Available online to ejournal.pnc.ac.id

Journal of Innovation Information Technology and Application

Journal Page is available to https://ejournal.pnc.ac.id/index.php/jinita

Design and Implementation of a Local Area Network-Based Multimedia Messaging Application

*Benjamin Kommey *1, Elvis Tamakloe 1, Joseph Oti Boateng 1, Elijah Ocupualor 1, John Emmanuel Ayarma 1*

1 Computer Engineering Department, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana

email:1 bkommey.coe@knust.edu.gh

ARTICLE INFO

Article history:
Received 03 October 2022
Revised 07 December 2022
Accepted 19 December 2022
Available online 30 December 2022

Keywords:]
Communication
Network
Internet
LAN
Mobile WiFi
Messaging

IEEE style in citing this article:
B. Kommey, E. Tamakloe, J. O. Boateng, E. Ocupualor, and J. E. Ayarma, "Design and Implementation of a Local Area Network Based Multimedia Messaging Application," Journal of Innovation Information Technology and Application (JINITA), vol. 4, no. 2, pp. 104–119, Dec. 2022.

ABSTRACT

Communication these days has taken a huge turn to depend extremely on the internet. Software such as WhatsApp, Facebook, Instagram, and general social media receive about a billion active users. This demonstrates how most communication has relied on the internet. However, this approach does not serve well under some conditions. Firstly, places without internet access suffer greatly from multimedia messaging. They are either unable to initiate a conversation at all or suffer a relatively slower user experience even when they message people in close proximity. Furthermore, the cost incurred in messaging people over these platforms with people nearby could be avoided should the communication be over a local area network rather than depending on the services of an Internet Service Provider. This project investigates the design and implementation of an efficient communication system for devices near each other that relies on any local area network to that a particular device is connected. Such targeted networks include mobile Wi-Fi and cable connections. This work designed and implemented a Local Area Network-based Messaging Platform, capable of working on all local networks regardless of being cable or wireless. The system is extensible, allowing for a node to re-broadcast the network onto other networks it might be connected to and, works without internet access, allowing the sending and sharing of text and media files.

1. INTRODUCTION

Communication refers to the exchange of information between two entities. This information may be of varied kinds including text, images, videos, audio, binary files, and other document kinds. A computer communication system facilitates and eases the process of this exchange from the end users. Various communication systems employ different strategies, and host networks and devices. Since the internet has been in widespread use with a high population of about 4.66 billion active users [1], many communication application giants like Facebook, WhatsApp and Telegram rely solely on the internet. However, there exists a vast majority of people who still do not have access to the internet. According to Weforum, about 3.7 billion people still do not have access to the internet [3]. Most of these people live in local areas or hinterlands. The drive to support such people to communicate with neighbors in close proximity and to avoid the cost of internet access in such cases has led to several experiments on building Local Area Network (LAN) based communication applications. Local Area Network communication apps simply rely on the user's currently connected network, usually a cable or Wi-Fi network to operate. This operation allows for communication with only users directly connected to the same network. However, with a few concepts, this idea can be extended to operate on joint networks. The reliance on the internet for most multimedia messaging these days poses a potentially crippling effect whenever unfortunate situations occur [2]. A town or community may lose internet connectivity due to several reasons including natural disasters.

Also, in some parts of the third world (such as rural areas), internet connectivity is not readily available and at worst, no such connectivity exists. An example is a locality in the Eastern Region of Ghana, '*Bɔn Ase*'. A more global example is the current Russia-Ukraine war which has led to blackouts in some areas [4]. The cost of internet connectivity especially in African localities is very high due to high recorded inflation. Communication within such areas becomes a challenge even in more developed locations where rescue workers and miners in many situations are faced with this struggle. For instance, the network connection of most Internet Service Providers (ISPs) is not reliable underground and therefore, cost-intensive to implement. In their daily work, underground miners and personnel face these issues resulting in their reliance on Wi-Fi technologies among others.

The nature of unreliable internet connectivity has given most attendees of conferences and gatherings a fair share of unpleasantness when online tools such as Slido are used to share opinions. Despite these challenges encountered by several communication schemes developed for area networks such as the wide area network (WAN) and metropolitan area network (MAN), employing LAN-based communication architecture ensures faster, reliable, and highly secured data transfer [5], [6]. In institutions where data security is extremely vital, LAN-based web applications have been developed for use in military missions [7] and many more.

The proposed design comes with a variety of features such as one-to-one messaging, group LAN chat rooms, and broadcast messaging to quickly send information to a group of selected individuals such as attendees of such conferences and gatherings. Also, it does not depend on the internet to operate, providing a much more reliable means of communicating during conferences, unlike internet-dependent tools. The mobile software application can also be used in corporate local networks and organizations allowing for the exchange of information across departments, offices, and between staff members. It is easy to integrate with a business setting because it is easy to install and does not require an external server to operate. It provides security through the encryption of all messages and files exchanged by its users. With the software application installed, individuals living in close proximity can identify and exchange text messages, and documents and even have audio calls with friends and family once they are all connected to the same network. It can be used on a home network, places such as coffee shops, and discussion areas to allow individuals to communicate. There is no need to spend money on the internet or voice data to communicate with users nearby once everyone is connected to the same network. We called this project digiLAN.

2. RELATED WORKS

[8] implemented a Voice Communication Over LAN (VCOL). It sought to build a system to replace standard VoIP solutions that existed for a few reasons. VoIP communications require special network infrastructure and resource personnel to incorporate into one's network. It was also affected by latency and jitter. Additionally, telephones are expensive, and an internet connection was required to use the system even for calls between parties in the same department or network. His solution was a Python desktop application that used a TCP connection to transmit audio between two endpoints. This allowed one to make a call to another connected on the same LAN without any new network hardware overhead. The implementation was for a single LAN. However, it did not extend to other networks and the voice data was not encrypted, leaving room for security vulnerabilities. [9] published their work on instant messaging over LAN in 2017. They proposed a communication scheme where one node is designated as a server on a network and is connected to all other communicating entities (clients). This proved very helpful in enterprise settings and communication within well-networked organizations. However, they did not make improvements in day-to-day communication between individuals without such enterprise networks. The designation of one node as server and server only made it fall short in such cases. Thus, the distinct android applications could not by themselves create a server broadcast to initiate communication. [10] from Babel University published a journal on an authentication scheme for LAN-supported instant messaging systems. They considered the use of fixed passwords and digital signatures to overcome the problems faced using any of these methods alone. They implemented the server-client model and used TCP/IP to communicate between hosts. They also implemented a peer-to-peer model between clients. Evaluations were made in two aspects of their system, user's login security and the overall performance of the system. Their evaluation did show quite encouraging results, however, there were drawbacks as well. The authentication scheme implemented in [10] had a flaw where knowing a secret key by an adversary will let him or her impersonate the real owner. Also, they implemented a chatting system where instant messages between clients were not end-to-end encrypted thus posing security threats to privacy. [11] collaborated to develop

