ISSN No:-2456-2165

Decentralized Communication Android SDK

[1]Rohan Maity [1]Dr. M.L. Sharma [1]Dr. Krishan Chandra Tripathi
[1]Maharaja Agrasen Institute of Technology, Rohini, Delhi, India

Abstract:- This paper introduces Decentralized Communications for Chat systems. This is different from traditional methods to build chat communication systems such as Whatsapp. Centralized based systems use centralized server for user sign up, login and user's friend list. In this study, an android chat sdk is proposed that is based on decentralized architecture which uses coroutines for all the networking operations to perform asynchronous operations without blocking. The system is controlled by the users. Each user will set up their own server having the database for friend's profiles and each user has to authenticate among each other before exchanging data or file. SDK will facilitate developers to create decentralized chat apps for users with the help of secure and decentralized communication protocol.

I. INTRODUCTION

➢ *Decentralization Defined*

A decentralized network does not work on single or central server or machine instead it distributes tasks among many nodes/machines.

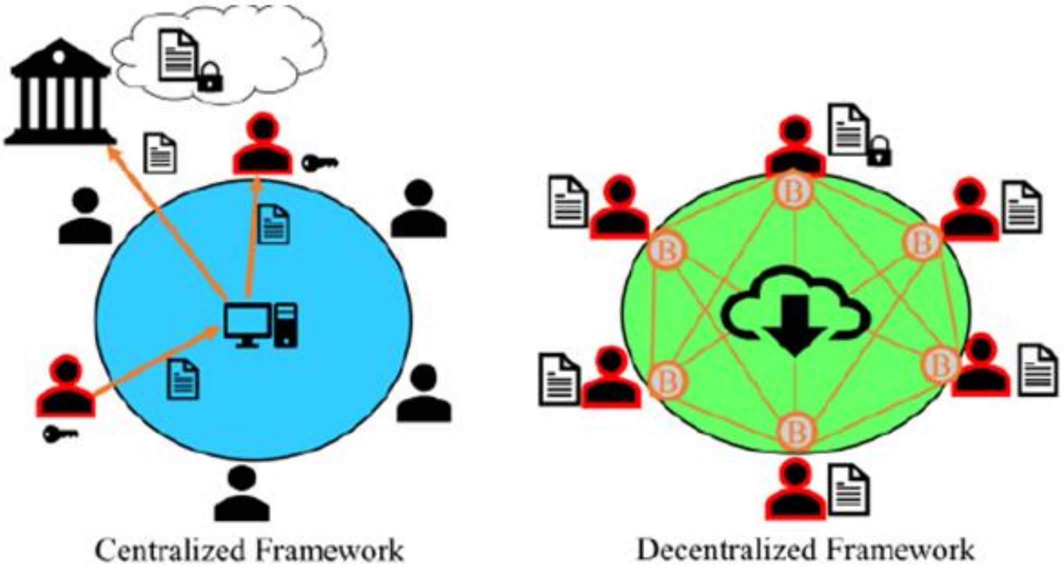


Fig 1:- Centralized System vs Decentralized Systems

The decentralized means there is no central point of control. A lack of a single authority makes the system fairer and considerably more secure. Since there is no reliance on single point there is low risk of data being compromised. Even if data at point is destroyed would result in no loss of data since all the information is stored on multiple devices around the world.

Disadvantages of Traditional Centralized Model

- Risk of a single point of failure . If the central server goes down, the individual "client" machines attached to it are unable to process user requests
- Limited scalability. All applications and processing power are dedicated to single server
- To scale your network, there is the addition of more storage, I/O bandwidth, or processing power to the server. This in turn increase the cost in the long run.

II. METHODOLOGY

Fundamentals of Decentralization Communication:

It includes :

- Decentralized Communication framework (in our case , Matrix protocols)
- Observer Data streams
- Decentralized services

- Decentralized Messaging Framework is responsible for message delivery in a peer-to-peer mode. It's role is to manage the observable data streams and they operate on decentralized services. The Message delivery is based routers where each router only knows its neighbour routers or end-points.

