Instant messaging application for Smartphone

Meruert Serik and Gulmira Beketovna Balgozhina

Gumilyov Eurasian National University, Merzoyana Srteet, 2, 010008, Astana, Kazakhstan

Abstract. The article proposes to develop an Instant messaging application for Smartphone that users can contact with friends and family by sending an SMS. Smartphone - software that will be able to implement certain elements of the educational process. Instant messaging application includes a convenient and effective set of tools to provide quick access to all information. They provide support for advanced features such as remote procedure call, group chat, file transfer, invisible mode.
[Serik M., Balgozhina G.B. **Instant messaging application for Smartphone**. *Life Sci J* 2014;11(1s):258-262] (ISSN:1097-8135). http://www.lifesciencesite.com. 47

Keywords: Instant Messaging, short message services, Multimedia messaging service, communication, Jabber protocol, Jabber server, Jabber client, user, Extensible Messaging and Presence Protocol.

Introduction

Instant Messaging (IM) is an Internet-based protocol application that allows one-to-one communication between users employing a variety of devices. Recently, Instant Messaging has already obtained the remarkable success as P2P communication tool. In some places, it already took the place of e-mail as the first choice for long distance communication. In the mobile area, the Short Message Services (SMS) and Multimedia Messaging Service (MMS) also attract thousands of subscribers by the richer and richer set of services. Jabber is the most widespread open source platform, using an XML encoded protocol, especially tailored to provide instant messaging and presence services over the Internet [1].

A messaging system needs a fast, real-time communication protocol to support real-time messaging capabilities, including instant messaging, conferences, alerts, news, polls, and file transfers. It should provide the reliability and availability expected by end users and the security necessary to protect privacy and information, and to meet regulatory requirements [2].

In this project we are trying to develop an Instant messaging application for Smartphone that users can contact with friends and family by sending an SMS. We choose the Jabber protocols [Jab] to base our work on. The key advantages of choosing Jabber is its open, standardized, secure, and extensible design. Due to the open design, Jabber protocols are verified and improved by hundreds of experts and developers. There are a great number of open source clients and servers available.

The XMPP extensions are defined by the Jabber community as Jabber Enhanced Proposals (JEPs) [3].

Main part.

Today, Instant Message is widely applied. By this kind of software, people who log online can get response in a short time after sending message which is a way to real-time communication and chatting[4]. For most people, it is cheaper than chatting by telephone; furthermore, the newest IM software integrated data transmission, voice chatting, video conference, e-mail and so on. Some analysts call IM as “real-time e-mail.” For enterprises, IM exploits new area for web-services. People approve IM software not only for fun and chat, but also for the excellent performance in business communication. The attraction in business is obvious to see: for one hand, it will tell the stuff weather other colleagues online, to prevent wasting time on telephone or informing conference and this kind of affairs. On the other hand, it provides a real-time dialog to catch the business opportunity, especially for the interaction between companies and their customers [5].

Now IM services have got wide world known for allowing people all over the globe to connect in almost real-time with anyone else in the world. In the market, there are ICQ, AOL, MSN, Yahoo Messenger, and a lot of similar applications.

Most of these applications rely on a central server that administers users and informs others about their presence and location, so a user can send directly messages to another user, if its location is already known.

The concept of presence has received substantial attention from the virtual reality community, and is becoming increasingly relevant both to broadcasters and display developers (picture 1).

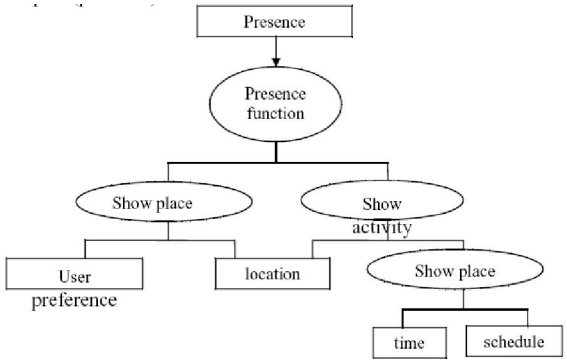

Fig.1. Presence Availability

The IETF published the Presence and Instant Messaging Model, RFC-2778 in Feb.2002. The model contains the several entities involved, a description of the basic functions they perform, and more relevant, the definition of a common vocabulary that can be used to facilitate discussions among different systems.