a Java Agent Development Environment (JADE) chat system which they called the JADE Instant Messenger. This application consists of three agents: UserAgent, ChatAgent, and ChatRoomAgent. A peer-to-peer architecture was implemented to allow users the flexibility to create and participate in decentralized systems without servers. The Java II Platform from Sun Microsystems was used as their development language of choice and Borland's JBuilder was used as their IDE. Also, they utilized a set of Java classes freely available on the internet, JADE, to develop the Agents. Messages being transmitted were encrypted and signed en route and messages received at the destination were verified and decrypted. The instant messaging capability of the application was achieved however, scalability was not tested on account of the large number of hosts required. Again, the system failed to handle errors and thus could not achieve robustness. Because no centralized account management was implemented, UserAgents do not have recognizable and persistent names between sessions. Security was also not tested systematically to evaluate vulnerabilities and the application was not tested on other platforms to determine its portability or platform-specific bugs. [12] designed a network communication system that runs on the client/server (C/S) architecture. Socket communication was implemented for their network communication as a peer-to-peer model. It partly adopts the TCP/IP protocol and the UDP protocol. The system allows users to communicate in real time, conveniently, and safely over LAN. The system was implemented such that, after a client initiates a connection request to a server and the server accepts the request, there would be no difference between the client and the server. This characteristic allows two users to send data back and forth equally. UDP was used to get User List and an error-correcting code was implemented at the application level for the UDP data to account for missing data. TCP was used to guarantee the data arrived at the destination correctly and in the same order in which they were sent. After a test between two users, users could get information about online users such as their IP address and the name of the local host. Also, network traffic was reduced, and the speed of transactions was improved because the program was designed to render two users in LAN to transfer data. Though this system proved viable, it fails to address scalability and only supports two entities at a time, a client, and a server.

[13] worked on an offline chat application with an emphasis on resilient disaster management. They offered a solar-powered device as a server to which clients connected over wifi. This meant such communication was possible only in the vicinity of the said device and people had to purchase and install the server for the communication to work. [14] designed a mobile intercommunication application for android. Their implementation was based on the Netty framework and relied on the mysql database. This meant that the application could not function without a computer running mysql on the network. Thus, two mobile phones could not communicate without an intermediary. Another downside of this approach is that their solution was not cross-platform and could run only on the android operating system. LAN Messenger, developed in 2012 by Qualia Digital Solutions, is a p2p chat application for intranet communication and does not require a server. Several useful features including event notifications, file transfer, and message logging are provided. Also, messages are protected by AES encryption with RSA as the key exchange mechanism and past conversations are logged and can be retrieved at any time. This application, however, is only supported on old Windows, Mac, and Linux versions. It sadly has no support for users and users have expressed concerns about it being outdated with frequent bug issues. They also expressed views on the functionality being good but the system has a poor design and implementation. Recently, there was also a report by a user where the installation file contained a Trojan horse virus. [15] developed HYChat, a Hybrid Interactive Chat System for Mobile Social Networking in Proximity. They used two different methods for communication, Over-the-top (OTT) and Device-to-Device(D2D) technology. They alternated between these two technologies depending on the distance between the two users. Their solution worked very well for both close communications and distance communication. However, this approach worked only for communication between two devices and did not support group communication. [16] worked on a chat application for android. Its implementation was based on java sockets. However, no attempt was made to have the implementation work on wider networks and required the client to know the IP address of the server to initiate communication. [17] implemented a voice calling solution over wifi for android. Its implementation was based on User Datagram Protocol as its signaling protocol. Despite being successful, it made no attempts at other methods of communication including text and file sharing.

[18] developed this software 12 years ago after studying the operation of the internet and drawing the conclusion that anyone is unsafe on the internet in terms of privacy and security. BeeBEEP was released under the terms of the GPLv3 license. The main purpose of this application is for office messaging and does not need an external server to let users communicate with each other. Messages are encrypted and delivered directly to the recipient, without intermediaries, through a block encryption algorithm, AES, with a random 256-bit key generated for every single point-to-point connection that is made. Although security

and privacy are ensured, it does not support Android systems. It also compromised its user experience in relation to the user interface by focusing only on privacy functionality. [19] implemented a Multi-hop File Transfer solution. They based their implementation on android WiFi Direct using Cognitive Radio Network. However, the destination of the shared files was intended to be the cloud. This in essence was not particularly helpful for offline communication and offline use cases. [20] implemented an instant messaging tool. The design was based on the java virtual machine and mysql database. This fixed internal communication issues for large corporations and organizations. However, the solution was not feasible to have the server run on mobile devices because of its mysql dependency.

3. METHODS AND PROCEDURES

Every system comprises of different components working in a harmonious synergy to achieve the desired goal. This project is no different and consists of various units or modules that come together to make up the digiLAN mobile software application. Detailed in this chapter is a bird’s eye view of the application. It also entails the system architecture, the system block diagram, the system workflow, and flowcharts of components of the system; all of which have been well expatiated with illustrations.

