IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 3, Ver. III (May-Jun. 2014), PP 61-66
www.iosrjournals.org

Bluetooth Messenger: an Android Messenger app based on Bluetooth Connectivity

[1]Amrita Deb and [2]Swarnabha Sinha
[1](Information Technology, Institute of Engineering & Management, India)
[2](Information Technology, Institute of Engineering & Management, India)

Abstract: *The project discussed here is an Android messenger application which connects using Bluetooth. The main concepts discussed here are : i) Bluetooth connection between two or more android smart phones, whereby users can chat with each other ii) Bluetooth connection between a server and an android smart phone ,whereby the user can update and synchronise his/her chat records with the server from time to time. iii) Data structures used in storing and updating the data (messages) against respective usernames. iv) State machines and finite expressions used to achieve robustness, thereby delivering error free messages. This whole project has been designed using python programming language. This app doesn't require an internet connection rather uses the in-built Bluetooth facility in a phone. Hence it comes in handy for charge-free short distance communication between individuals within a certain range (30 or 150 ft, depending on the hardware).*
Keywords: *Android, Bluetooth, Messenger, Python, Short distance communication, sl4a*

I. Introduction

Messenger applications are very much in vogue these days. Whatsapp, WeChat, Hangouts, etc are a rage in the app world. But all these messenger applications exploit either mobile data which is a paid service or Wi-Fi which is not always available and when available the connection strength fluctuates from place to place. Even when one wants to send a message to another person on the same floor or a few feet away they have to rely on the availability of these two. Our motive was to create a messenger that facilitates communications in a small firm (say college or hospitals or clinics) free of cost. To accomplish this, two technologies came into our minds—(i) Wi-Fi (ii) Bluetooth

Wi-Fi vs. Bluetooth:

Wi-Fi devices tend to interfere with signalling of surrounding devices which can prove quite harmful when dealing with certain sophisticated and life saving devices(such as in a medical environment).In addition to this, Wi-Fi technology requires comparatively more power and also other devices: Wi-Fi Router, Wi-Fi Modem, Wi-Fi Hotspot.

On the other hand, Bluetooth requires low power resulting in longer battery life. It can be used within the range of 30-feet, which is reasonable within a small building. The data-transfer rate of the Bluetooth technology is quite acceptable, i.e. 721Kbps. It is readily available since it's in-built in every Android phones and moreover the cost of setting up a Bluetooth network is much less in comparison to that of Wi-Fi
Hence for all the aforesaid reasons we chose Bluetooth over Wi-Fi as our medium of communication.
Apart from general messaging between to users, the application also has an additional feature which allows users to save their chat history to an Android based server for storage thus not crunching on the phone memory as well

II. Methodology

This app significantly uses Bluetooth hence on start the application firsts checks if the phone's Bluetooth is on and if it's not the app asks user to give permission to turn it on. After the phone's Bluetooth is checked, if it is the first time the application is being run it will ask the user to enter its name. This name entered by the user will be sent during the Introduction section of messaging where the users are introduced to one another by the name they take up. A file 'udetails.txt' is created recording the user's input. This data is fetched and sent to every phone with which the user communicates. This feature gives a better identification of the user since while scanning for nearby Bluetooth devices it is often very difficult to identify the user.

The app has essentially three types of terminals:

i) The communicating 'server phone' terminal
ii) The communicating 'client phone' terminal
iii) The 'storage server' terminal

(i)& (ii) form the two phones involved in messaging whereas (iii) is the Android based storage server

The app offers three basic operations:

i) communicating with a nearby user via Bluetooth
ii) transmitting chats to Android based storage server

After the app has turned on the phone's Bluetooth and is ready, it first asks its user with what it wants to connect: a device (for operation (i)) or a server (for operations (ii))

When connected to a device:

