International Journal of Information Technology, Vol. 15 No. 1, 2009

A Jabber Based Framework for Building Communication Capable Java Mobile Applications

Mohy Mahmoud, Sherif G. Aly, Ahmed Sabry, Mostafa Hamza

Department of Computer Science and Engineering
The American University in Cairo
P.O. Box 74, New Cairo, 11835, Egypt
sgamal@aucegypt.edu

Abstract

The main aim of this framework and library is to provide Java Micro Edition developers a powerful mechanism that facilitates the process of building communication capable mobile applications. The framework enables mobile devices to abstractly exchange streams of data using an underlying communication infrastructure, and also supports the continued connectivity of mobile devices as they roam into different physical locations. To achieve this, the famous Jabber protocol was initially ported to mobile devices, and supporting functionality that monitors the roaming behavior of mobile devices was built as part of a framework and supporting library. Eventually, in order for communication streaming to be successful between roaming mobile devices, such devices must be connected to the Internet currently through either Wireless Fidelity (WiFi) or Bluetooth, and may be easily extended to use other technologies such as WiMAX, GPRS, and 3G. In this article, we describe the porting of the Jabber protocol to mobile devices, building of the communication framework, codenamed BEEP[1], and associated library in detail, and describe some sample applications that have been built using the framework.

Keywords: Mobile, Jabber, Roaming, Java, XML, Framework.

I. Introduction

Great achievements have been made in the field of mobile computing. Mobile devices recently experienced great leaps in their computing and communications abilities. Users can now communicate using their devices between almost every single inhabited point on the planet. The introduction of the Global Service Mobile (GSM) and the Internet was considered the spark of the new age of telecommunications. Nowadays, GSM supports the Wireless Application Protocol (WAP) and Enhanced Data rates for Global Evolution (EDGE). The third generation protocols of the mobile industry support higher data rates that can be measured in Mbps and that can be used for bandwidth hungry applications such as video conferencing and broadband Internet browsing. Other technologies such as

[1] Implementation of the framework can be found at www.jabber.org

WiFi and Bluetooth have also contributed to better low range communication abilities for mobile devices. Not to mention the unprecedented advanced in the computing ability of mobile devices and the enhanced utilization of valuable resources such as memory and battery power, a plethora of applications that once have been a dream have now become a reality on mobile devices.

According to the statistics done by the GSM association there are at least one thousand new users that purchase mobile devices around the world every minute. Such statistics indicate the dissemination of mobile devices into our daily lives in a way that has already allowed a very significant percentage of our populations to be equipped with valuable computational and communication abilities everywhere they go. An era of mobile computing research of the underlying infrastructures, frameworks, and applications that support mobile devices is now a true reality.

In this research, we introduce a framework and library that allows mobile devices to identify the location of users, and stream data amongst one another in a standardized fashion using the Internet cloud, transparently utilizing existing communication technologies such as WiFi and Bluetooth, and can be easily equipped to support any other similar communication technologies that currently exist, or that may emerge in the near future. Such technologies that currently exist include WiMAX, 3G, and GPRS. However, the usage of low range transmission technologies such as WiFi and Bluetooth facilitates the process of identifying the location of users to a proximity of 10-100 meters in the case of Bluetooth, and around 70 meters in the case of WiFi. Cell based location identification techniques can be use in the case where GPRS and 3G are used, that, which needs support by the cell infrastructure itself as opposed to Bluetooth and WiFi that do not need any such infrastructure. Thus, the usage of Bluetooth and WiFi is considered rather significantly advantageous for the location identification portion of the framework itself.

Furthermore, the framework facilitates the process of building applications that require data communication between mobile devices that continue to roam from one physical location to another, without the applications themselves having to keep track of the location of users. Such applications include instant messaging, file transfer, video and audio conferencing, white boards, and online gaming.

This paper is organized as follows: Section 2 will give an overview of related research work, section 3 discusses the framework used and gives the detailed description of the specification based protocols, section 4 concludes the article, section 5 lists the acknowledgements, and finally, the references are listed in section 6.

II. Related Work

Instant Messaging (IM) started commercially in 1985 by Quantum Computer Services, America Online nowadays. The company created the free, standalone AOL Instant Messenger (AIM) application [11]. This application contains the list of contacts and the ability to send text messages to them. The next step was taken during the 1990s by a small company called Mirabilis. The application ICQ (I seek you) built by that company gained millions of subscribers. Yahoo's Messenger and Microsoft's MSN Messenger share a place between instant messaging applications that are widely used with some advanced features [11]. Nowadays, some big organizations such as Compaq Computer and United Airlines are beginning to use instant messaging as a business communication

tool. Other e-commerce firms use instant messaging for establishing a real time session with their customer service representatives [11]. Instant messaging typically does not provide persistent message storing, inboxes, and file attachments. They are not designed for complex collaborative work requiring real-time services such as video conferencing or Internet telephony.