Let's take a look at picture 2, it's a simplified concept model the presence system collects and publishes presence information to users who interested in this information, including personal, application and services.

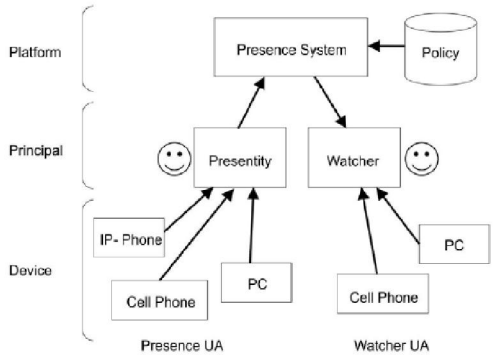

Fig. 2. Presence System Concept Model

The policy database controls the communication of the presence information by Buddy List or judging who and when can see the information. The presence service consists of two types of clients: Presentity and Watchers. The Presentity provides presence information to the system which is collected from several Presence User Agent (UA), as PC stared, phone is busy or cell phone without signal. The Watcher can either take the form of a fetcher or a subscriber. The fetcher simply requests the current state of Presentity from the presence service. On the other hand, a subscriber requests notifications from the presence service of future changes of some Presentity. There is a special kind of fetcher called Poller. The Poller "fetches" presence information on regular basis. The changes of presence information are distributed to subscribers via notifications. The presence service also keeps information about watchers and its activities in terms of fetching or subscribing to presence information. The presence service may also distribute watcher information to other watchers using the same mechanisms that are available for distributing presence information. The model defines the presence protocol as the interaction between presence service, Presentities and Watchers [6].

Jabber technologies use client-server architecture, not a direct peer-to-peer architecture as some other messaging systems do. It is almost identical to that of email. Each user has a local server which receives information for them. The various local servers transfer the messages among themselves for delivery to users. We could see in figure 4, the architecture is similar to the e-mail system where each client is connected to a local server and every communication from or to this client passes this server. User accounts and user data, such as contact lists and preferences, are stored on the local server. User identities are unique for each server by a unique address, like Alice@Jabber.org. Alice can not only contacts Bob who shares the same Jabber server A, but also exchanges information with other servers user Cindy or friend David different IM systems over the Internet through a component called Etherx, which takes care of translation between different servers or messaging system. A transport is a special server with the sole purpose of bridging from Jabber to other services (ICQ, AIM, MSN, etc.). When a user logs onto Jabber, a thread is created in the transport to handle all communications to and from that user. In addition, a separate thread is created in the transport for each service that the user is subscribed.

Every entity, for instance, a server, a component, a user connected with a client, is identifiable by a Jabber ID (or JID). Each Jabber ID contains a set of ordered elements as domain, node, and resource in the following format: user@host/resource or [node@] domain [/resource]. Domain name is the primary identifier. It represents the Jabber server to which the entity connects. The node is the secondary identifier. It represents the "user". Resource is an optional third identifier. Used to identify specific objects belong to a user. Resources enable a single user to maintain several simultaneous connections [7].

Consider the **Jabber server** that plays three primary roles:

- Handling client connections and communicating directly with Jabber clients
- Communicating with other Jabber servers
- Coordinating the various server components associated with the server

Jabber servers are designed to be modular, with specific internal code packages that handle functionality such as registration, authentication, presence, contact lists, offline message storage, and the like. In addition, Jabber servers can be extended with external components, which enable server administrators to supplement the core server with additional services such as gateways to other messaging systems.