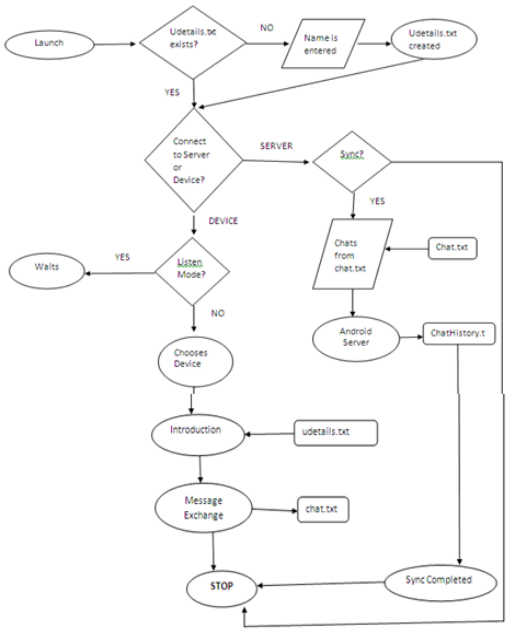
Whenever two phones are communicating one has to be a server and the other has to be the client. The server will remain in listen mode until it gets a connection request from a client. When a phone chooses to be a client it is shown a list of nearby Bluetooth devices with which it can communicate to. The client selects a member from the list of available devices and if that device is in listen mode it will accept the connection and send a message to the client notifying it of the establishment of connection and requesting its identity. The client phone will then access its 'udetails.txt' where its name is stored and send it to the server. The server receives the client's name and in turn sends its own name to the client by accessing its 'udetails.txt'. This part of the communication is termed as 'Introduction' where the two users are introduced with each other and are then all set to chat.

All the messages exchanged during a chat session is recorded and written into a file 'chat.txt.'

When connected to a server:

When the app connects to a server, it is to transfer the 'chat.txt' file which records the messages of the user's chat sessions with other users. The phone sends the message 'sync' to the server followed by the contents of chat.txt and its name as mentioned in'udetails.txt'. The server has a file named 'ChatHistory.txt' (or creates one if it's not present) where it stores the chat histories of all the phones it serves. When the server receives the keyword 'sync' in the message from a phone it stores the message that comes after it mapped with the phone's name and updates its 'ChatHistory.txt'

III. Flowchart



IV. Algorithms

Step algorithm of communicating server phone:

1. Launches Bluetooth Messenger
2. Inputs its name
3. Chooses to be connected to a Device
4. Chooses to be in Listen Mode and waits
5. When a client tries to connect it asks for identity
6. Receives client identity and saves it
7. Sends its identity to client
8. Inputs and exchanges Messages

9. All messages exchanged gets stored in 'chat.txt'

Step algorithm of communicating client phone:

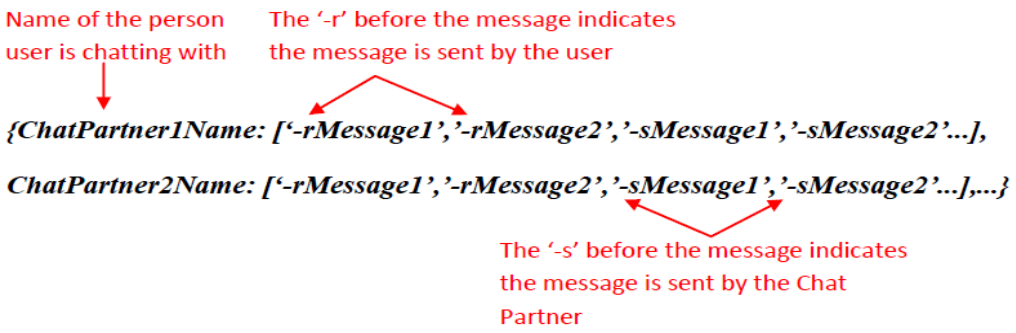
1. Launches Bluetooth Messenger
2. Inputs its name
3. Chooses to be connected to a Device
4. Chooses not to be in Listen Mode
5. Selects the device it wants to chat with from 'List of Nearby Bluetooth Devices'
6. Sends its own identity from udetails.txt on server phone's request
7. Receives server's identity
8. Inputs and exchanges Messages
9. All messages exchanged gets stored in 'chat.txt'

Step algorithm of the Android based storage server:

1. Server remains in Bluetooth Accept mode
2. User sends a message saying 'sync' followed by contents of its chat.txt with its name from udetails.txt
3. Server receives the sync message and stores all the contents that follows
4. Server picks out the name of the user which is differentiated from the contents of chat.txt by the symbol '|'
5. It searches its ChatHistory.txt for the particular username
   - 5.1 If it finds that name in its records it updates it
   - 5.2 If it does not find the username in its record it creates a new key with that username and adds the messages
6. 'Sync Completed' message is announced

V. Data structures

The data structures used in this project includes data dictionary, array list and/or combination of them. The data structure of 'chat.txt' present in the user's phone, which holds the record of the user's chat with different chat partners it has communicated with, is as follows:



Here it is a simple dictionary containing the Chat Partners names (according to their udetails.txt) as keys and the value of each key is the messages exchanged between the user and the Chat Partner stored in the form of an array list**.** Every message in the array list is preceded by either '-r' or '-s'. These distinguish the messages of User from that of Chat Partner. '-r' means that the message is sent by the user to Chat Partner and '-s' means the message is received by the user from Chat Partner

The data structure of 'udetails.txt', which contains the name of the user, is a simple array containing one value i.e. the user's name

["User"]

The contents of 'chat.txt' and 'udetails.txt' are combined together and sent to the server during syncing in the following manner:

{ChatPartner1Name: ['-rMessage1','-rMessage2','-sMessage1','-sMessage2'...],

ChatPartner2Name: ['-rMessage1','-rMessage2','-sMessage1','-sMessage2'...],...}|"User"

The data structure of ChatHistory.txt on the server side, in which the server stores the chat history of all the devices it serves, is a nested dictionary as follows:

{User1: {ChatPartner1Name: ['-rMessage1','-rMessage2','-sMessage1','-sMessage2'...],

ChatPartner2Name: ['-rMessage1','-rMessage2','-sMessage1','-sMessage2'...],...},

User2: { ChatPartner1Name: ['-rMessage1','-rMessage2','-sMessage1','-sMessage2'...],

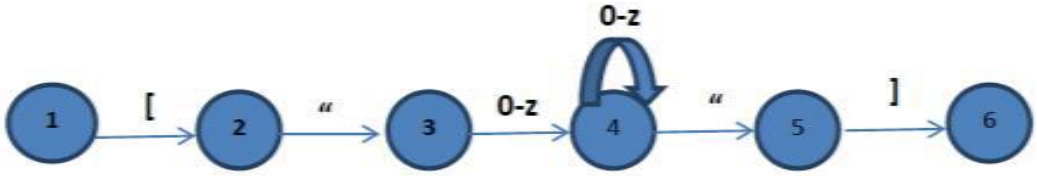
ChatPartner2Name: ['-rMessage1','-rMessage2','-sMessage1','-sMessage2'...],...}...}

The outermost dictionary's keys denote the names of the users the server serves as storage for. The value of each of these keys is another dictionary. The keys of this inner dictionary are the names of the various chat partners that respective key of the outer dictionary has chatted with. The value of the inner dictionary's keys is an array list containing the messages exchanged. The semantics are same as that in chat.txt.

VI. Finite Machines And Expressions

The data is prone to error due to external disturbance or noise hence misinterpretation by the user. To increase the robustness of the application we introduce the element of Automata, where we use the concepts of finite machine and finite expressions to get data /files transferred properly without any error. Like this very project necessitates the creation of a text file called 'udetails.txt' in the client's smart phone which contains the name of the user.

The finite machine of the above process of sending the file 'udetails.txt' is as follows:



The finite expression of the above process of sending the file 'udetails.txt' is as follows:

$$\ [\ “ \ (0\text{-}z)^{+} \ ” \ ]$$

In the same way we can send the other text files 'chat.txt' and 'chathistory.txt' which contains the conversations between users.