A. Instant Messaging Protocols and Standards

Two protocols were defined for IM in the past years, SIP/SIMPLE and Wireless Village.

i. SIP (The Session Initiation Protocol)

SIP, defined by the Internet Engineering Task Force (IETF), is a standard signaling protocol used for setting up, controlling and terminating "interactive communication sessions" between two or more contributors [21]. Simplicity and decentralization are the key features that make SIP the choice of next generation services [16]. SIP "Is an application layer text based peer to peer protocol based on the HTTP communication layer" [5]. It can also support multimedia sessions and telephone calls. The SIP architecture is designed to be independent of the underlying transport layer. It can run on TCP, UDP or SCTP. SIMPLE is defined as a set of extensions to SIP to support IM and presence. Three methods can be added to support instant messaging SUBSCRIBE, MESSAGE and NOTIFY.

ii. The Wireless Village Protocol

The Wireless Village protocol or WV is standardized at the Instant Messaging and Presence Services (IMPS) working group of Open Mobile Alliance (OMA). It is divided into two components WV server and client. It supports five main services: The Presence Service, Instant Messaging Service, Group Content Service, Shared Content Service and Service Access Point.

The difference between WV and SIP/SIMPLE is that WV is dedicated to 2G and 2.5G communication networks only while the latter can fit well in 3G IP-based communication ones [5].

iii. Presence

Presence is a protocol that allows different IP entities or devices over the network to communicate with each other. It depends on the publish and subscribe (pub/sub) concept as the entity publishes its availability information over the network. Presence provides a means to accept and store such data associated with each IP entity or device over the network [17, 18]. It can have multiple resources for the same entity i.e. the desktop, laptop, IP phone, email or IM client represent multiple resources for the same such. The management of these resources is quite challenging. The Jabber protocol provides a multi-presence protocol to tackle this multiple resources issue [17]. Although presence is no more than an on/off property of the entity across the network, it extends such property to become the real-time state of this entity. For example, the presence properties include the location, direction, multimedia, –Audio or Video- played at the current state and even the mood of the entity. Presence is more into creating a virtual world that resembles this entity [17].

Presence represents one part of the communication revolution. Presence needs a way to know the capabilities of the entities over the network. For example, there must be sufficient information about if the entity has a video camera or microphone attached to it. The capability information can be obtained from a service discovery extension to the Extensible Messaging and Presence Protocol (XMPP). However, this service discovery needs to be more dynamic especially if the application is "poll" demanding [18]. It also needs a data exchange protocol to benefit from the presence information provided. Hence, the Instant Messaging shows up to deliver messages to the users based on their presence status. Furthermore, throughout the last few decades, Internet users have been acquainted with Instant Messaging (IM). Using this trust between the users and the IM protocols, XMPP managed to convert simple IM to real-time communication. Using standard XML name spaces make this protocol compatible with XML data exchange formats. The XML data forms that the XMPP protocol offers is light enough to be delivered at real-time [17].

iv. Jabber

Jabber is based on a core transport layer of streaming XML. It is an alternative to closed Instant Messaging and presence services [9]. The streaming technology has been formalized within the IETF as XMPP. It is a pure XML signaling channel that can be utilized to serve applications beyond IM [8]. The XMPP RFCs (3290 – 3923) provide the standards for near-real-time messaging and structured data exchange [9]. The advantages behind Jabber are that it is open, standard, proven, decentralized, secure, extensible, flexible and diverse. The Jabber protocol is founded on the SIP standard to deliver IM solutions. The attributes of XMPP that jabber uses, allows jabber to be robust in developing and deploying large scale applications [5]. Because Jabber is based on XML, it enables building custom functionalities on top of the protocol [7, 11]. Developers could easily use XML to create new tags for the representation for upcoming introduced formats they want to support.

XMPP uses XML streams and stanzas not documents. XML streams are the containers for the data exchange between two entities over the network while the XML stanzas are the first-level child elements sent over those streams [9]. XMPP defines three core stanza types which are:

- **The <message /> stanza:** It is used for sending messages from one entity to another. XMPP is optimized for instantaneous delivery because the delivery of messages is dependent on presence. There is also support for offline messages, if the recipient is not online.
- **The <presence /> stanza:** It is a mechanism for publishing the availability status over the network of an entity. Such availability status could be presented in either the entity is online, away, busy or any other customized status. In IM applications, such status goes to everyone in the user's contact list.
- **The <iq /> stanza:** it is the info/query stanza and it is a request-response mechanism that facilitates making requests of and receiving responses from other users.

Clients, servers and components identify themselves with the Jabber Identifier or JID, similar to the email identifier. JID consists of three components: A username, a host and a resource. The host is mandatory for connection while the resource and the username

are optional. The host should be a valid DNS hostname. The username is concatenated to the host with the symbol '@'. This enables any clients in different jabber domains communicate with each other [7].
XMPP is not responsible only for managing streams and routing stanzas. This is the minimal XMPP implementation. But for having a good XMPP implementation, it should contain more features [9]. A Jabber server should support the following:

- Transport Layer Security (TLS) and this is for channel encryption.
- Simple Authentication and Security Layer (SASL) and is used for adding authentication to existing connection-based protocols.
- DNS SRV for port lookup.
- Various profiles for addresses that contain Unicode characters.
- Unicode Transformation Format 8 (UTF-8) for entirely internationalized stream content.
- An XMPP protocol for binding the resources to streams for network-addressing uses.

