IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. XI (Mar-Apr. 2014), PP 113-119
www.iosrjournals.org

Alltalk™- A Windows Phone Messenger with Cross Language Communication

Shruti Shetye[1], Akhil Abraham[2], Royston Pinto[3], Sonali Vaidya[4]
[1](BE-IT Student, Information Technology, St. FrancisInstitute of Technology, India)
[2](BE-IT Student, Information Technology, St. Francis Institute of Technology, India)
[3](BE-IT Student, Information Technology, St. Francis Institute of Technology, India
[4](Lecturer, Information Technology, St. Francis Institute of Technology, India)

Abstract:*In day to day life, messengers or chatting applications provide facility for instant messaging over the internet. Exchange of messages takes place in universally used languages like English, French, etc. where both the users know how to communicate in a common language. Thus chatting on mobile phones is a luxury when both the parties involved know a common language. Hence we have implemented ALLTALK™ which is a Windows 8 phone based chatting application which makes cross language communication possible using mobile programming and networking technology.This application will enable the communication between two persons irrespective of the language each user wishes to use individually. The various modes of communication available in this messenger are through text and voice. Due to the best processing power provided among the available smartphones and high battery life we choose to work on windows 8 platform. Thus we have successfully eliminated the language barrier and enable ease of communication through this application.*

Keywords: *Cross Language communication, instant messenger, socket connection, translator,Windows phone app.*

I. Introduction

Today communication technology is at its best and communication happens every second. With new technologies like social networking site, internet calls, Whatsapp, Facebook and other types of messengers' people are connected 24x7 and sharing of information is just a touch away. But when it comes to communicating with people of different cultures and backgrounds language becomes a barrier. A third person or a separate software is often needed to overcome this hurdle. There are over 6,000 languages around the world, with each person having his own language, culture and tradition. Looking at the diversity in languages and having the above mentioned communicating technologies around, language becomes a barrier.

Some solutions to this problem can be using language translation applications or software which allows you to enter the required text as input and get the output in the desired language. But this is a time consuming task where the users have to repeatedly keep entering the text that is required to be translated and understood. This way it is inefficient to maintain a continuous healthy conversation between two persons.

Hence we have implemented an application called ALLTALK™ which can improve these inefficiencies in communication. It is a messenger which will enable people around the globe to talk with each other continuously in the language of their own choice without changing between apps to get the text translated and chat. It is an integration of a chatting application and a translator app which makes the communication between different languages easier and more efficient. It eliminates a third person or application to translate the text into the desired language and thus saves time.This application is based on three tier architecture. The first tier consists of the clients who use the messaging interface of the application. The second tier is the application server which acts as a link between the clients and the database server and the translation engine. The last tier contains the database server.

II. Related Theory

A. Translation Applications

In 2008, Google launched the Google translate API which was used to translate phrases, sentences and content of website..SimultaneouslyMicrosoft launched the Bing translate API. Both these can be implemented on different platforms for developing translator software. Recognizing the need for handy translator's mobile application developers created many applications to translate text and images using the Google and Bing API. Some of the applications are as follows:

Google Translate is a free app that allows translatingspeech; text, handwriting, and even text into images by using Google translate APIs. It can translate between over 70 languages and has an added feature of the results spoken aloud. The Conversation mode is ideal for communicating with someone when you don't share a common language, but the app can struggle with some accents. There are also download offline language packs

in case there is no network connection for the app to work. **iTranslate** is one of the most popular translation app on iOS and is now available for android users as well. It's an app that covers over 70 languages and provides a dictionary, text-to-speech, Romanization (which converts unfamiliar characters into English letters), and voice recognition via an in-app purchase. **Bing Translate i**sMicrosoft's free language translation offers for Windows Phone similar features like Google Translate but free of cost. It allows translating text, speaking into your phone, and hearing translations spoken aloud. It also offers the option to snap a photo of a sign or a menu and translate the text within it. There's an offline mode that enables to download a language pack and use the app even when there is no network connection.