3.1. digiLAN system architecture

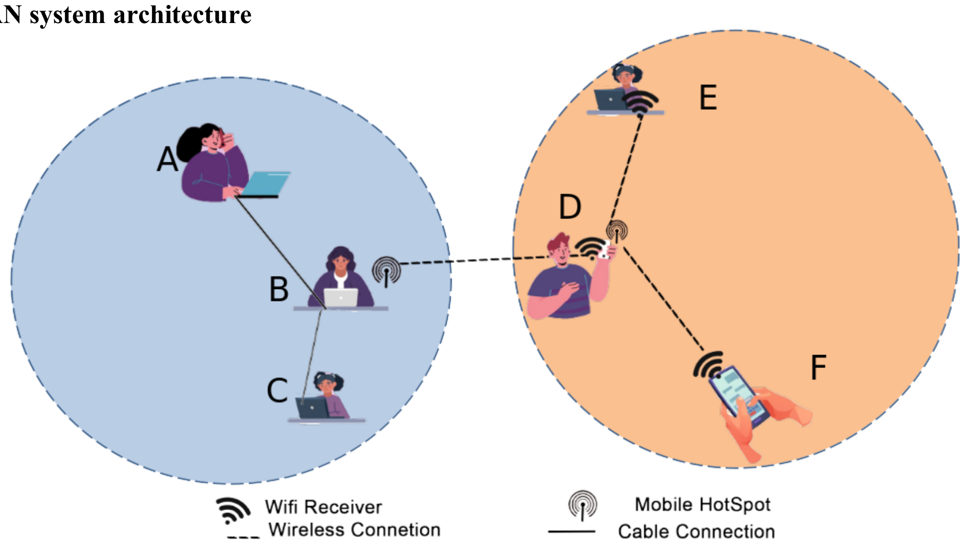


Figure 1. digiLAN system architecture

As shown in Figure 1, the digiLAN system architecture, it can be observed that there are two networks represented by the two large circles bordered by dashed lines. Within these networks are devices or hosts connected using various interfaces. Ethernet connections are represented by solid lines while wireless connections such as Wi-Fi are represented by dashed lines for instant communication between mobile or smartphone devices [21] and computers. The two networks are linked using a wireless connection. The goal of the project is that these two networks, irrespective of how they are interfaced and without regard to how the devices within them are connected in the physical layer, should be able to communicate with each other on the digiLAN mobile software application platform.

3.2. digiLAN system block diagram

For any system to function, the components must work together. This implies that the components relate to one another in a particular way for the purpose of achieving the intended goal of the system. Block diagrams are mostly employed to depict a mapping of how the various components relate. A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks.

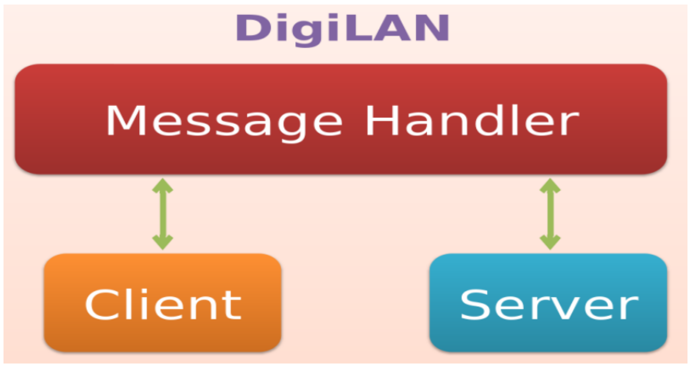


Figure 2. digiLAN system block diagram

The digiLAN application as shown in Figure 2, is made up of three main constituents namely: The client, the Server, and the Message Handler. At any point in time of the application in execution, the said modules will be running in the background concurrently. The components are of high cohesion and are loosely coupled. According to [22], cohesion refers to the degree to which the elements inside a software module belong to each other. Coupling on the other hand is the degree of independence between software modules, a measure of how closely connected two routines or modules are, and the strength of the relationships between modules. Combining these two design paradigms supports the general goal of high readability and maintainability. In addition, other desirable traits of software such as robustness, reliability, reusability, and understandability are advantages as a result.

3.2.1. Websockets

To enable real-time communication, WebSockets were used. To understand the operation of the client and server modules of the Enkomo application, it is expedient to throw more light on the operation of WebSockets. WebSocket was standardized by the IETF in 2011 and has since been adopted in a wide range of applications. WebSocket is a computer communications protocol providing full-duplex communication channels over a single TCP connection. It has lower latency compared with half-duplex polling alternatives. This allows us to build real-time web applications on top of TCP. It also integrates well with HTTP and other protocols. It depends on HTTP 1.1 and generally relies on the HTTP Upgrade Header to change connections from HTTP to WebSockets. Websockets have been used in implementing several messaging applications and push notification systems.

3.3. digiLAN system workflow

This subsection describes the sequential steps that the digiLAN software application goes through to make communication and multimedia messaging possible, that is from the point of launching the application; to starting a server or connecting to a server, message delivery, message reception, and user termination of the application. The system workflow is depicted in Figure 3.

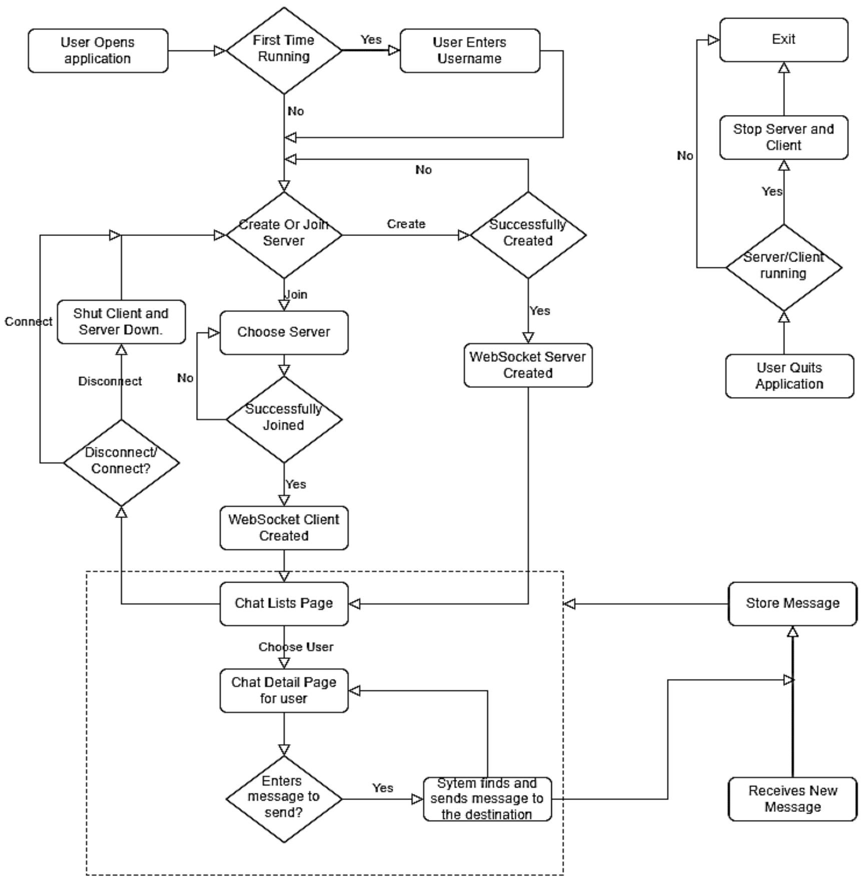


Figure 3. digiLAN system workflow diagram

Each of the components in the digiLAN software application has internal workflows by which they function and play their role effectively by designation. These workflows are illustrated below and described appropriately. The Client component is typically a Web-Socket client.