The finite expression for 'chat.txt' is as follows:

$$\{[\ “\ (0\text{-}z)^{+} \ ” \ : \ [\ [\ “ \ - \ (q\text{-}r) \ (0\text{-}z)^{+} \ ” \ _{?,}]^{+} \ ] \ _{?,}]^{+}\}$$

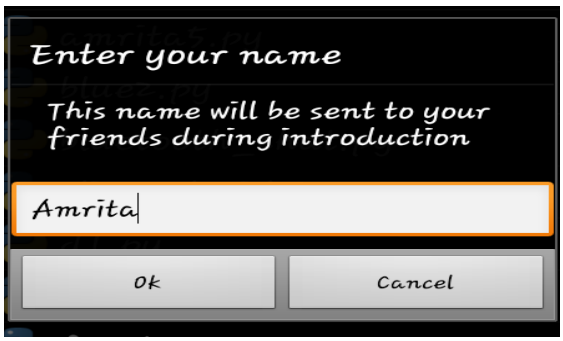
The finite expression for 'ChatHistory.txt' is as follows:

$$\{[\ “ \ (0\text{-}z)^{+} \ ”\ : \ \{[\ “\ (0\text{-}z)^{+} \ ” \ : \ [\ [\ “ \ - \ (q\text{-}r) \ (0\text{-}z)^{+} \ ” \ _{?,}]^{+} \ ] \ _{?,}]^{+}\} \ ] \}$$

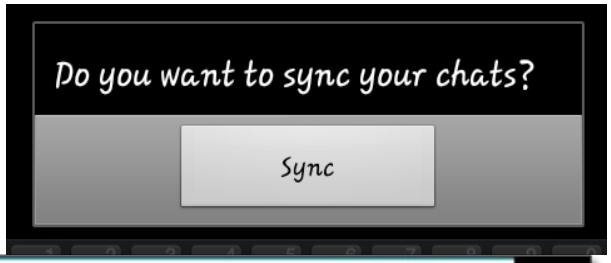
VII. Results

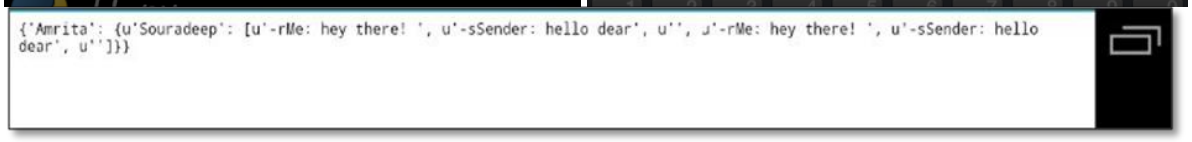
The first screen is the splash screen for the app. When the user clicks on 'Launch' the app checks if the phone's Bluetooth is on and if it's not it asks user's permission to turn it on

After the Bluetooth is switched on and if it is the first time the user is running the app then it will ask the user to enter a name which will be showed during the introduction section of the connection establishment

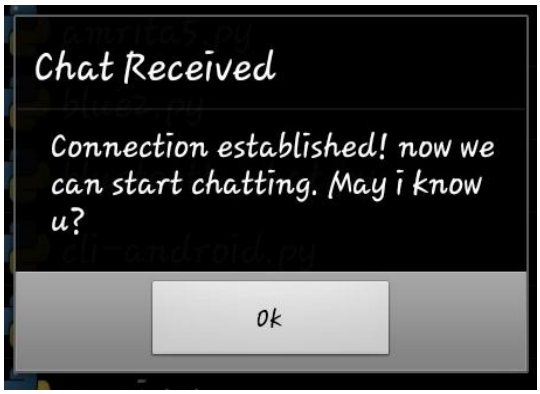


Once that's covered, the app asks whether user wants to connect to a server(for syncing chats) or device(for chatting).When user chooses 'Server' and clicks on 'Sync' the recent chats in the user's phone's 'chat.txt' gets updated in the server's 'ChatHistory.txt'. Once the process is completed 'Sync completed' message is announced on both machines.