- Observable Data streams: these streams have an observer attached to it. So that when any changes occur in the stream, observer is immediately notified. This concept is taken from Reactive paradigm.

- Decentralized Services : these are the microservices pattern managing server to server connections when exchanging data between them. This can be implemented by sockets. They tend to be executed as much as possible in end-users devices. [1]

- Control of data of third parties in Centralized architecture:

The United States of America. and United Kingdom. have been forcing WhatsApp, Facebook, and other social media platforms to give the encryption keys of certain individuals, allowing these authorities to read their private messages. This treaty has disturbing implications. WhatsApp uses the Signal Protocol, which guarantees end-to-end encryption, preventing even their employees from reading your confidential messages. This means the only way WhatsApp could comply with this is by fundamentally altering their software in contradiction to the Signal protocol. Consider being forced to remove the lock on *your* door to allow the NSA, the FBI, or the police to come visit anytime without notice, warrant, or probable cause. These demands will render the online communications of everyone less secure, not just those of suspected criminals.

- Need of decentralized communication android SDK:

Decentralization is paramount to restoring privacy and freedom to the internet. As long as there are centralized service providers for communication platforms, the government and third parties. will use them to tighten its authoritative grip.

Android being the most popular mobile platform, and most of the people use chat apps on android, decentralized communication android SDK will help developers to create decentralized chat apps for users. Users can use these chat apps to communicate without losing power over their data to third parties. User will be in control of their own data by hosting its own server and stay assured of its data not being compromised.

- Implementation of SDK

We will be using Matrix protocols on which SDK will work upon. Matrix is a secure and decentralized communication protocol. This specification is an open standard designed to make privacy and security the default of the web.

Matrix is an open source and open standard for decentralised, real-time communication over IP.

- it's interoperable, meaning it is designed to interoperate with other communication systems, and being an Open Standard means it's easy to see how to interoperate with it
- Matrix is decentralised, which means there is no central point - anyone can host their own server and have control over their data
- it is designed to function in real-time, which means it is ideal for building systems that require immediate exchange of data, such as Instant Messaging. [4]

Each user connects to a single server, this is their *homeserver*. Users are able to participate in *rooms* that were created on any Matrix server since each server *federates* with other Matrix servers. This means you can talk to anyone on any server. It also means you can host your own server, giving you control over all of your data. Self hosting also gives you the ability to customize your server to fit your needs Each message that is sent in a room is synchronized to all of the other servers that participate in that room. If one server goes offline, everyone else in the room can continue talking. Once that server comes back online it will be sent all of the messages that it missed while it was down.

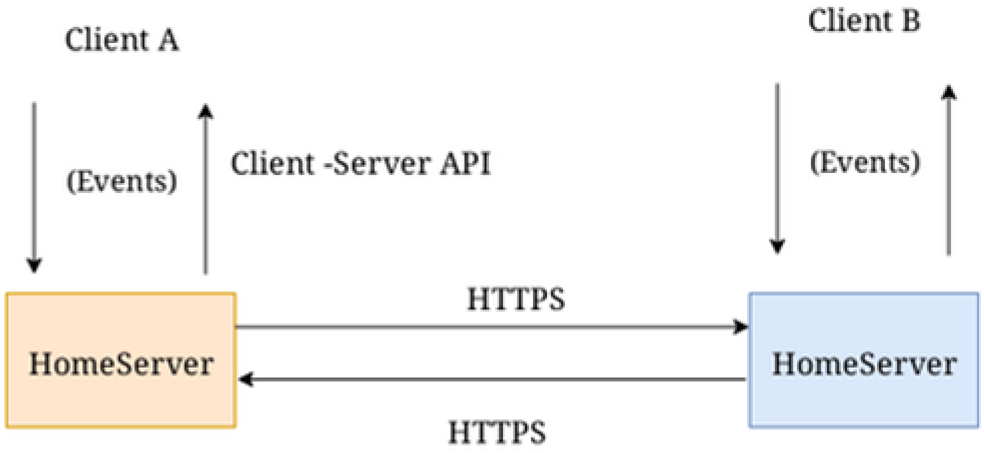


Fig 2:- DCN System architecture

SDK is built in Kotlin language.