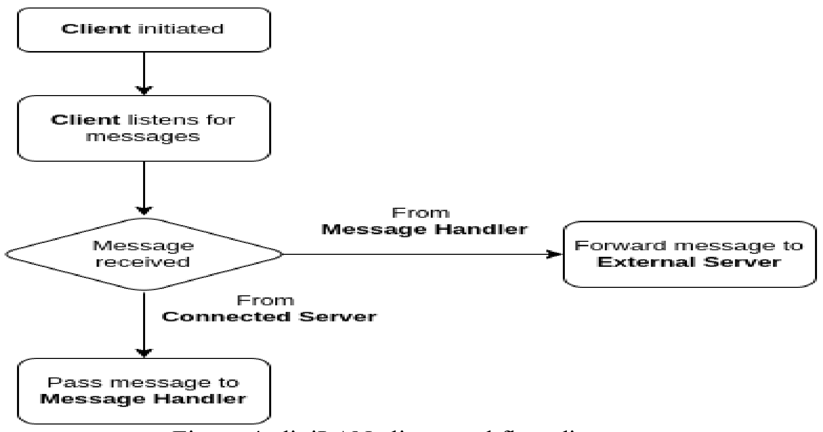

Client initiated
Client listens for messages
Message received
From Message Handler
Forward message to External Server
From Connected Server
Pass message to Message Handler

Figure 4. digiLAN client workflow diagram

The Client workflow as shown in Figure 4 is a receiver of messages from an external server it is connected to. It also sends messages through this same server. The exchange of information between the client and the server is made possible via their unique IP addresses. When the client module is initiated, it listens for messages on port 3355. The client may receive messages from two sources; the Message Handler Module or the external server to which it is connected. By external server, it means that it is running on another device other than the one on which the client is running. A message sourced from the Message

Handler means it has a destination other than the current device. It is therefore forwarded to the External Server for further routing to the intended destination. On the other hand, a message received from the external server means it is meant for this very host. Hence it is passed to the Message Handler. The Message handler processes the message to determine its type and make it available on the user interface.

The digiLAN software application Server is also a Web Socket Server. The workflow of the server is as depicted in Figure 5.

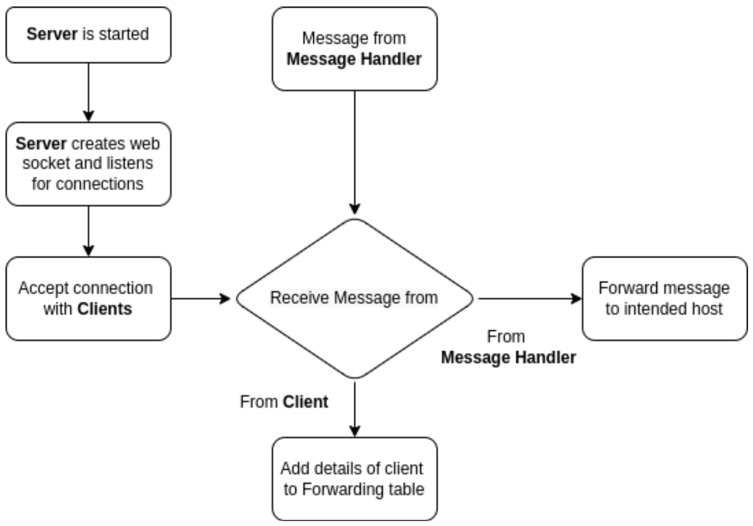

Server is started
Server creates web socket and listens for connections
Accept connection with **Clients**
Message from **Message Handler**
Receive Message from
Forward message to intended host
From **Message Handler**
From **Client**
Add details of client to Forwarding table

Figure 5. digiLAN server workflow diagram

The Server receives messages from external clients connected to it. It also routes messages received from the Message Handler Module through the appropriate path. Running the server module will create a Web Socket that listens for connections on port **3355**. Once an external client connects to this server, its network information is added to the server's table. In this way, the server keeps track of all the clients in its network. A message received by the server from the Message Handler module is forwarded to the intended host client if it is on the same network as the server (i.e. if its details are found in the routing table of the server). Otherwise, the routing table of the server is passed to the Message Handler to determine an appropriate path to route the message. Following this step, the message is then forwarded to another server which is external to the current host to continue with the routing.

3.4. digiLAN system message handler

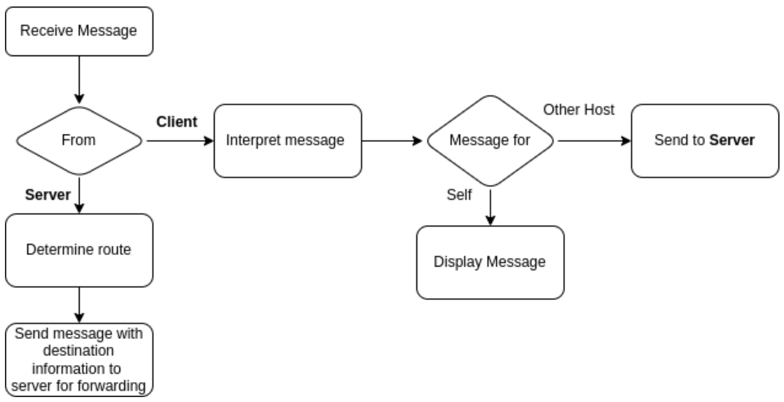


Figure 6. digiLAN message handler workflow diagram

The Message Handler workflow as shown in Figure 6 is essentially the interpreter of messages received in the application. Messages bound to the current host will be passed to the Message Handler by the internal client module. In such a situation, a good interpretation is made of the message. If the message is a text, it is presented as such. A text message is displayed as a formatted string. An image-containing message will be presented in a presentable format on the user interface. Conversely, should a message be bound to another host it is passed to the internal server module? Finally, the routing table of the internal server and some information about messages may be passed to the Message Handler to determine a suitable route to which the message should be passed. This information is relayed to the internal server based upon which the message is forwarded.

3.5. digiLAN system software development process

The software development process employed in this project was the Agile method. We began with a minimally viable version of the application and made incremental changes toward the project objectives.