More core services must be available also for having an IM Jabber server such as IM session management and contact-list storage.

v. Jingle

Numerous extensions can be built on top of Jabber's streaming layer. Most of the protocol extensions that have been done until now are for supporting textual messaging. Through the Jabber community, developers are interested in having extensions for Jabber to support multimedia communications [8]. Jingle is one of the developed solutions for video, voice and file sharing as an extension for Jabber. It is a protocol that powers the XMPP protocol with the ability to stream different multi-media sessions. Jingle uses the XMPP standard protocol for the signaling initiations and negotiations. After the session is authenticated, the media will be out of the XMPP band. It will be through the UDP connection and management. Although Jingle is still an emerging technology, it has been successfully deployed as its first release at Google Talk for voice and file transfer since 2005 [8].

B. Applications and Frameworks

i. Wireless Mobile IM

Mobile Instant Messaging (MIM) is the next global messaging market opportunity and it will be the expected evolution of the Short Messaging Services (SMS). According to the estimates done by International Data Corporation (IDC), around 1.2 billion mobile users around the world are using SMS. By the end of 2010, global wireless messaging will have more than 1 billion subscribers added to the current users. According to surveys done by IDC, end users prefer text messages rather than initiating a voice call that may need more time and could potentially add more delay. They prefer that the message could be read later rather than responding to a message call even if the message is delayed. MIM will be more advanced to the extent that mobile users will typically turn to it instead of SMS. MIM is the first presence-enabled wireless applications that allow mobile users to initiate one-to-one communications [16]. MIM allows for real-time

communication and it is far away from the delay associated with email and SMS, and also enables mobile users to have a contact list for easy communication. Moreover, it can support communications beyond voice without the need for 3G and can be offered over presented 2G or 2.5G [16]. Internet enabled mobile devices started to have their own implementation of IM applications. Microsoft extended MSN Messenger to be used with the MSN Mobile Service. It uses WAP as the connection for sending and receiving messages and presence [11]. Yahoo also released versions specific for mobile devices for Yahoo Messenger IM. Such applications are for Windows CE based mobile phones, PalmOS and WAP enabled handhelds.

ii. SJSU Mobile Jabber System

The SJSU Mobile Jabber System is an approach for developing a wireless-based text chatting [5]. It enables mobile phone users to exchange text messages. The implementation of SJSU Jabber IM is developed using the jabber protocol and wireless technologies, including JME. SJSU Jabber IM supports the following functionalities: registration, authentication and login, roster view, roster management, one-to-one and one-to-many text chatting.

The SJSU Jabber IM architecture is a 4-tier client-server architecture, having the presentation layer as the first tier, which contains a JME-based client for JME enabled devices, and Internet-based client for users to complete web based membership registrations. The middleware layer includes a wireless server besides a web server. The former is for wireless communications between wireless clients and text chatting system, and the latter is for internet-based membership and administration functionalities. The application layer includes the SJSU jabber IM server for text chat sessions between mobile devices. Finally, the data store layer logs and handles all IM messages going back and forth between the users.

iii. "i"Mobile EE – An Enterprise Mobile Service Platform

iMobile EE is a mobile service platform that allows mobile devices to communicate together and access services. Those services can be normal browsing, IM, sending SMS and remotely control multimedia contents.

The architecture of iMobile EE is based on three main components: Devlets which provide interfaces and drivers to the various mobile devices, Infolets which are responsible for creating the abstraction of information using a specific protocol and Applet which handles the post-processed information gathered by infolets. Moreover, the core of iMobile EE is the proxy engine that hosts the devlets, infolets and applets. The proxy engine main functionality is to synchronize between the major three components.

iv. Google Talk

Google Talk is a simple free way to talk and send instant messages from any Gmail account to any other Gmail account. Google talk users sign in with their Gmail username and password. Google Talk is a famous Jabber client that supports the basic XMPP protocol plus some extensions of the protocol. It utilizes the Gmail server, which is one of the well-known Jabber servers available free of charge to any Jabber client. It is a desktop application and there is no version available for limited resource devices.

The Beep Framework, provides the basic XMPP protocol that Google Talk provides, it also uses the Gmail server. Furthermore, the Framework provides a basic foundation to the developers to create any type of application. Another advantage that the Framework has is that it supports roaming, which is not supported and not needed in desktop applications but rather in limited devices.

v. Smack

Smack is another open source Jabber client. It provides the basic XMPP protocol with some extensions of the protocol. It is a platform independent application utilizing the power of Java. It could utilize any Jabber server, with the ability to use SSL. However, it works on JSE working as a desktop application.

The Beep Framework is very similar to smack in the architecture, providing the basic XMPP protocol. However, it has the advantage of being a JME based application that supports Roaming for mobile devices.

vi. MGTalk

MGTalk is a Jabber client for the JME MIDP 2.0 platform that supports connectivity with the Gmail server allowing registered users to chat with their Gmail contacts and also retrieve their email messages from the Gmail server. Although supporting email popping from the Gmail server can be a further enhancement in Beep framework, it supports connectivity to any Jabber server not only the Gmail server. Another important difference is that Beep is a library that can be used to base Jabber clients on, not an application that implements Jabber Protocol.

vii. PLIM

Context-awareness is one of the new paradigms in the field of mobile industry. The current state and the activity of the user are retrieved to be broadcast over a network. In the current instant messaging system, the presence context information is tied with the messaging functionality [22]. On the other hand, the concept of Presence, Location and Instant Messaging or (PLIM) is to a use publish-subscribe mechanism for distributing context information between services available and users in a network.