➢ *Coroutines for networking*

In function there is no way we can suspend it or resume it. The only operations on the functions we can perform are start and finish. Once the function is started we must wait until it's finished. If we call the function again it starts execution from it's beginning.The situation with the coroutines is different. You can not only start and stop it but also suspend and resume it. It is still different than the kernel's thread because coroutines are not preemptive by themselves (the coroutines on the other hand usually belong to the thread, which is preemptive). [5]

Kotlin also has great feature "Coroutines". Coroutines are a new way of writing asynchronous, non-blocking code. Coroutine are like threads but light weight and like threads, coroutines can run in parallel, wait for each other and communicate. The biggest difference is that coroutines are very cheap and threads, on the other hand, are resource heavy to start and manage. A thousand threads can be a serious challenge for a modern machine.

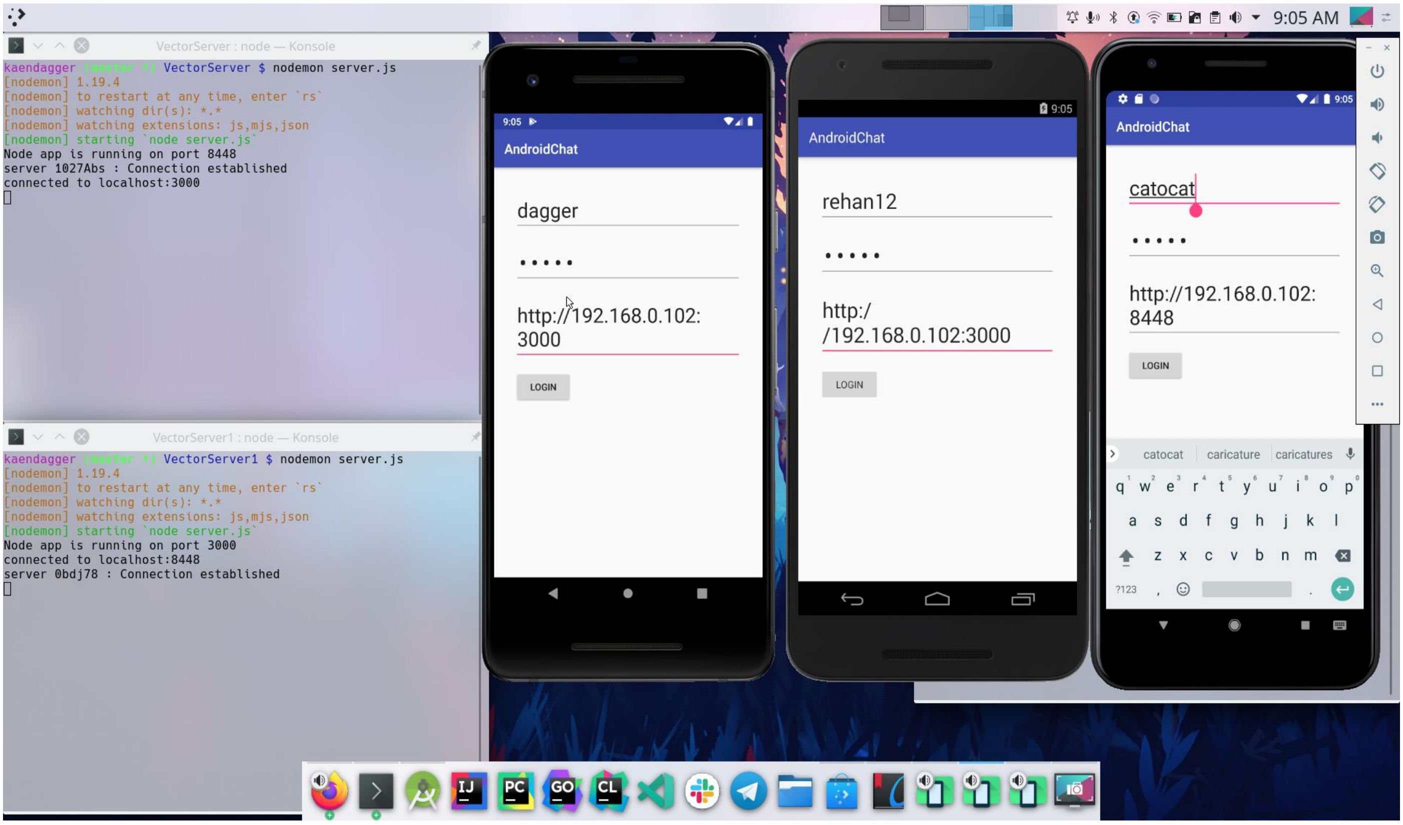
All the network calls (like login, authentication, sending messages etc.) use coroutines for performing non-blocking networking operations