The only disadvantage of these apps is that the text to be translated is to be copy pasted every time into the input area, hence users cannot continuously chat with persons talking in a different language. Also in certain apps like iTranslate two people can chat but both have to share a table top with a single device. Thus we implemented ALLTALK™ where people can remotely chat in different languages.

B. Instant Messaging

Early usage of IM&P (Instant Messaging and presence) started with the introduction of the UNIX operating system. Users were able to get the limited Presence information and send instant messages using "FINGER" and "TALK" commands respectively in the UNIX environment. The presence information was limited to the last time a user accessed the account and the location. The instant messaging capabilities were limited to plain text messaging. In UNIX systems, users were able to manage the information they wished to share as a response to a "FINGER" query. They also had the control over accepting or rejecting a talk request [1].Internet relay chat (IRC) was introduced to the online community in 1988 in order to provide real time, conversational capability among users who were connected to a public network anywhere in the world [3]. IRC offered an environment where multiple users can join and leave a chat room at any time. It also eliminated the basic restriction of being on the same network to chat while still offering the means to initiate a private communication between two users.

ICQ ("I Seek You") beta version was released in November1996 by Mirabilis. ICQ enabled users to chat simultaneously over the Internet without joining a chat room. By January 1997, ICQ had 27,000 users with a growth rate of 100 percent per week. Meanwhile, America Online's (AOL) Instant Messenger (AIM) increased its subscribers to ten million users. In mid-1998, America Online (AOL) acquired ICQ, which had achieved more than ten million users by that time. Microsoft MSN Messenger and Yahoo Messenger were both released within a year after that acquisition. With the introduction of AIM, ICQ, Yahoo! Messenger, and MSN Messenger IM became a field where large corporations were developing proprietary codes, which were not interoperable. In 1998, Jabber project was initiated to build an IM client and server that could interact with the various proprietary systems by using a superset of all of the major consumer IM systems [4]. The Jabber/XMPP technologies were originally created by Jeremie Miller, a developer from the "silicon prairie" of Cascade, Iowa, who had written one of the first parsers (XParse) for XML 1.0.Tired of having to run four different IM clients to communicate with all his colleagues, and finding no choice but closed services such as AOL Instant Messenger (AIM), ICQ, MSN Messenger, andYahooMessenger, Miller decided to build an XML-based open-source alternative. He announced the Jabberd server project in January 1999[2]

To overcome the lack of interoperability and other concerns in IM, such as security, authentication, scalability and integration with other business applications, the IETF formed two working groups focusing on instant messaging and presence at different points in time. The following sections will examine the generic model as well as the standards prescribed by the SIP for IM&P Leveraging Extensions (SIMPLE) and Extensible Messaging and Presence Protocol (XMPP) working groups [1].

C. Current Protocols

Currently there are many instant messaging or Chatting applications using various protocols like the SIP for IM&P Leveraging Extensions (SIMPLE) and Extensible Messaging and Presence Protocol (XMPP).There is ejabberd, which is XMPP server that provides quite good features of open source, Whatsapp and Facebook messaging uses some modified version of this. Ejabberd based protocols are used by some other chat applications likes Samsung's ChatOn, Nimbuzz messenger.

XMPP architecture consists of three elements, the XMPP client, XMPP server and gateways to foreign networks.The underlying connection between two XMPP elements id using TCP.Before the content session, a TCP connection for xml streaming must be established first[5].

Even though XMPP is not wedded to any specific network architecture, to date it usually has been implemented via a client-server architecture wherein a client utilizing XMPP accesses a server over a TCP connection, and servers also communicate with each other over TCP connections. [8]The latest update of the window 8 phones SDK provides us with features like Incoming sockets, new improved socket APIs, text to

speech API forover40 languages and core kernel which is the same that runs on windows 8PC. Thus, this makes Windows 8 Phone a fitting choice to realize the goal of ALLTALK™.

III. Proposed System

A. System Architecture

The ALLTALK™ messaging app is based on the client server architecture. All the messages sent by the clients are inserted into a database with the parameters such as the message_id, sender_id, receiver_id, source language and target language. Whenever the user wants to view the previous message thread he/she sends a request to the server for the same. Thus the client uses a very light weight application which occupies minimal space on the device. All the messages and contact details of the users are stored permanently in the database and retrieved only when required to view.

