 2018.

[9] I. F. Anshori, "Implementasi Socket TCP/IP Untuk Mengirim dan Memasukan Fiile Text Kedalam Database," *JURNAL RESPONSIF*, pp. 1-5, 2019.

[10] A. N. Purwari, N. Suharto and A. Rasyid, "Perancangan Aplikasi Voice Chat dengan Socket Programming pada Android untuk Jaringan Lokal," *Jurnal Jaringan Telekomunikasi*, pp. 113-117, 2020.

[11] E. Salihu and G. Blakaj, "Workplace Chat Application Using Socket Programming in Python," *UBT International Conference*, pp. 1-12, 2021.

[12] A. Shah, G. Servar and U. Tomer, "Realtime Chat Application using Client-Server," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, pp. 2575-2578, 2022.

[13] K. D. Sowjanya and P. B. Reddy, "Development of Voice Call Transfer Service between Android," *REVISTA GEINTEC-GESTAO INOVACAO E TECNOLOGIAS*, pp. 467-480, 2021.

[14] V. BO and K. GR, "Multi-User Chat Application using Client Server Architecture," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, pp. 772-778, 2020.

[15] F. A. Yulianto and K. E. P. U., "Analisis Kualitas Layanan Jaringan Internet Menggunakan Metode QoS Standar TIPHON," *Jurnal Jaringan Telekomunikasi (Jartel)*, Politeknik Negeri Malang, vol. 9, no. 1, 2019.