➢ *Self Hosting of server:*

To host your own server then the reference implementation, Synapse could be installed. Synapse is a reference "homeserver" implementation of Matrix. User could setup Synapse server, configure the database for messages, friend list, rooms list etc. own its own.

Every user can run one or more clients, which connect through to a homeserver. The homeserver stores all their personal chat history and user account information.

III. RESULTS



```
kaendagger (master *) VectorServer $ nodemon server.js
[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Node app is running on port 8448
server 1027Abs : Connection established
connected to localhost:3000
```

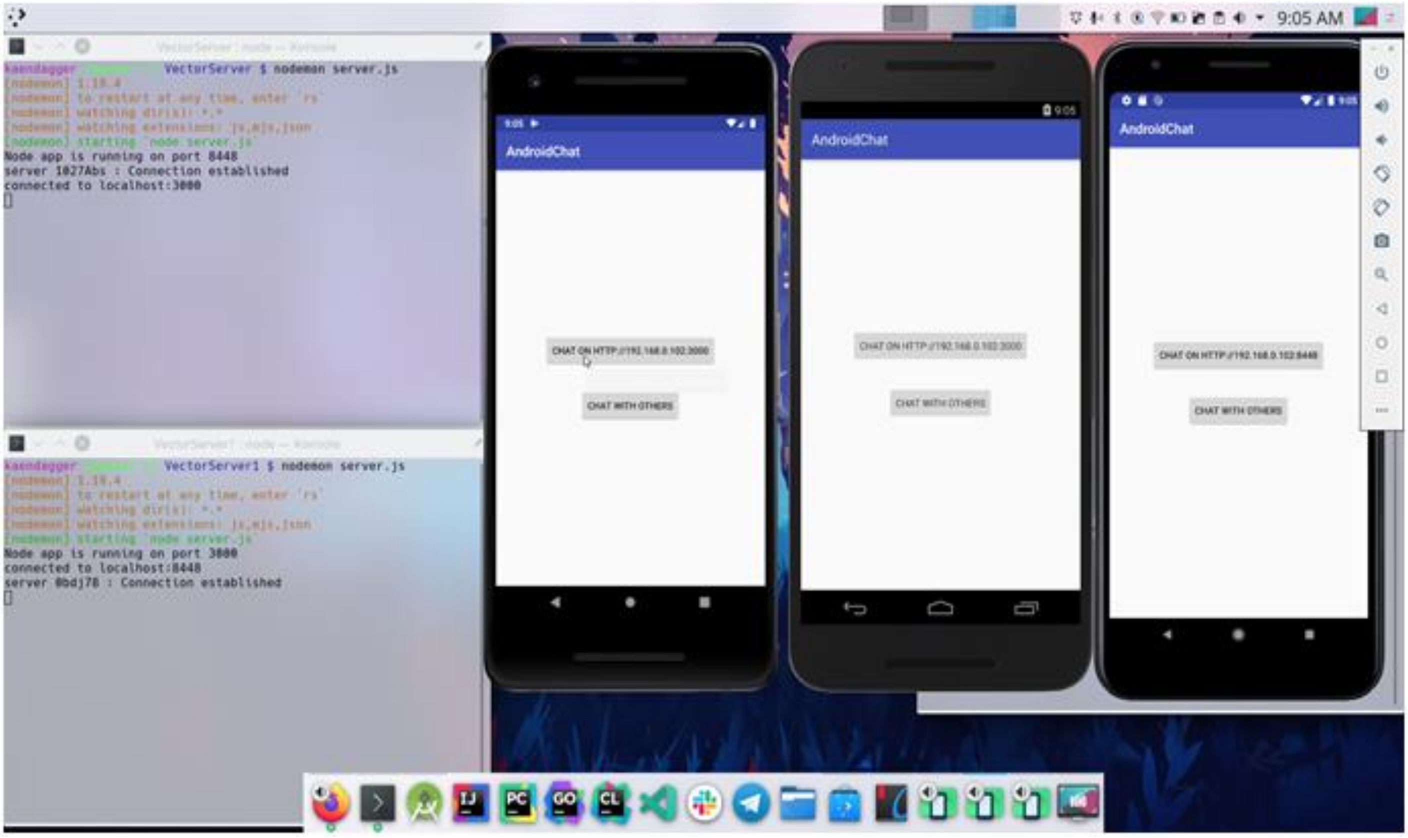
```
kaendagger (master *) VectorServer1 $ nodemon server.js
[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Node app is running on port 3000
connected to localhost:8448
server 0bdj78 : Connection established
```