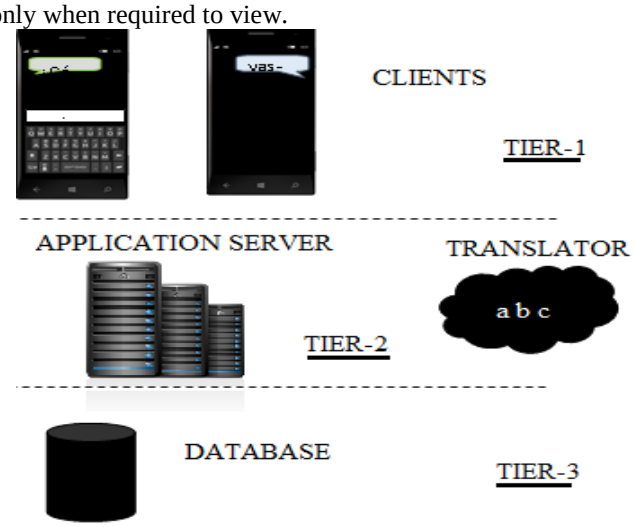

Fig 1 System Architecture

ALLTALK™ is based on three tier architecture. The first tier consists of the clients who use the messaging interface of the application. The second tier is the application server which acts as a link between the clients and the database server and a translation engine. The last tier contains the database server.

1. Client

The client is the windows 8 enabled device on which the users would install the app from the Windows App Store.

- On the first installation of the app the client has to register with the system by providing the user name, unique email-id and the preferred language for communication.
- The system generates a random ID which is provided to the user through mail which is the ATPIN.
- The client is identified throughout the lifetime by his/her ATPIN.
- A registered user can later change the language preference, add contacts, delete contacts, send messages, receive messages, and listen to the sent and received messages.
- The sending and receiving of the messages is accomplished using TCP socket programming.

2. Application Server

- The application server, which is the second tier is the intermediate for the clients to add and remove messages from the database.
- The server receives the messages from the clients in a string format with a transaction code appended to each string.
- On receiving the string it extracts the transaction code from the string and accordingly performs the specified functions.

Since there can be many clients that do different tasks at the same time we have programmed the server to accomplish multitasking using multithreading. Each string that the server receives leads to the formation of a worker thread. A queue data structure is implemented so as to en-queue and de-queue the threads and access the processing functions.

3. Translation

So as to achieve language independent chatting we are using the translation API which is provided by Microsoft Bing Translation engine. The Bing server provided us with a client ID and a key which is used to acquire the Admin Access tokenin order to implement the API's using a web service.

- Each client sets his/her preferred language of communication.
- Whenever a message is sent by a client it is always converted to an intermediate language which in our case is English.
- Whenever any message is received by the client socket it first extracts the transaction code to detect whether the message string is for display purpose or processing.
- The string content which contains the messages are compared to check whether the target language of the text is same as the source language.
- If it matches, then the text is displayed as it is else parameters like the source language, the target language, the sender_id, the Admin Access Token are sent to the translation server and the retrieved message is displayed in the message thread.

B. System Flow

The System contains three main entities which are the client, the server and the database server. The client is the windows 8 enabled handheld device, whereas the server is the application server which handles the traffic on the incoming and outgoing data in the form of messages, registration details, etc. At the end we have the database which is the storage entity which has data about all the transactions.