3.6. Unique ID generation

The basic function of digiLAN as a communication system is to facilitate and ease the process of exchanging information between end users. However, to ensure that users can send and receive messages from the intended parties and that privacy and confidentiality are also ensured, the concept of user identification (User ID) is employed. The User ID is a logical entity used to identify or represent an end user in the digiLAN application. Each user has a unique identifier to distinguish between them. Several attributes of the user object were considered to verify which is unique enough to be used as an identifier. These include username, password, contact, etc. Another attribute that was considered outside the user model was the IP address of user devices. Furthermore, the IP addresses of user devices proved unique enough to be considered user identifiers. The challenge faced in considering this was the difficulty in obtaining the IP address of ios devices. Also, users had to memorize the IP addresses of each other to connect and communicate. Again, a different approach was considered. It is where the attributes of the user device on which the digiLAN application is installed are. It was observed that each device has certain unique attributes that could easily be accessed and utilized. With this implementation, the user ID is generated on the first run of the digiLAN application. The user ID generation process includes the following:

- accessing the details of each device
- Obtaining the timestamp when the application was executed for the first time
- Adding the device details to the timestamp as a string
- Hashing the resulting string using a sha256 hashing algorithm

The result of the hashing algorithm is then used for user identification. This is stored as a key-value pair with a username provided by the user to be easily identified over the network he is connected to. Although the use of the IP address of the devices on which the digiLAN application was installed had inconveniences associated with it, its relevance was still considered, and this led to the implementation of a better alternative functionality where the system scans for available servers once connected to a network without the user having to memorize IP addresses.

3.7. Scanning for available servers

To use the digiLAN platform on a device, it should be connected to a network. The device could connect as a client to another device on the network running a digiLAN server. Alternatively, the device could start a digiLAN server to which other devices on the network could connect. The following outlines how a client will connect to a server. The device connecting as a client will ping all the hosts on the network. Other hosts on the Local Area Network are determined by extracting the subnet mask of the network from the IP address of the device. This gives enough information to know all the possible IP addresses of the network assignable to a host, ie. all IP addresses except the network address and the broadcast address. The digiLAN server and client communicate on port 3355. Therefore, each host is pinged and port 3355 is scanned. An open port 3355 implies that a digiLAN server could be running on the said device. Every device whose port 3355 is open is displayed in a list so that a user can make a choice for a connection attempt.

3.8. Routing

The digiLAN employs a message-passing system and dual routing to determine the message destination. From the core of the app, the message handler implements a basic table where it stores information regarding every user. This information includes their username and a token denoted as the \`userID\` which is generated on the first run of the app. This token is generated using a combination of a timestamp and a user-selected name. The timestamp information ensures that this user ID is as unique as possible. Reliance on device information such as MAC address information was inconsiderable because some devices, such as iOS devices, do not allow direct access to such information. The message Handler keeps a record of which message-sending mechanisms (client and server components) are directly connected to the user it intends to communicate. This information removes the necessity to send duplicate messages through both channels. The Client has no need to implement a routing algorithm as it is connected to only a single host. However, the server, being the core of message passing and deliveries implements its own internal routing table. This table stores information on the userID, username, WebSocket channel, and a user's next hop address (nextHop). Whenever a message designated to a particular user is received, a check in the routing table is conducted to find their information. Then the routing table checks again to see if this said user has a nextHop address. Should it have one, we find and return the information of this nextHop user and send the message through its WebSocket connection. However, if the said user had no nextHop address, then we would send it directly to them as this meant they are a directly connected client. This feature allows for easy message passing and avoids implementing separate algorithms to calculate routes to send messages to a destination.

3.8.1. File transfer

Generally, files have large block sizes. As such transmitting an entire file at a time may not be suitable. As such, the files to be sent are broken into multiple parts. Each part is of the size 2MB. The individual parts are then encoded to JSON as a list of integers. We also attached the total number of parts there for the file and what index is the current part. The message also contains a unique identifier generated from the sender's user ID, the recipient's user ID, and a timestamp for the message. This helps to distinguish when a user decides to send the same file twice to avoid corrupting the files on the recipient's end during reassembly. On receiving these parts, the recipient attempts to combine these individual parts into a single file. However, there are no guarantees that the individual parts will be received in order. To tackle this problem, file parts that are received out of order are cached to disk. In an occasion where this caching fails, the message part may be stored in memory until written to the destination file.

3.8.2. Circular networks

The concept of circular networks plays a crucial role in the implementation of our network architecture.

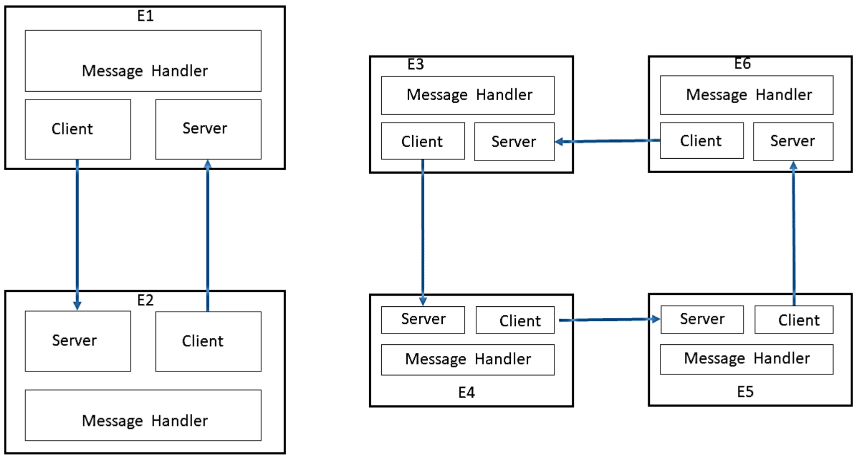


Figure 7. Circular network depiction diagram

Every digiLAN application as described earlier has three main modules: the Message Handler, Client, and Server. The client module connects to the server of another digiLAN application while the server module allows the extensibility of the network. However, there can be instances where for two digiLAN applications, E1 and E2 as illustrated above, E2's client may be connected to E1's server and E1's client connected to E2's server. This situation in Figure 7 is described as a circular network. When this happens, certain direct messages that specify the receiver might work without any issues but other types such as broadcasting the user table become an issue. Because the broadcast has no specific destination, the broadcast message tends to loop or circulate infinitely in a circular network. Thus, this increases the latency of the network. Similar situations can arise among multiple digiLAN applications. A timeout was considered to address this issue, however, that was not ideal. Thus, another method where a special message is broadcasted over the network was employed. If no circular networks exist, the message is expected to reach all connected nodes in the network and end. However, when the message reaches back to the client, the connection is immediately terminated on the assumption that there exists a circular network. In this regard, a new message type called the "check" message was implemented. This is a special message designated to check for circular connections. This message is broadcasted by the current user, if the check message is received back by the current user, it is assumed there exists a circular connection and the client is disconnected. Otherwise, the message is passed on to other connected nodes.