A scenario for context-awareness, by the combination of Presence, Location and Instant Messaging or (PLIM) with a calendar/scheduling service, presence will be automated to be changed automatically into “in a meeting” once the user enters a meeting room. Telematica Instituut developed the PLIM framework that focuses only on location and presence [22]. It works over a Bluetooth medium or “location provider” for indoor location retrieval.

The Location and presence modules are loosely coupled with indoor position information. A PLIM server is an extended Jabber server for handling the users and the services. The Beep Framework is similar in that as it uses a Jabber server, but does not only use Bluetooth. It supports Bluetooth, WiFi and is extensible that it can accept any communication layer added.

III. The BEEP Framework and Library

In this section, we present details of the BEEP framework and associated library that supports data exchange between mobile roaming devices. For this purpose, we initially present some usage scenarios of the framework to clarify understanding, and subsequently show how the Jabber protocol had to be initially ported to mobile devices for the purpose of supporting data exchange.

A. Usage Scenarios of the Framework

To recap, Jabber is considered to be an open and secure technology that facilitates instant messaging, as well as a set of standards for real time communication. This framework integrates location awareness, features of the Jabber data communication, as well as applications that need both location awareness, and data communication. For example, social networks such as Fracebook and mySpace can benefit from the location awareness features provided by this framework as well as the messaging abilities to incorporate the ability of social networking users to send data to roaming mobile users. As users roam around, the location awareness system updates the BEEP framework presence feature with the instant changes of the user's location. Subsequently, those updates can be used to update the user presence information in social networks, and eventually exchange data. Such data could be in the form of instant messaging, file transfer, or even real time voide and video communication.

To demonstrate another example, the framework can be used to identify the location of employees walking around the premise of a given organization. If one of the users moves from one room to a meeting room, the location awareness portion of this framework shall detect this change of the user's location and subsequently update the BEEP framework with such change. Consequently, the framework can change the presence status of the user from "available" to "busy" automatically with status message indicating the presence of the user in a meeting. Furthermore, the BEEP Framework can easily construct a communication layer between itself and any other social network and update this social network with the new status of the user as being busy with a status message in meeting. Futuristically, the framework can be extended to cover all features provided by Jabber protocol, such as conferencing, Collaborative editing, calendaring, gaming, home automation, Ecommerce, and customer services.

B. The Need to Port the Jabber Protocol to Mobile Devices

The Jabber Protocol allows clients to exchange data via XML streams with single or multiple servers. Clients initiate the XML streams while communicating with the server. On the other hand, communication between servers is achieved by routing the elements of the protocol over an XML stream from one server to another.

The message, Presence and Info/query Protocols represent the backbone of the Jabber Protocol. The Jabber Protocol supports exchanging text messages between two clients, a client and a server, or between two servers via the message protocol where instant messages are sent in the form of an XML stream.

The Presence protocol signifies the status of any Jabber user to be either available or not available. The Available state indicates that this Jabber user is currently online and ready to receive data. A user is not eligible to receive any data if the Presence status of the user

is set to be unavailable. The Info/Query Protocol (IQ) allows the Jabber entities to pass XML-formatted queries to the server and receive the responses back.

C. The BEEP Architecture

In order to build the functionalities and features supported with Jabber protocol on mobile devices, the architecture of the Beep framework was constructed as shown in Figure 1. The architecture is described below.