Sending Messages

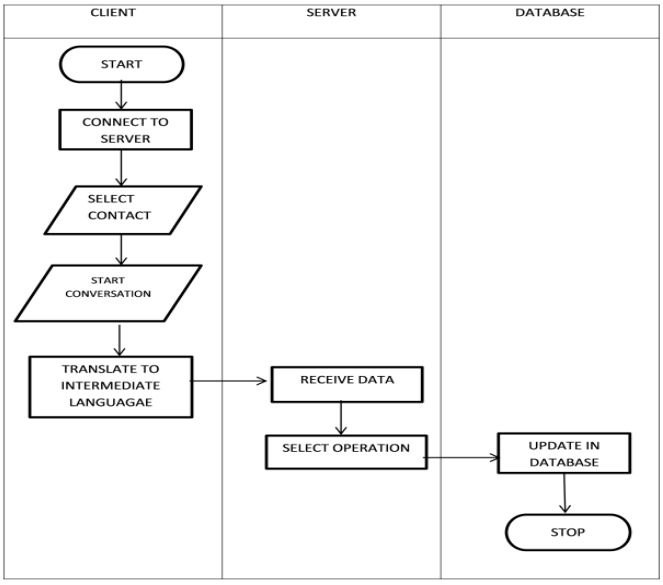

Fig 2 System Flow Chart for sending messages

- **Client**
  - **Connect to server:** The app starts the socket and sends a connection request to the server to start the conversation.
  - **Select contact**: The user gives an input to select the contact with whom he wishes to communicate.
  - **Start Conversation**: When the socket connection is successfully established, the user can send the messages in the language of his own choice.
  - **Translate to intermediate language**: The message text is converted to an intermediate language which will be stored on the server.
- **Server**
  - **Receive data:** The server receives the data through the socket. As soon as a string is received a worker thread is created which is assigned the string to work upon.
  - **Select Operation**: The specific transaction code is extracted from the string and appropriate operations are performed like inserting a message, deleting a contact, adding a contact etc.
- **Database**
  - **Update Database**: The data are accordingly updated in the database from the server.

Fig 3 Functioning of sending process

Receiving Messages

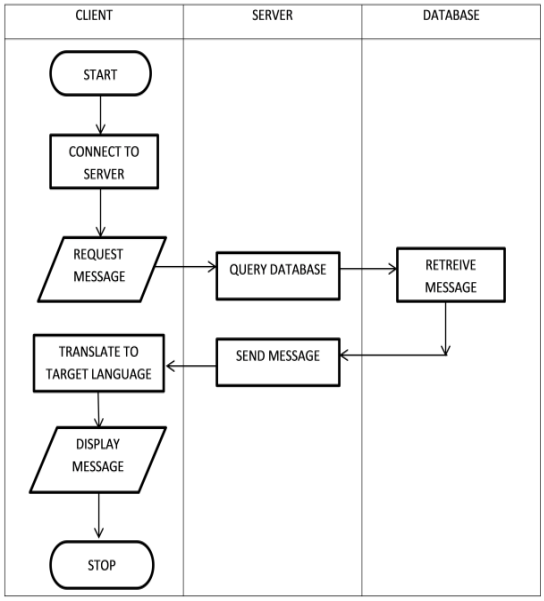

Fig 4 System Flow Chart for receiving messages

- **Client**
  - **Connect to Server**: Initiates a request to the server on client startup to create a TCP socket connection.
  - **Request Message:**The client sends a request to the server to check whether there are any pending messages to be forwarded to the intended client.
  - **Translate to target language:**On receiving the messages from the server, the strings are compared with the intended target language. If they don't match, then the messages are translated to the target language.
  - **Display message:** The translated messages are then displayed as a continuous message thread.

- **Server**
  - **Query Database:** The server further appropriately queries the database for any such messages.
  - **Send message:** The messages that are retrieved from the database are sent to the client in order of the timestamp through the socket connection.
- **Database**
  - **Retrieve Messages**: The server runs a query to select the messages of the specific sender and receiver id and returns them in order sorted according the timestamp.

msg_id	timestamp	sender_id	receiver_id	msg	sender_lang	receiver_lang
101	<Binary data>	ATiceouza	ATrutietye	How are you?	French	German
102	<Binary data>	ATiceouza	AThilraham	hey how are you?	French	English
103	<Binary data>	ATyston	ATiceouza	I have already d...	Spanish	French
104	<Binary data>	ATtinah	ATrunan	yes, its nice	Hindi	Hindi
105	<Binary data>	AThilraham	ATiceouza	I am fine	English	French
106	<Binary data>	ATiceouza	AThilraham	what are you u...	French	English
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig 5 Functioning of receiving process