3.8.3. Message formats

The digiLAN Application uses several message formats to make communication possible. Generally, all messages are in the JavaScript Object Notation - JSON. JSON (pronounced 'Jason 'or 'JAY-sawn') is a very popular format for data exchange. It is an Open Standard file format and uses human-readable text to store and transmit data objects in the form of key-value pairs. The values may be numbers, strings of characters, arrays, or serializable (serialization is the process of converting a data structure into a format suitable for transmission or storage in such a way that it can be reconstructed later in the future) data structures. JSON is language-independent and very popular in electronic data exchange between web applications and servers. The JSON format was originally specified by Douglas Crockford in the early 2000s, who worked at Yahoo [23]. Below is a general JSON representation of the messages transmitted in the digiLAN application.

```
{
"to": [recipient],
"destination": [recipient ID],
"from": [sender],
'type': [type of message - file, user-table, introduction, introduction-ack, fetch-table, user-table, check],
'unique identifier': [unique file identifier when doing file transfer],
'message': [message-content-goes-here],
"sent": [true if message is sent else false],
"read": [true if message has been read else false],
'timestamp': [timestamp of the message],
}
```

The message type distinguishes one kind of message from another.

When a new device connects to the digiLAN network as a new client, for example, it introduces itself on the network. The data in an introductory message includes the **username** set for the said device, the **user ID,** and the user's public encryption key**.** The User Table is the digiLAN server's repository of devices currently connected to it at any point in time. During the introduction session, the introductory message content is added to the user table. Consequently, the user table is broadcast to other devices connected to the server on the network. A **user-table** message contains the routing table of the server.

Message Format

Column Name	type	userID	username	public Key
Data type	introduction	string	string	string

3.8.4. Fetch table

Users use this message type to request a user table (list of all connected devices and relevant information) from the server so they can choose whom to chat with. A **fetch-table** message returns to the user the list of all connected users.

Message Format

Column Name	type	User ID	username	public Key
Data type	fetch-table	string	string	string

3.8.5. Text

This is used to request that the server forward a message to a particular connected client. This is the most common message type. Text messages have a general format. Text messages are just sent as strings of characters between endpoints without any need for data serialization or special encoding. This is noticeably fast and is the primal messaging preference in most cases.

Message Format

Column Name	to	destination	from	type	message	sent	read	timestamp
Data type	string	string	string	text	string	bool	bool	Datetime

3.8.6. File

The file message type is only used when a file is to be transmitted. The file is broken down into chunks of 1MB and sent. Due to the use of JSON as the message format, the file chunk is serialized as a list of bytes prior to message transmission. At the receiving end, the list of bytes is read to decode the file chunk. Finally, all the chunks are combined to form the original file. By sending the file in chunks instead of the whole file at once, network traffic is stabilized.

Message Format

Column Name	to	dest	from	type	unique identifier	msg	sent	read	percentage	Time stamp
Data type	string	string	string	file	string	string	bool	bool	float	Date time

3.8.7. Check the message

This handles the processing of 'check' messages. These messages are special and designated to check for circular connections as expatiated above. If the check message received belongs to this user, then this signifies that a circular connection happened somewhere and the client is disconnected, otherwise, the message is passed on to the next nodes connected. In this way, circular connections are tackled and are not expected to make the network congested.

Message Format

Column Name	type	from	message
Data type	*String* (check)	*string*	*string*

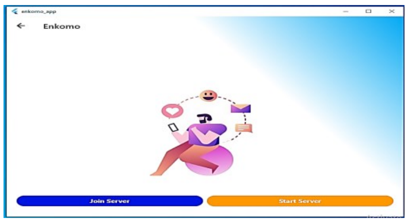
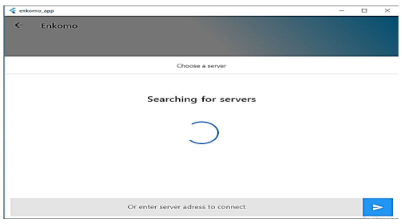
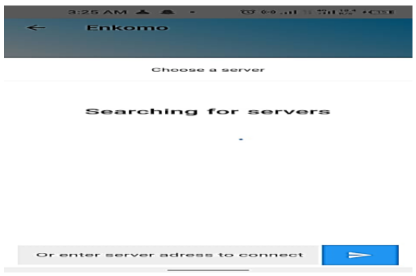
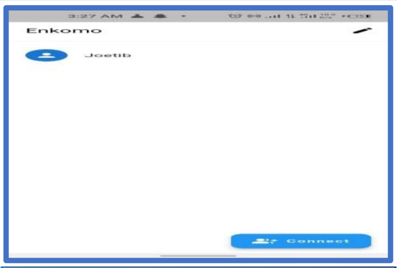
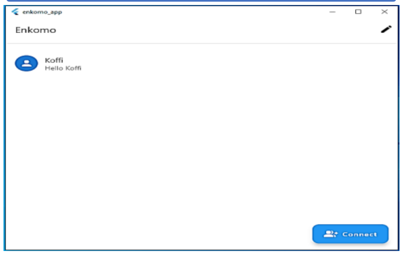
3.8.8. Saving messages to disk

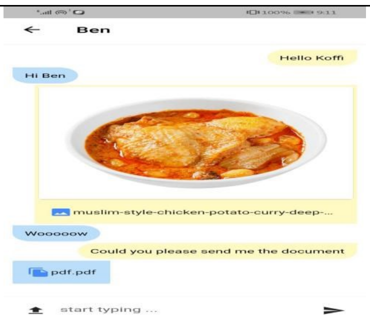
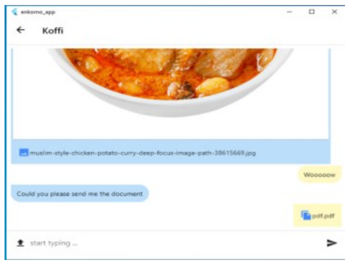
To save messages to disk, the flutter Hive database is used. Hive is a lightweight and blazing-fast key-value database written in pure dart. It is a NoSQL database that is secure with strong built-in encryption and works across multiple platforms. The shared preferences package is also used to read and write simple key-value pair data or store persistent data locally on the user's device. The data stored in shared preferences include the user's name, uniquely generated ID, and their public and private keys. They are later retrieved on application start-up for subsequent use.

3.8.9. Screens and functionalities

The digiLAN application is made up of different pages, each of which allows the user to perform a particular activity. Summarized in Table 1 is the description of the functionalities.