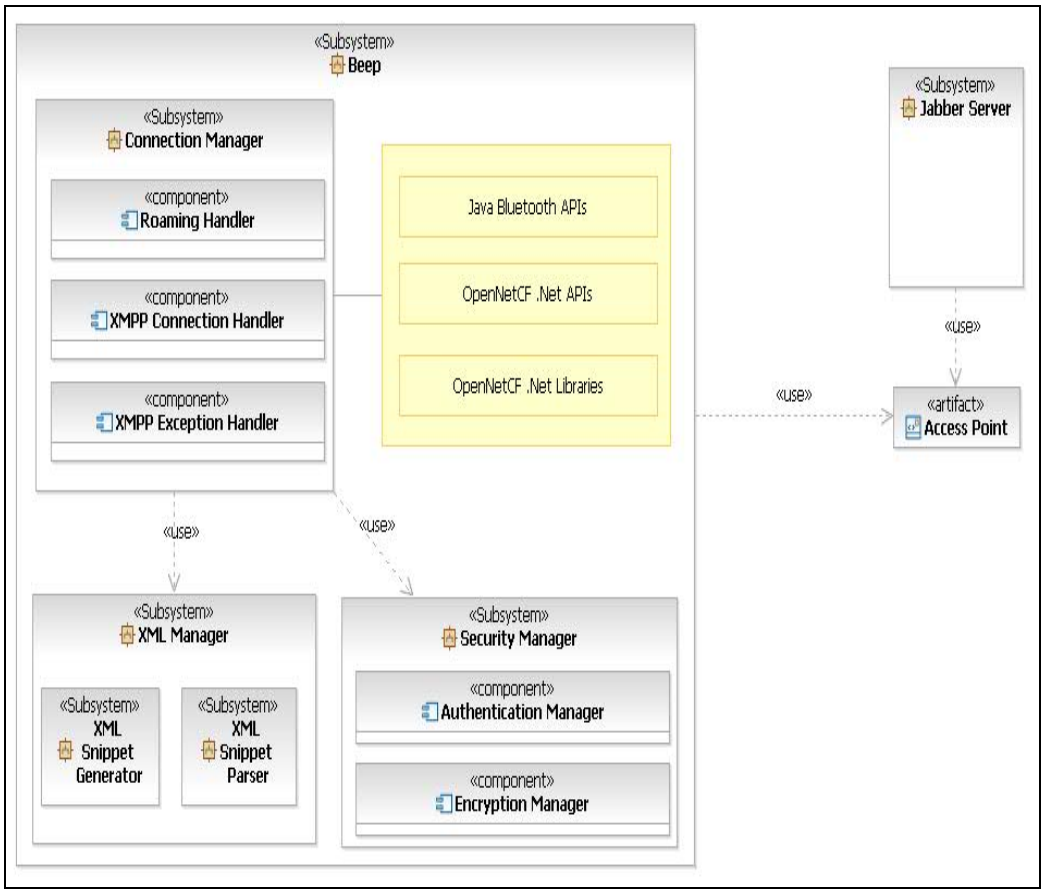

Figure 1 The BEEP Architecture

The Beep Subsystem shown in the above figure represents the Jabber client in a Jabber oriented environment. It includes all the subsystems and components that reside on the mobile device (client).

i. The Connection Manager Subsystem

This subsystem is responsible for all the work related to connectivity, whether it is about establishing and managing XMPP connections or about discovering the available wireless media within range.

- **The Roaming Handler Component:** The roaming component frequently checks for the availability of the connection, in case the connection is lost, it discovers

nearby posts and connect to the nearest post available. Then a reconnection signal is sent to the XMPP connection, which will send the new location of the user to the server, which will update the location of the user.

- **The XMPP Connection Handler Component:** This component is responsible for establishing and managing all types of XMPP connections that are done between the client and the server.
- **The XMPP Exception Handler Component:** This component represents the exception handling mechanism that will handle all XMPP-related exceptions and recover from the error.

ii. The XML Manager Subsystem

This subsystem is composed of the following:

- **The XML Snippet Generator Component:** This component is responsible for generating different XML snippets exchanged between different Jabber entities. In order to extend the snippets that can be sent to the server and include more types of snippets, the mechanism of generating the XML snippet for a specific packet is built-in the implementation of the packet itself.
- **The XML Snippet Parser Component:** This component is responsible for parsing different XML snippets exchanged between different Jabber entities. It depends on using a set of XMLPull parsing APIs

iii. The Security Manager subsystem

This subsystem is composed of the following:

- **The Authentication Manager Component:** This component is responsible for offering different authentication mechanisms that are supported in Jabber protocol. It supports SASL authentication and TLS authentication with the Jabber server.
- **The Encryption Manager Component:** The Encryption Manager component is responsible for encrypting the XML snippet exchanged between the Jabber client that resides on the mobile device and the Jabber server. It includes classes that support hashing techniques namely: RSA-MD5, it is intended to support other algorithms in the future.

D. BEEP Packages

The aforementioned main subsystems were decomposed to the packages described in the subsections below. Some of the class diagrams that can be feasibly presented will be shown below.

i. The Connection Package

This package as shown in Figure 2 contains all the classes that are used in dealing with XMPP connections. It interacts with the roaming package in order to create connections over one of the available wireless media: Bluetooth, Wi-Fi, and any other communication technologies that may be added in the future.