IV. Results And Discussion

In order to ensure efficiency of the system we tested the application with 50 clients connected to the server continuously for a period of sixty minutes. The client side application was installed in thirteen windows 8 enabled devices and the rest were emulated in the Windows 8 Phone emulator. The success rate of the application is depicted in the graph below in which

- CASE 1 – Successful users
- CASE 2 – Lost Connection
- CASE 3 – Slow Processing
- CASE 4 – Socket Disconnected

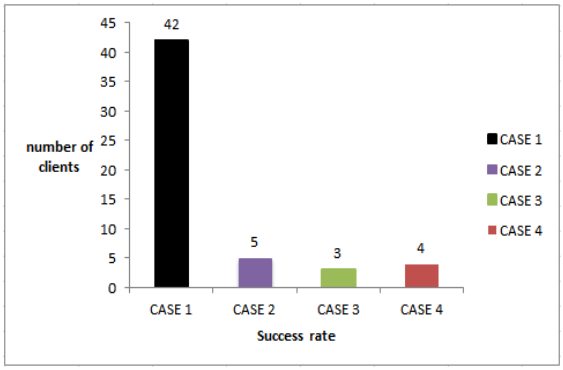

Fig 6 Success Rate

It was observed that out of the 50 clients connected thirty clients had a successful chatting experience with absolutely no flaws. 10% clients had connectivity issues due to the frequent internet disconnections. Another 6% client did not have a smooth conversation due to the slow processing time required to translate the message and display. While the remaining 4 clients could not establish a successful socket connection. This can occur if more than the threshold numbers of clients try to access the same process or operation simultaneously; however, this issue can be solved if the same request is tried again after waiting for a random interval of time. Thus we have achieved efficiency of 85%.

V. Conclusion

In this paper, we have presented a windows 8 phone app named ALLTALK ™ which supports cross language communication. It is a bridge between instant messaging apps and translator apps which improve the

communication ease between two users knowing different languages. Exchange of messages takes place in universally used languages like English, French, etc. where both the users know how to communicate in a common language. Thus chatting on mobile phones is a luxury when both the parties involved know a common language. This application enables the communication between two persons irrespective of the language each user wishes to use individually. It eliminates the need for switching between two applications to understand the messages that are exchanged.Thus we have achieved the integration ofthe two technologies of chatting and translating thus saving time and improving the quality of communication irrespective of the language known to the users. After testing the app we came to a conclusion that the efficiency of the application is 85%.In future this application can be used as a medium by service companies to communicate with their customers in their native languages, which is an effective way of providing customer service through instant messaging by embedding the chatting facility into their existing service software.

References

[1]. MChatterjee, S.,Abhichandani, T. ,Haiqing Li, TuIu, B. JongbokByun, *Instant Messaging and PresenceTechnologies for College Campuses , Network, IEEE, 19(3),* 2005, 4 - 13.
[2]. Peter Saint-Andre., *Streaming XML with Jabber/XMPP, Internet Computing, IEEE , 9(5),* 2005, 82 - 89.
[3]. D. Stenberg, "History of IRC (Internet Relay Chat)," http://daniel.haxx.se/irchistory.html, Mar. 29, 2011.
[4]. C. Yudkowsky, "Byte of Success: An IM Strategy," http://accounting.smartpros.com/x37234.xml, Nov. 10th, 2004.
[5]. http://www.tml.tkk.fi/Publications/C/21/nie_ready.pdf
[6]. Vanessa Wang, Frank Salim, Peter Moskovits, Building Instant Messaging and Chat over WebSocket with XMPP, *The Definitive Guide to HTML5 WebSocket,* 4 (New York: Springer-Apress, 2013) 61-83.
[7]. Jongmyung Choi, Chae-Woo Yoo,The Internet of Things ,Connect with Things through Instant Messaging, 4 (New York: Springer-Springer Berlin Heidelberg, 2008) 276-288.
[8]. Peter Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. IETF RFC 3920, Oct 2004.