Jabber client consists of components: All clients**,** which will be able to:

- communicate to the server through TCP sockets;
- parse well-formed XML;
- understand the Message data type;
- express presence (online/offline/unavailable) information to the server and understand incoming presence data, understand the Info/Query data type and have some preset queries such as logging in, rosters, searching, and setting user information [8].

Consider the Jabber Protocol and Client/Server Interactions.

The main protocol of Jabber is the IEFT XMMP (Extensible Messaging and Presence Protocol) Draft. It allows client-to-server and server-to-server communications. The different entities of the Jabber architecture pass data to each other using XML streams, which is essentially the exchange of data in the form of XML fragments in "streaming" mode over a network connection. Jabber manages to cover all the extensive needs in an IM system with these three basic elements, through clever use of XML namespace:

</message> (client to client conversation): protocol for message

</iq> (query message): protocol for presence feature

</presence> (update a client's availability): protocol for session establishment and roster management

And there are two special namespaces:

</error> (standard error-handling specification)

</x> (allows developer to extend the base protocol)

The following segments show a detailed description of communication between clients and servers in Jabber:

Authentication / Logging In

If a registered user wants to contact with other users, it must be authenticated at a server. When a request arrives, after checking the session for this user exists or not, the authentication or registration procedure starts. Jabber uses a one-way secure hash function (or, SHA1) to authenticate the user. This function creates a value by hashing the session ID and the user password. The server verifies that the correct hash value was returned by the client. User information with the correct authentication can be stored in a flat file or using a Lightweight Directory Access Protocol (LDAP) service.

Establishing session

After some pre-conditions are met, such as Stream Authentication and Resource Binding, the client tries to contact with the server. Once contacted, the client sends a Session Request message-an XML stream- to the server. If no error occurs, the server grants a session over an open socket, and informs the client that the session has been created. This XML stream is kept open during the lifetime of the session, and all communications between the client and server are sent and received over this socket.

Sending and Receiving Messages [9].

After a client has established a connection, it is able to send and receive messages to and from other clients.

The source client produces a XML-standard and addresses it to the destination client. The source client's server is in charge of delivering this message to the destination client (in case it is connected to the same domain server) or to forward this message to the destination's server. The main fields that a message includes are: Intended Recipient (obligatory), Message Type, Message Body, Message Subject and Conversation Thread.

Rosters

In Jabber, one's list of contacts is called a roster. A roster is stored by the host so that a user may access roster information from any connected resource. It contains subscription information for the client's account, including the user's nickname and contact list. It is shared between the client and server. Roster management includes the following 3 basic aspects:

- ✓ Receiving One's Roster on Login — upon connecting to the host, a node should request the roster (however, because receiving the roster may not be desirable for all resources, e.g., a connection with limited bandwidth, the node's request for the roster is optional.
- ✓ Adding a Roster Item — at any time, a node may add an item to its roster. The host is responsible for updating the roster information in persistent storage, and also for pushing that change out to all connected resources for the node. This enables all connected resources to remain in sync with the host-based roster information.
- ✓ Deleting a Roster Item — at any time, a node may delete an item from its roster. Note: as with adding a roster item, when deleting a

roster item the host is responsible for updating the roster information elsewhere.

Subscriptions

A subscription is a request to send and/or receive presence information from contacts each time you log onto [10].

Jabber. Presence subscriptions are managed by the server and are stored in the roster. When a user logs onto the Jabber server, it sends an update on that user's presence to all of the people on the user's subscription list.

There are 5 basic Subscription States:

2. None
3. To - the User is subscribed to the Contact's presence (only)
4. From - the User is an "observer" in the Contact's roster (only)
5. Both - Both the Jabber User and the Contact are subscribed to each other's presence
6. Initial - neither has knowledge of the other's existence (or state in the other party's roster)

Presence

1) Presence update;
2) Presence subscription management;
3) Status: free-form text describing a user's presence (i.e., gone to lunch).

We created activity flow diagram – "Communication between clients and servers", which is located below.