Table 1. Screens and functionalities

Screen	Page description	Functionality
	Desktop Connection Page	The connection page or screen is a page where a user can specify whether to join an existing server on a network or to create one themselves.
 enkomo_app Enkomo Choose a server Searching for servers Or enter server adress to connect	Desktop Select Server page	This page shows a list of scanned servers on the network. The user can then select a server to join or connect to by clicking on the particular server from the list of servers.
 3:25 AM Enkomo Choose a server Searching for servers Or enter server adress to connect	Android select server page	
	Android chat list page	The chat list shows a list of other users the current user has connected to exchange information. The user can select from the list of users to continue chatting with any of them by clicking on a particular user.
	Desktop Chat List Page	

 Ben Hello Koffi Hi Ben muslim-style-chicken-potato-curry-deep-... Woooooow Could you please send me the document pdf.pdf start typing ...	Android chat details page	The chat detail page displays all the conversations that have happened between the current user and the other user with whom the conversation took place. It also displays an input field where a user can type a message to send to the other. A user can send text messages and documents, audio files, and video files as attachments.
 enkomo_app Koffi muslim-style-chicken-potato-curry-deep-focus-image-path-38615669.jpg Woooooow Could you please send me the document pdf.pdf start typing ...	Desktop chat details page	

3.9. Developmental tools

Like every software, digiLAN application was built using several tools including the preferred Integrated Development Environment and available frameworks and libraries. The development tools and stacks used in the implementation of this application include the following:

Flutter

Figure 8. Flutter logo

Flutter, as seen in Figure 8, is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase. Flutter supports web, Android, iOS, Desktop (Windows, Macintosh, and Linux), and embedded platforms. It has a great and vibrant community with large support. Flutter was built using the Dart programming language developed by Google. Flutter code compiles to ARM or Intel machine code as well as JavaScript, for fast performance on any device. With a flutter, developers can build and iterate quickly with Hot Reload. Hot Reload is a feature that allows you to see updates to User Interface almost instantaneously as your code changes. Developers also get to control every pixel to create customized, adaptive designs that look and feel great on any screen. Flutter was chosen because it makes development fast, productive, and flexible.

3.9.1. Device info plus package

This was used to obtain the specific device information on which the digiLAN application is installed. This information forms part of the data used to generate a unique id for the user.

3.9.2. Shared preferences package

This package was used to store some user data locally on the user's device. The SHA256 encryption algorithm encrypts certain data such as user ID. WebSocket channel package to create web sockets as a communication channel between the various clients and servers.

4. RESULTS

Upon finishing the project, we tested the application to verify its functionality. The application was able to send text messages as well as files between devices connected on a Local Area Network as well as between devices across different connected Local Area Networks. Some key areas were of utmost importance to the project which included latency, and memory usage of the application. We tested these two major areas and below are the results.

4.1. Latency

Latency was measured by recording a timestamp of a message before sending it. This timestamp is attached to the message. The current time is recorded on the receiving end, and the difference is computed.

4.1.1. Text message testing

Text messages take an average of 500 milliseconds in transit between two nodes. This includes the time taken for JSON encoding and decoding.

4.1.2. File transfer testing

During File transfer, file parts take an average of 800 milliseconds between nodes. Careful investigation revealed that this sudden shoot-in latency was influenced by the JSON encoding of the bytes of a file into a list of integers. The chart below in Figure 9 details the outcome of how long messages take in transit and the number of nodes they went through.

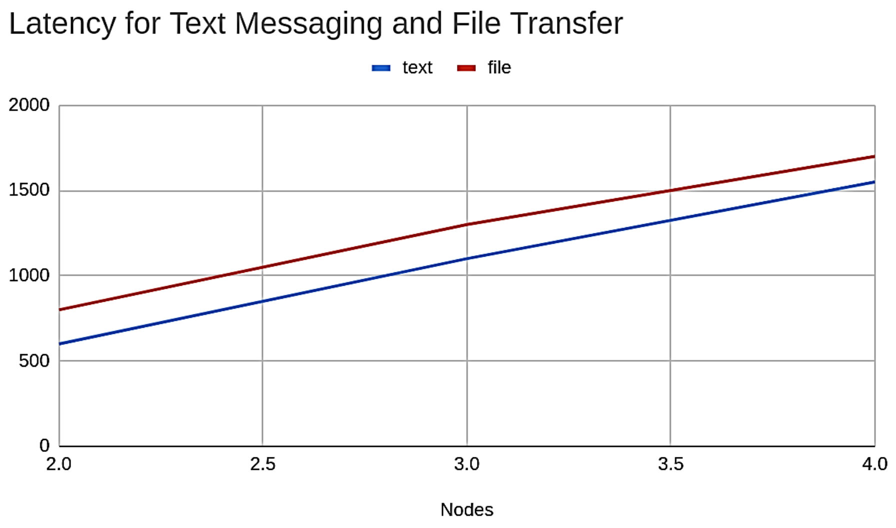


Figure 9. Graph depicting the increase in latency with regard to the number of nodes a message travel through

4.2. Memory usage

On the Windows platform, the solution takes about 40MB of memory space for its operation. However, on android devices, about 100MB of memory is used. Sending text messages does not significantly impact memory usage nonetheless, file sharing results in an increase in memory usage. File parts usually 2MB in size are processed in memory before transit, and on receival. During the process of caching this file part to disk on the recipient end, occasionally and on failure to write to disk, the file part is stored in memory for a while resulting in a significant increase in memory usage. A maximum of 250MB was recorded on all tests regarding file sharing.

Table 2 contains the latencies for the test and evaluation. These values are the time it took to transmit a message between the ends of two, three, and four nodes connected consecutively, measured in milliseconds.

Table 2. Average values derived from the latency test

Message Type	2 Nodes	3 Nodes	4 Nodes
text	600ms	1100ms	1550ms
file	800ms	1300	1700ms

Based on the results obtained from this work, a comparison with related works has been detailed in Table 3 to show the contribution of this work.

Table 3. A comparison of results with related works

References	[24]	[25]	This work
average memory usage	n/a	82MB	70MB
average latency	n/a	n/a	800ms
data security level	moderate	high with IDS (snort)	moderate
applicable platforms	Cross-platform	Cross-platform	Cross-platform
cost of prototyping	moderate	high	low

Therefore, employing this digiLAN application for close proximity transfer of data achieves very suitable outcomes and thus provides reliable, economical, and secured communication with optimal use of computer resources.