dagger
•••••
http://192.168.0.102:3000

rehan12
•••••
http://192.168.0.102:3000

catocat
•••••
http://192.168.0.102:8448

Fig 3:- Initial Set up, users connecting to different server

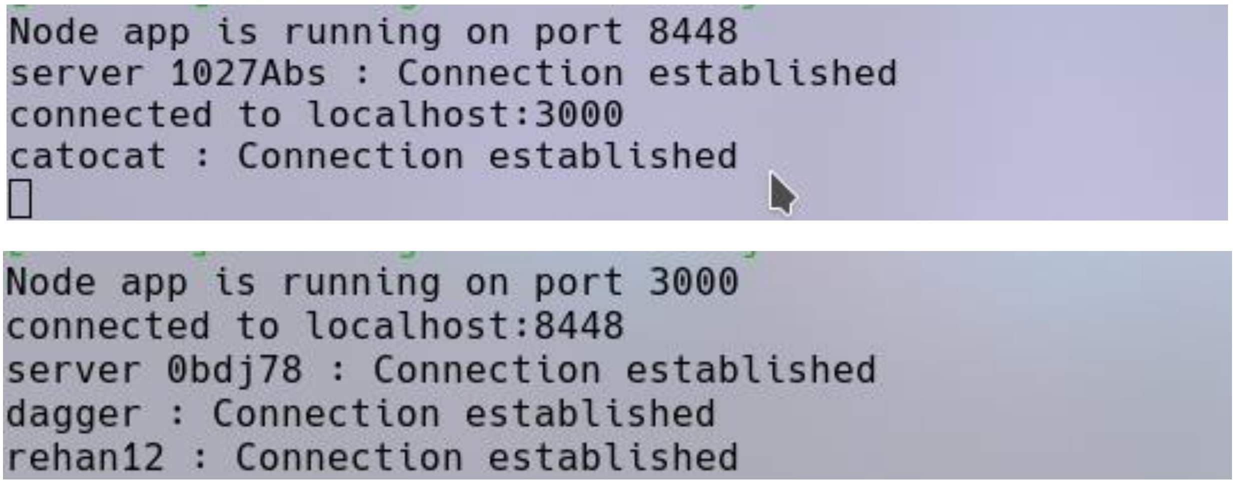


```
kaendagger          VectorServer $ nodemon server.js
[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Node app is running on port 8448
server 1027Abs : Connection established
connected to localhost:3000
```

```
kaendagger          VectorServer1 $ nodemon server.js
[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Node app is running on port 3000
connected to localhost:8448
server 0bdj78 : Connection established
```