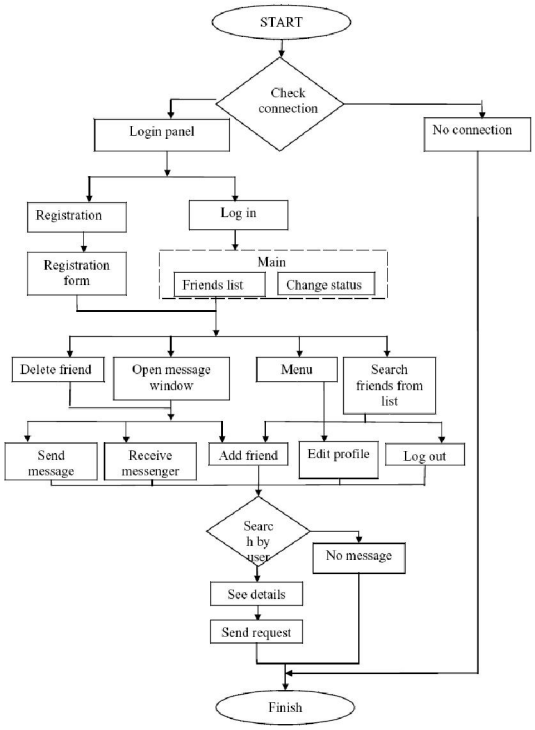

Fig. 3. "Communication between clients and servers"

Conclusion.

Thus, IM systems have become very popular over the past years and the number of people who use ICQ, AIM or the Microsoft Messenger has risen enormously. Now, instant messaging services are going to the mobile domain. The basic benefit of mobile instant messaging is still the "see before you connect" aspect, but mobility adds a number of features to make it even more attractive to the user. Along with the progress of wireless transmission and the volatile increase of wireless network users, the demands of using handset and combining the presence status of friends online to delivery message on the mobile network as well as using handset and IP network to execute over-platform integrated information transfer are all emerged gradually.

The most attractive part of Jabber is its open source platform. The Jabber protocols are free, open, public, and easily understandable. It offers support for interoperability Security and Scalability. Meanwhile the client implementation is Simple. But the disadvantage is its heavy XML coding and not so widespread and popular. It is more or less a concept but without powerful occupation of the market.

Corresponding Author:
Dr.Balgozhina
Gumilyov Eurasian National University, Merzoyana Srteet, 2, 010008, Astana, Kazakhstan

References

1. Introduction to Instant Messaging Software. Date Views 01.07.2013 www.docs.sun.com/source/819-0063/im-intro.html
2. Rochus Schreiber and Ragnar Eggen. Why wireless instant messaging may be the key to migrating mobile customers to 3G service. Date Views 04.07.2013 www.accenture.comhttp://developer.android.com/
3. Jabber Technology Overview. Jabber Software Foundation. Date Views 05.07.2013 www.jabber.org
4. Wikipedia. Date Views 04.07.2013 www.en.wikipedia.org/wiki/
5. Mallick, M., 2013. Mobile and Wireless Design Essentials: Wiley – India, pp: 454.
6. Rogers, R., J. Lombardo, Z. Mednieks and B. Meike, 2013. Android. Application Development: Programming with the Google SDK. O'REILLY, pp: 318.

7. Serik, M., N.T. Chyndaliev and N.F. Musina, 2012. Work with system commands. USA: International Center for Education and Technology, pp: 130-131.
8. Serik, M., N.T. Chyndaliev and N.F. Musina, 2012. Teaching methods of the BIOS. USA: International Center for Education and Technology, pp: 131-132.
9. Serik, M., A.K. Alshanov, N. F. Musina and K.R. Yesmakhanova, 2010. The realization of open remote metods of education. Comparative education, Teacher Training, Education Policy, School Leadership and Social Inclusion. Sofia, pp: 471-472.
10. Serik, M., A.K. Alshanov, N. F. Musina and K.R. Yesmakhanova, 2010. In education-didactic system as means of improvement of quality of education. Voronezh Scientific Conference, pp: 471-472.

1/22/2014