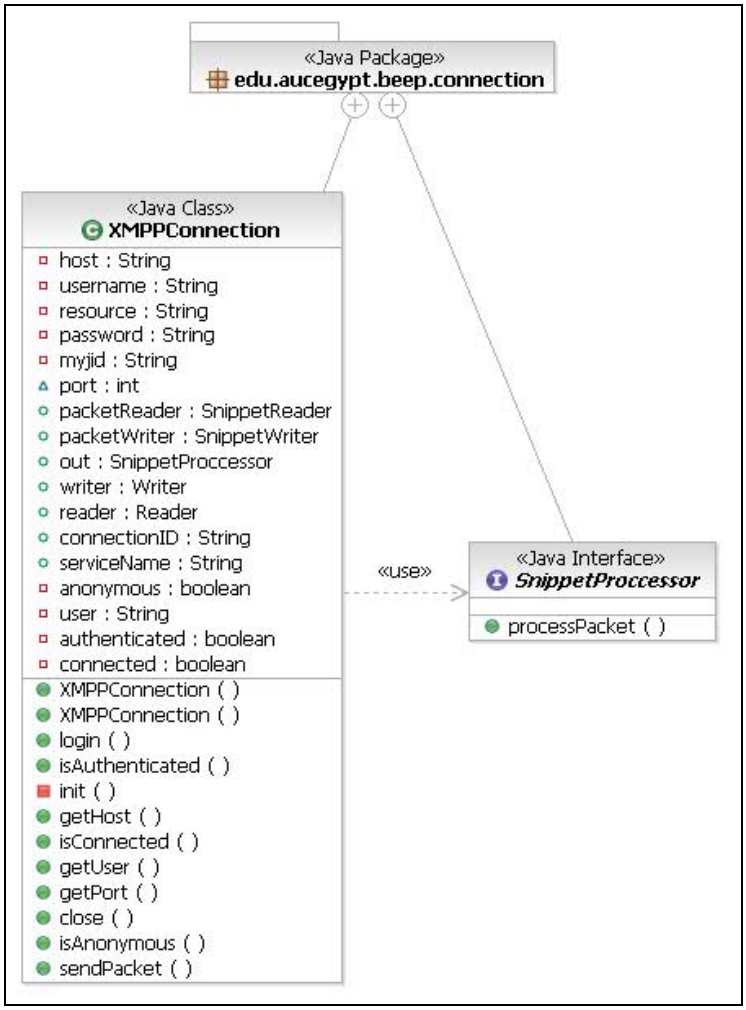

Figure 2 The Connection Package

- **The XMPPConnection class:** This class is primarily responsible for creating an XMPP connection between the user's defined application and any other Jabber server. Also, XMPPConnection uses the SnippetProcessor interface to be able to process the sent XML snippets from the snippet reader. SnippetProcessor is overridden within the application with the appropriate classes, such as chat, group chat...etc

ii. The Snippet Package

This is the core package of the Beep framework. It contains all the classes that represent different Jabber XML snippets that are used in the interactions between different Jabber entities. In addition, it includes two important classes: SnippetReader and SnippetWriter that are used for exchanging the generated snippets.

- **The Authentication class:** The class Authentication represents an IQ snippet that can be used to login to an XMPP server, as well as discover login information from the server. By default, the packet will be in "set" mode in order to perform an actual authentication with the server. In order to send a "get" request to get the

available authentication modes back from the server, change the type of the IQ packet to "get".

- **The Bind class:** The class Bind represents an IQ snippet that is used to bind a user to a resource and to obtain the JID assigned by the server. There are two ways to bind a resource. One is achieved by simply sending an empty Bind packet where the server will assign a new resource for this connection. The other option is to set a desired resource, but the server may return a modified version of the sent resource.
- **The DefaultPacketExtension class:** This class is the default implementation of the PacketExtension interface which provides a very simple representation of an XML sub-document. Each element is a key in a Map with its character data (CDATA) being the value.
- **The IQ class:** This class represents the base IQ packet. IQ packets are used to get and set information on the server, including authentication, roster operations, and creating accounts.
- **The IQType interface:** The IQType is an interface to represent the type of the IQ packet. The types are GET, SET, RESULT, and ERROR.
- **The Message class:** This class represents XMPP message packets. A message can be one of several types: NORMAL, CHAT, GROUPCHAT, HEADLINE and ERROR.
- **The MessageType interface:** MessageType is an interface to represent the type of the Message packet.
- **The Packet class:** Is the base class for XMPP packets. Every packet has a unique ID (which is automatically generated, but can be overridden). Optionally, the "to" and "from" fields can be set, as well as an arbitrary number of properties. Properties provide an easy mechanism for clients to share data.
- **The PacketExtension interface:** Is an interface to represent packet extensions. A packet extension is an XML subdocument with a root element name and namespace. Packet extensions are used to provide extended functionality beyond what is in the base XMPP specification. Examples of packet extensions include message events, message properties, and extra presence data.
- **The Presence class:** Represents XMPP presence packets and implements the PresenceType interface. The type of Presence packets are either:
    - AVAILABLE
    - UNAVAILABLE
    - SUBSCRIBE
    - SUBSCRIBED
    - UNSUBSCRIBE
    - UNSUBSCRIBED
    - ERROR
- **The PresenceMode interface:** Is an interface to represent the mode of the Presence packet. The modes are:
    - AVAILABLE
    - CHAT
    - AWAY
    - EXTENDED_AWAY

  - DO_NOT_DISTURB
  - INVISIBLE
- **The PresenceType interface:** Is an interface to represent the type of the Presence packet.
- **The Registration class:** Is a class that represents registration packets. XMPP servers may require a number of attributes to be set when creating a new account. For example the username, user's first name…etc.
- **The Roster class:** Is a class that represents XMPP roster packets.
- The RosterItem class: A roster item, which consists of a JID, their name, the type of subscription, and the groups the roster item belongs to.
- **The RosterItemStatus interface:** Is an interface that represents the subscription status of a roster item. It is an optional element that indicates the subscription status if a change request is pending.
- **The RosterItemType interface:** Represents the subscription type of a roster item.
- **The Session class:** Represents the session between the class and the client. If a server supports sessions, it must include a session element in the stream features it advertises to a client after the completion of stream authentication. Upon being informed that session establishment is required by the server the client must establish a session if it desires to engage in instant messaging and presence functionality.
- **The SnippetReader class:** Listens for XML traffic from the XMPP server and parses it into packet objects.
- **SnippetWriter class:** Writes packets to a XMPP server. Packets are sent using a dedicated thread.
- **StreamError class:** Creates a new instance of StreamError. An instance of this class is created when an error occurs in the XML stream open between any two Jabber entities.
- **XMPPError class:** Represents an XMPP error sub-packet. Typically, a server responds to a request that has problems by sending the packet back and including an error packet. Each error has a code as well as an optional text explanation. Typical error codes are Description, Redirect, Bad Request and many others.

iii. The Security Package

This package is one of the most critical packages of the framework. It is used heavily during the process of exchanging XML snippets between different Jabber entities. The package includes a set of classes that serve the purpose of securing data sent over the network between Beep framework and a XMPP server.