9:05 AM
AndroidChat
CHAT ON HTTP://192.168.0.102:3000
CHAT WITH OTHERS
AndroidChat
CHAT ON HTTP://192.168.0.102:3000
CHAT WITH OTHERS
AndroidChat
CHAT ON HTTP://192.168.0.102:8448
CHAT WITH OTHERS

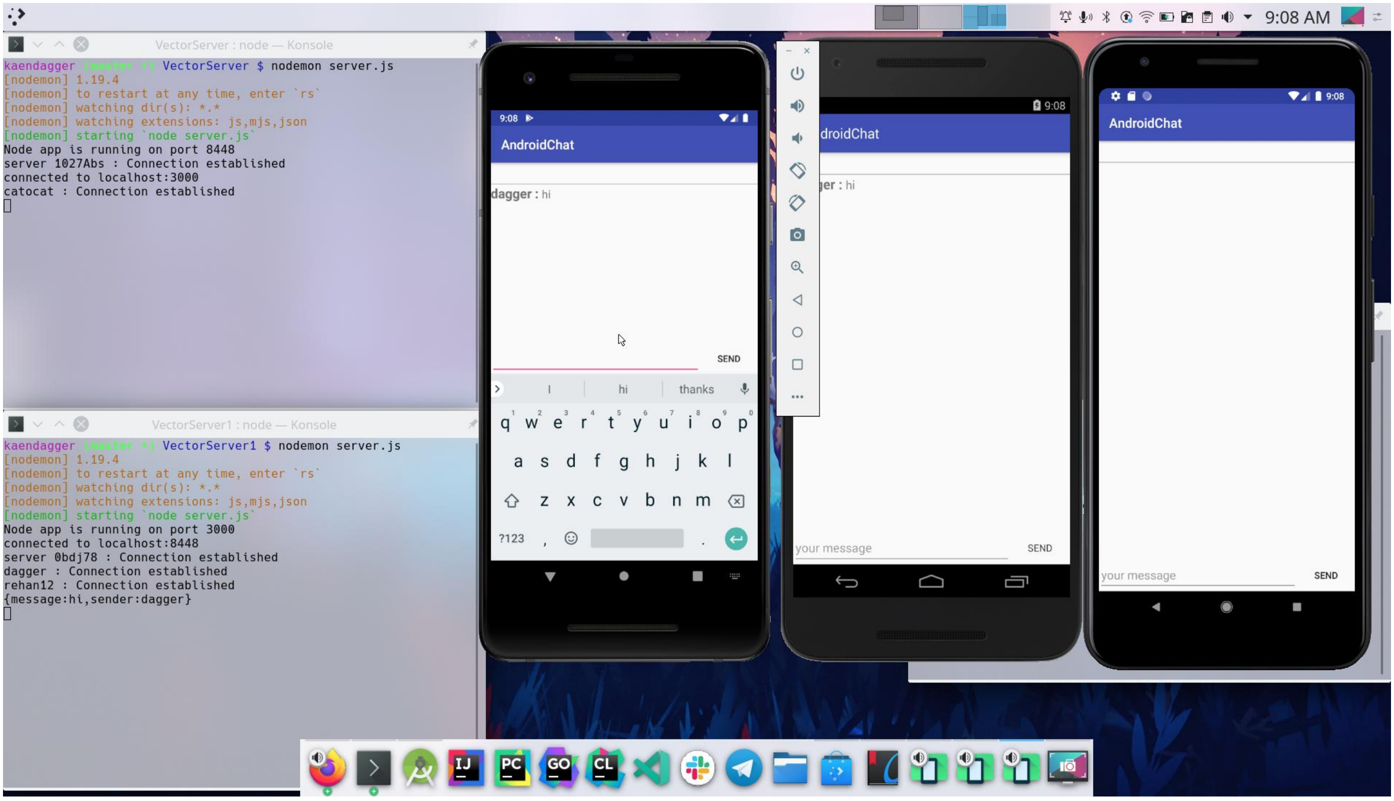
Fig 4:- Select Chat on particular server



```
Node app is running on port 8448
server 1027Abs : Connection established
connected to localhost:3000
catocat : Connection established
```

```
Node app is running on port 3000
connected to localhost:8448
server 0bdj78 : Connection established
dagger : Connection established
rehan12 : Connection established
```