5. CONCLUSION

This project saw the successful building and implementation of a Local Area Network Based Multimedia Messaging Application codename digiLAN that is available for multiple platforms including Windows, Android, and Linux. The digiLAN software application is useful for safe, secure, and effective intra-office communications, sharing views in an audience space, and can also be used for communicating with individuals nearby. The digiLAN architecture can be adopted in the implementation of several applications. For instance, the digiLAN system architecture can be adopted in the implementation of a paperless document-handling system for the institution. Individuals including lecturers, students, deans, provost, colleagues, etc. can share information amongst themselves once connected to the institution's wifi without needing internet connectivity or data. Also, this architecture could be used to implement a system that allows our institutions to share resources such as course materials, lecture recordings, tutorials, e-books, etc. with students without internet connectivity. Once students are connected to the institution’s Wi-Fi with digiLAN s network architecture implemented, resources on the institution’s local servers could be shared. With such an implementation, the cost of close proximity communication would be saved. Hence a more cost-effective approach for use by institutions.

REFERENCES

[1] Oberlo.com. 2022. “10 Internet Statistics Every Marketer Should Know in 2021.” *[Infographic]*. [online] Available at: <https://www.oberlo.com/blog/internet-statistics> [Accessed 8 April 2022].

[2] J. Johnson, “Internet users in the world 2021,” *Statista*, 10-Sep-2021. [Online]. Available: https://www.statista.com/statistics/617136/digital-population-worldwide/. [Accessed: 21-Apr-2022].

[3] S. W. Written by Douglas Broom, “Coronavirus has exposed the digital divide like never before,” *World Economic Forum*. [Online]. Available: https://www.weforum.org/ agenda/2020/04/ coronavirus-covid-19-pandemic-digital-divide-internet-data-broadband-mobbile/. [Accessed: 21-Apr-2022].

[4] K. Collier, & Y. Talmazan,. (2022, March 9). Ukraine facing major regional internet outages as Russian invasion continues. *NBCNews.com*. Retrieved April 21, 2022, from https://www.nbcnews.com/tech/tech-news/ukraine-facing-major-regional-internet-outages-russian-invasion-contin-rcna18973

[5] JavaTpoint, “Computer Network Types.” https://javatpoint.com/types-of-computer-networks. (Accessed Nov. 11, 2022).

[6] “Telecommunications, the Internet, and Information System Architecture.” https://www.umsl.edu/~joshik/msis480/chapt07.htm. (Accessed Nov. 11, 2022).

[7] F. van Langen, “An architecture design for LAN-based web applications in a military mission-and safety-critical context.” M.S. Thesis, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, October, 2016, Available: https://essay.utwente.nl/71191/.

[8] D. Masinde, “Voice Communication over LAN.” Webuye, Bungoma: Masinde Muliro University of Science and Technology, 2021.

[9] A R, C. Patel , H N, C., Prasad K, K. and Sultana, S., 2017. Instant messaging over LAN using Android application. [online] *Irjet.net*. Available at: <https://www.irjet.net/ archives/V4/i4/ IRJET-V4I4671.pdf> [Accessed 21 April 2022].

[10] A. Yousif, M. Kareem, and Sadkhan, S., 2007. AN AUTHENTICATION SCHEME FOR INSTANT MESSAGING SYSTEM. *Journal of Al-Nahrain University Science*, 10(1), pp.146-149.

[11] M.B. Mu'azu, M.T. Garba,, Jibril, Y. and Boyi, J., 2007. Development of a JADE Based Distributed Instant Messaging System. In *IC-AI* (pp. 759-764).

[12] G.L. He and L.T. Li, 2009, May. Design and Realize Communication System Based on LAN of C/S Architecture. In *2009 WRI World Congress on Software Engineering* (Vol. 3, pp. 78-81). IEEE.

[13] O. M. Junio and E. P. Chavez, "Development of Offline Chat Application: Framework for Resilient Disaster Management," *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*, 2018, pp. 510-514, doi: 10.1109/IICSPI.2018.8690351.

[14] Du, X. and S. Tang, 2021, September. Design and Implementation of Mobile Phone Intercom App Based on Android. In *2021 International Conference on Computer Network, Electronic and Automation (ICCNEA)* (pp. 91-95). IEEE.

[15] J. Zuo, Y. Wang, Q. Jin and J. Ma, "HYChat: A Hybrid Interactive Chat System for Mobile Social Networking in Proximity," *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, 2015, pp. 471-477, doi: 10.1109/ SmartCity.2015.115.

[16] A.P. Joby, 2016. Socket Programming-Wi-Fi Chat App for Android Smartphone. *Research gate. Net*.

[17] D. S. Kolluru and P. Bhaskara Reddy, "IP to IP Calling Through Socket Programming," *2021 Asian Conference on Innovation in Technology (ASIANCON)*, 2021, pp. 1-7, doi: 10.1109/ASIANCON51346.2021.9544997.

[18] M. Mastroddi, "BeeBEEP - Free Office Messenger - Official Website", Beebeep.net, 2022. [Online]. Available: https://www.beebeep.net/. [Accessed: 28- Apr- 2022].

[19] N.J. Shoumy, *et al.* (2020). Multi-hop File Transfer in WiFi Direct Based Cognitive Radio Network for Cloud Back-Up. In: , *et al.* InECCE2019. Lecture Notes in Electrical Engineering, vol 632. *Springer*, Singapore. https://doi.org/10.1007/978-981-15-2317-5_33

[20] C. Lu, 2017. Design and Implementation of the Instant Messaging Tool Based on JAVA. *International Journal of Advanced Pervasive and Ubiquitous Computing (IJAPUC)*, *9*(2), pp.16-44.

[21] K.D. Sowjanya. and C. Srinu, 2016. Instant Message Transfer between Two Smartphones Using Wi-Fi. *International Journal of Advanced Engineering, Management and Science*, *2*(12), p.239700.

[22] Y. Edward; Constantine, Larry LeRoy (1979) [1975]. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design.* Yourdon Press.

[23] "D. Crockford: The JSON Saga", YouTube. August 28, 2011. Retrieved February 21, 2022. https://www.youtube.com/watch?v=-C-JoyNuQJs

[24] Y. Liu, "Design and Application of Communication Multimedia System based on BS." *Advances in Computer Science Research (ACSR)*, vol. 90, 3rd International Conference on Computer Engineering, Information Science and Application Technology, pp: 294-298, 2019.

[25] B.P. Winasis and B. Sugiantoro, "Design and Implementation of Network Monitoring System on Local Area Network with Social Media Twitter Notification." *International Journal on Informatics for Development*, vol.6, no. 2, pp: 1-6, 2017.