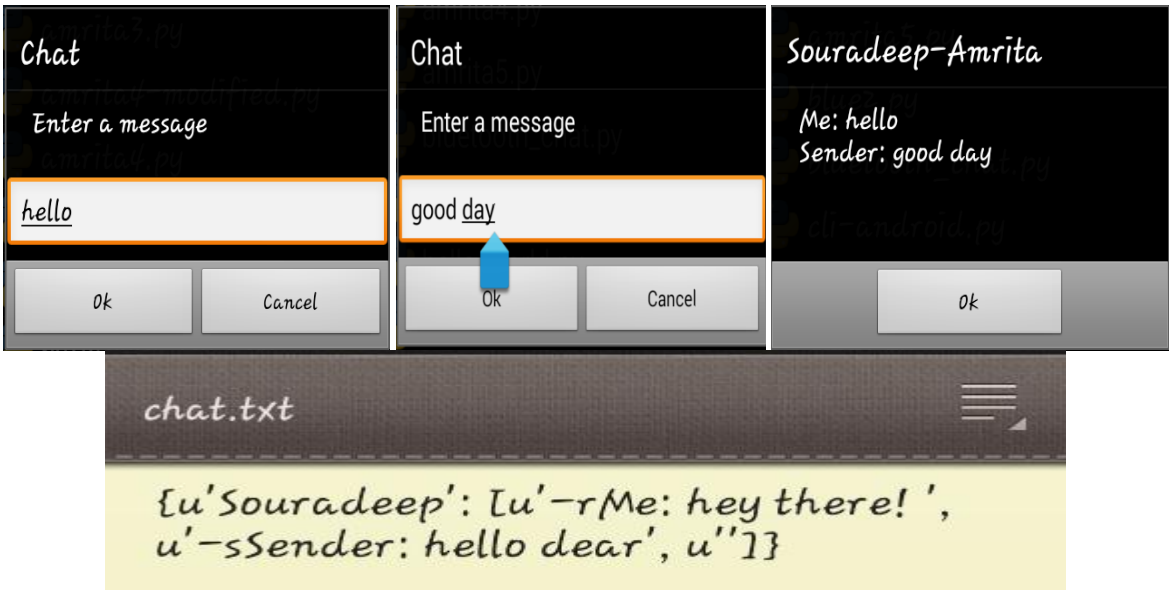




When the user chooses 'Device' he is asked whether he wants to be in the 'Listening Mode'. If the user chooses 'no' he is given a list of all the nearby Bluetooth devices, else he is only notified when someone wants to establish a connection with him



When a user is in 'Listen Mode' he serves as a server while the other serves as a client. The client chooses a server from the list of Bluetooth devices. Connection establishment notification is sent to client from communicating server requesting its identity commencing the 'Introduction' section (discussed above)
After this the message exchanging commences, and every message exchanged gets stored in the 'chat.txt' file of respective phones



VIII. Future Work

Enhancing security by encryption:

The encryption is a very important part in this project. Since the project has functions like synchronise or update where the server is sent message files (here chat.txt) from the user's phone, it becomes very necessary that the server as well as the phone holds the correct copy of messages. All the conversations between the user and his chat partners is stored in the text file 'chat.txt', so if someone accesses that file and changes or tampers with the messages then there can be discrepancy and inconsistency. The same can happen when someone tampers with ChatHistory.txt stored on the server side.

So we can use python's library cipher to encrypt our data in the files of chat.txt and chathistory.txt.

Making two player games using the same technology: like tic-tac-toe etc.

IX. Conclusion

From a proper analysis of positive points and constraints of the system it is inferred that the system is working as per the objectives of the project. Installation is a hassle-free task. It requires internet connection for one time to install the application and then it is on its own. The user interface is user friendly and does not require specialized training or skills to operate it. The methodology section of this technical report describes the implementation of this system in detail. The data structures used in this system need to be tuned in to get a more efficient system. We hope that the implementation algorithm and the data structure described here will aid the developers' community in near future.

Acknowledgement

We would like to express our deepest appreciation to all those who provided us the support to complete this project. A special gratitude to our project guide Mr.Avranil Tah, whose advice on technical aspects has been priceless. We would also like to thank our project mentor Prof.Avijit Bose for the invaluable suggestions and encouragement given to us throughout the tenure of the project. Furthermore we would like to thank the faculty of Information Technology of our college who gave us the permission to go ahead with the project and to use the required computer laboratories.

References

[1]. **https://code.google.com/p/android-scripting/wiki/ApiReference**
[2]. **http://www.tutorialspoint.com/python/**
[3]. **http://www.python.org/**