Fig 5:- Users connected to different server

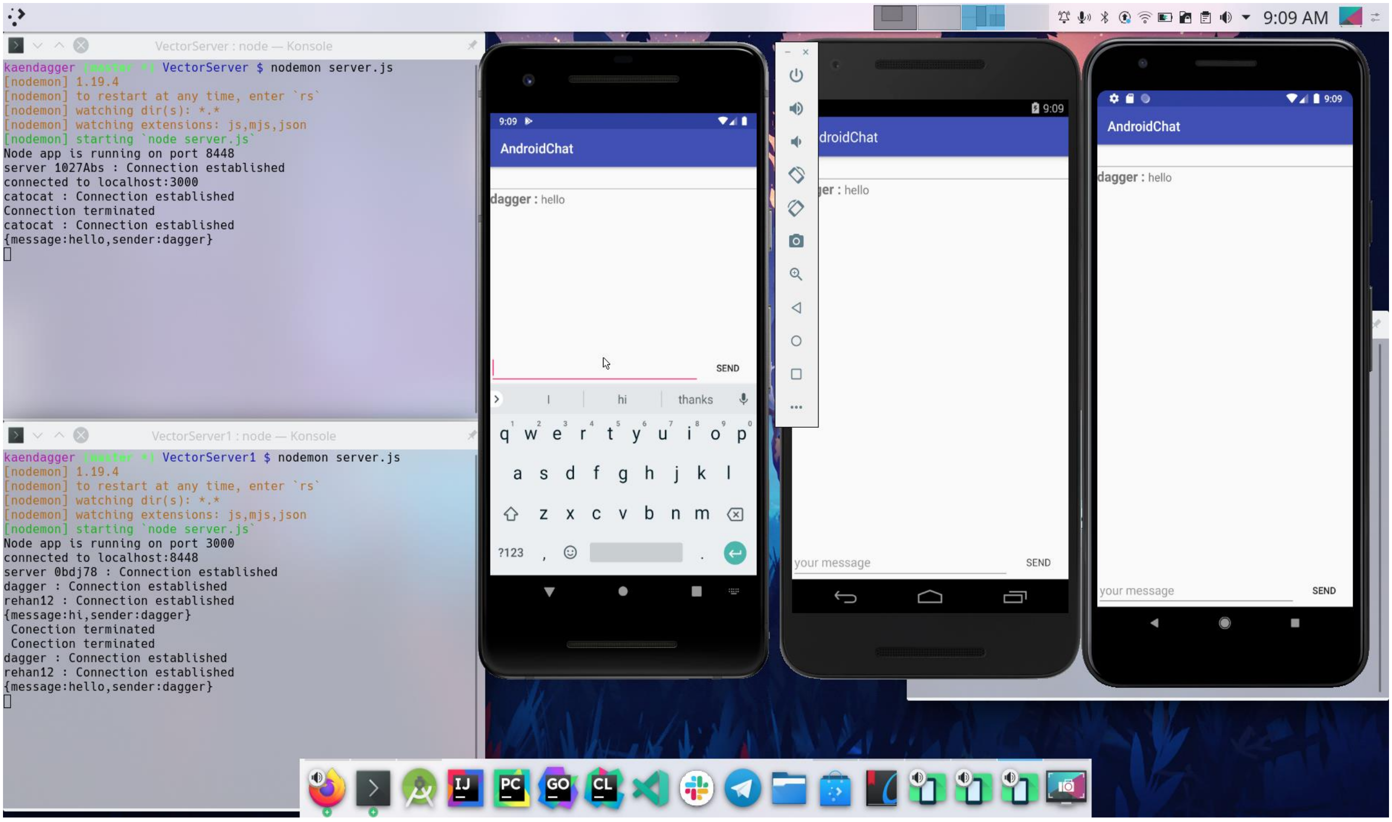


```
VectorServer : node — Konsole
kaendagger (master *) VectorServer $ nodemon server.js
[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Node app is running on port 8448
server 1027Abs : Connection established
connected to localhost:3000
catocat : Connection established
```

```
VectorServer1 : node — Konsole
kaendagger (master *) VectorServer1 $ nodemon server.js
[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Node app is running on port 3000
connected to localhost:8448
server 0bdj78 : Connection established
dagger : Connection established
rehan12 : Connection established
{message:hi,sender:dagger}
```