- **The DigestException class:** This class is responsible for throwing exceptions in case any error happens within the digest process.
- **The GeneralSecurityException class:** This class is responsible for raising a General Security Exception for general exception used for security purposes.
- **The MD5State class:** This class represents a Fast implementation of RSA's MD5 hash generators in Java. In addition, it contains the internal state of the MD5 class.
- **The MessageDigest class:** Is used to generate message digests using RSA's MD5 hashing techniques.

- **The MessageDigestImpl class:** This class is an implementation of the abstract class Message Digest.
- **NoSuchAlgorithmException class:** This is class represents the exception that will be thrown when an unsupported hashing or encryption algorithm is being used. (Note that currently we do only support MD5, other mechanisms may be added in the future).

iv. The XML Package

This package contains the classes dealing with XML stanzas exchanged between Jabber entities. It contains one class XMLParser that is used in parsing all types of XML snippets. Figure 3 demonstrates the package.

- **XMLParser class:** This class is the XML parser that is used to parse any XML snippet received from the server.

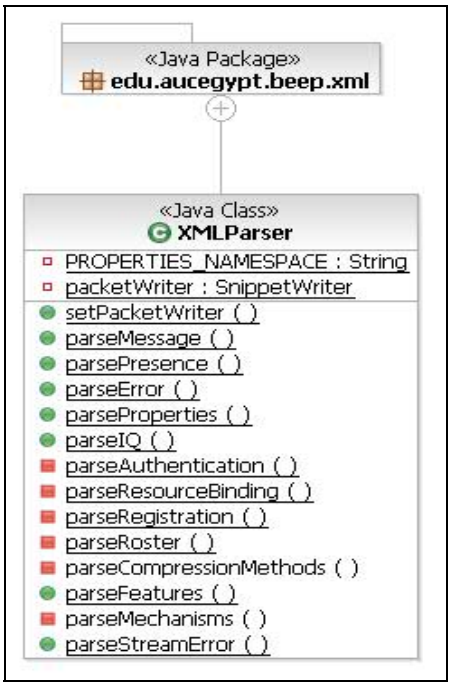

Figure 3 The XML Package

v. The Roaming Package

This package as shown in Figure 4 contains the classes providing the service of Wireless media discovery and management. The package empowers the Beep framework with the ability to roam between two access points or more without loss of data.

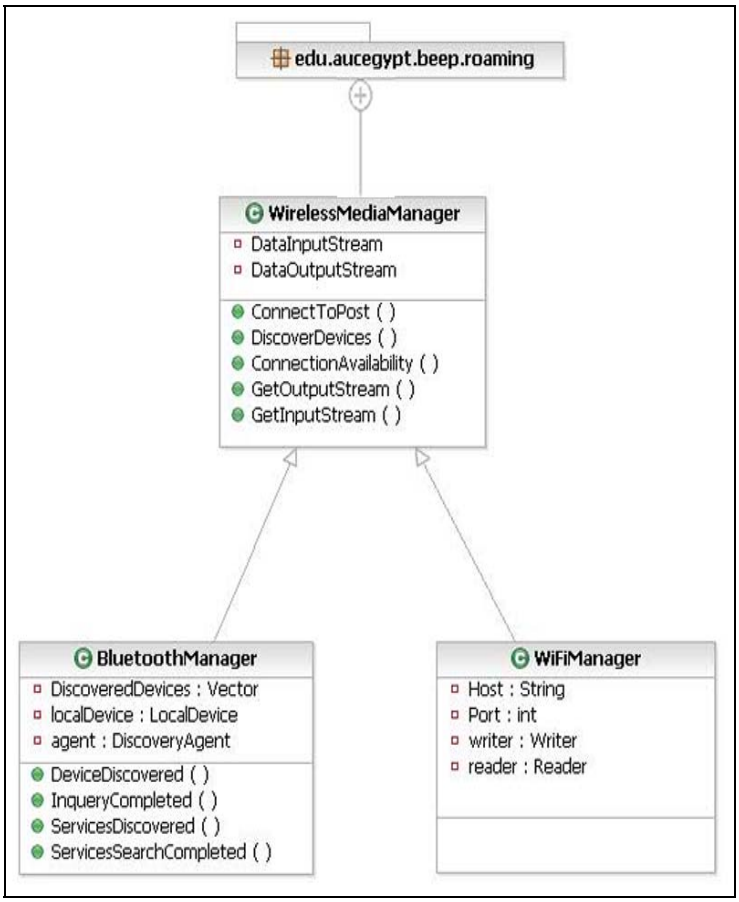

Figure 4 The Roaming Package

- **WirelessMediaManager class:** This is the base class for all wireless media managers. Any other class that inherits this class will be able to register itself as a wireless media manager in the XMPPConnection class.
- **BluetoothManager class:** This is the class responsible for managing Bluetooth discovery and management mechanism offered by Beep framework.
- **WiFiManager class:** This is the class responsible for managing Wi-Fi discovery and management mechanism offered by Beep framework.

vi. The Exception Package

This package as shown in Figure 5 defines the appropriate exceptions that are used in the Beep framework in order to handle any thrown exception during run-time. Thus, enhances the defect containment mechanisms used to ensure higher software quality.