dagger : hi

Fig 6:- User on different Server didn’t receive message



```
VectorServer : node — Konsole
kaendagger (master *) VectorServer $ nodemon server.js
[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Node app is running on port 8448
server 1027Abs : Connection established
connected to localhost:3000
catocat : Connection established
Connection terminated
catocat : Connection established
{message:hello,sender:dagger}
```

```
VectorServer1 : node — Konsole
kaendagger (master *) VectorServer1 $ nodemon server.js
[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Node app is running on port 3000
connected to localhost:8448
server 0bdj78 : Connection established
dagger : Connection established
rehan12 : Connection established
{message:hi,sender:dagger}
 Conection terminated
 Conection terminated
dagger : Connection established
rehan12 : Connection established
{message:hello,sender:dagger}
```

dagger : hello

Fig 7:- Users on different servers communicate using matrix protocols

- *Comparison other existing platforms*

Factor/ Name	Matrix-ios-sdk	Matrix-android-sdk	Implemented-sdk
Language	objective-c/swift	java	Kotlin
Environment	Could be built on MacOS only	Could be built on MacOS, Linux, Windows	Could be built on MacOS, Linux, Windows
Set up cost	Costlier , since only MacOS is supported	Any machine with 8gb RAM ,i-5 processor works	Any machine with 8gb RAM ,i-5 processor works
Asynchronous operation	RxSwift, use threads	RxJava, uses threads	Coroutines, uses light-weight threads
Code Sharing b/w platforms	No, only for iOS	No, only for android	Yes, with Kotlin/Native code could be shared between iOS and android
E2ee, Room, notifications features	yes	yes	No

Table 1

IV. CONCLUSION

SDK will facilitate developers to create decentralized chat apps for users with the help of secure and decentralized communication protocol. And users can have a self-secured individual user database. User can store all their personal chat history and user account information without losing power over their data. User will be in control of their own data by hosting its own server and stay assured of its data not being compromised.

REFERENCES

[1]. Paulo Chainho , Steffen Drüsedow†,Ricardo Chaves, Nuno Santos, Ricardo Lopes Pereira , Haensge† , Anton Roman Portabales, Decentralized Communications: Trustworthy Interoperability in Peer-To-Peer Networks

[2]. Kundu, Anirban. (2012). Decentralised indexed based peer to peer chat system. 416-419. 10.1109/ICIEV.2012.6317378.

[3]. Mohamad Afendee Mohamed, Abdullah Muhammed and Mustafa Man, A Secure Chat Application Based on Pure Peer-to-Peer Architecture

[4]. M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta and A. Neal, Decentralised real time interoperable communication with matrix

[5]. Kahn, Gilles & Macqueen, David. (1977). Coroutines and Networks of Parallel Processes.

[6]. Kremenova, Iveta & Gajdos, Milan. (2019). Decentralized Networks: The Future Internet. Mobile Networks and Applications. 10.1007/s11036-018-01211-5.