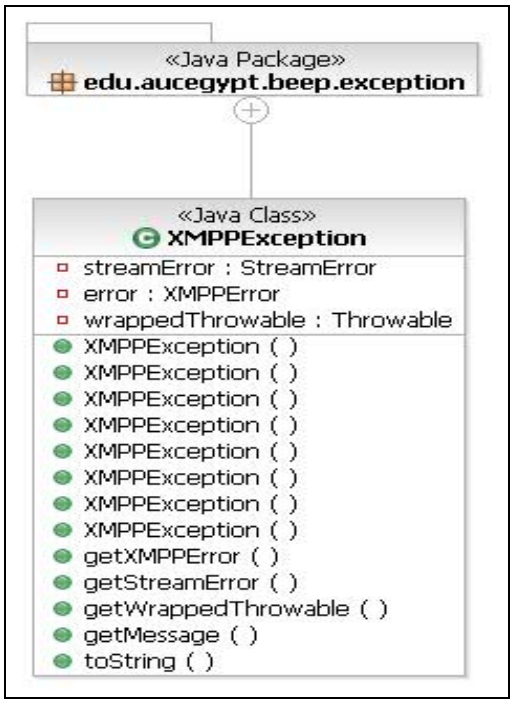

Figure 5 The Exception Package

- **The XMPPException class:** Is a generic exception that is thrown when an error occurs performing an XMPP operation. XMPP servers can respond to error conditions with an error code and textual description of the problem, which are encapsulated in the XMPPError object. When appropriate, an XMPPError instance is attached to instances of this exception. When a stream error occurred, the server will send a stream error to the client before closing the connection. Stream errors are unrecoverable errors. When a stream error is sent to the client an XMPPException will be thrown containing the StreamError sent by the server.

vii. Utility Package

This package contains all the utility classes used in the framework. Currently, it has two classes only which are used for providing utility methods for String objects and encoding/decoding methods for the Base64 encoding as defined by RFC 2045, N. Freed and N. Borenstein. Other classes maybe added in further stages of implementation.

- **The Base64 class:** This class performs the Base64 encoder and decoder. This class provides encoding/decoding methods for the Base64 encoding as defined by RFC 2045, N. Freed and N. Borenstein. RFC 2045: Multipurpose Internet Mail Extensions (MIME).
- **The StringUtils class:** This class contains a collection of utility methods for String objects. Most of the functions in this class are used in parsing the XML stanzas that are exchanged between the client and the server.

E. Beep Modules

i. The Bluetooth Module

- **The Bluetooth Server:** The Bluetooth module is responsible for searching for the nearest Bluetooth server. That server is running a service which is identified in the framework. The Bluetooth server acts as a bridge between the framework and the internet. It takes the requests from the client and passes them to the internet.
- **The Bluetooth Client:** In the framework there is the implementation of JSR 82 (Java APIs for Bluetooth). By overriding its methods, the framework searches for the Bluetooth posts in range. Then it starts to search for posts that have the specified service. The first post found that contains the desired service; it starts to connect to it automatically.

ii. The WiFi Module

The Smart Device Framework (SDF) of OpenNETCF was used to implement the WiFi module for PocketPC devices to provide a set of functionalities that were not possible to develop using JME libraries. SDF is an application framework which enriches and extends the .NET Compact Framework. Currently Beep WiFi module supports:

- Search for WiFi access point within range.
- Connect to any of the available unsecure WiFi access points.
- Connect to any of the available secure WiFi access points if the key is provided.
- Disconnect from the access point that the device is already connected to.
- Roam between available access points to ensure maximum availability of the connection.

The XMPPConnection class in the Java side of the framework is the controller of this module as it is the only class that can connect to this module through socket programming. A designated port is reserved for exchanging the request/response messages exchanged between the XMPPConnection and the Beep WiFi module. The module has two running threads. The first one is responsible for reading any commands that are sent by the XMPPConnection. The other one handles the response write back in order to know the result of the previously sent request.

iii. The Roaming Module

The roaming module is responsible for controlling the wireless media modules. It is responsible for controlling the logic behind the roaming functionalities as follows:

- It connects to the WiFi module using a localhost connection that facilitates the communication between the two applications. For the Bluetooth, it just creates an instance of it due to the fact it is written using the same language.
- It sends specific commands to connect, reconnect, or search for available access points. It receives other commands that indicate connected or access points are available to connect to.

- The sweep for available access points is a command sent to the wireless media modules according to a schedule. Every five seconds a command is sent to trigger a new search to refresh the list of available access points.
- It is the module responsible for taking the decision whether to disconnect and connect to a new access point or not.
- Whenever the wireless media module is ready for switching access points, the roaming module is notified to save the current state of the connection and establish a new connection with the same parameters. For example, if the user is logged-in the roaming module saves the username and the password and provides them in the new instance of the connection.
- Again in the period of switching access points (sometimes it takes time according to the available hardware) the user is blocked from sending and receiving any data.

F. Sample Application

The Beep framework has some excellent features that can be utilized by JME developers. A sample application was developed that utilizes some of the features supported by the framework. The Application starts with creating an XMPP connection and a simple sign in interface where the username, password and resource can be entered by the user.

```
try{
   conn = new XMPPConnection( "talk.google.com", 5223, "gmail.com",
            msg,XMPPConnection, NOWIFI_NOBLUETOOTH);
} catch(Exception ex) {}
```

The credentials of the user are retrieved and sent to the server through the function login provided by the XMPPConnection class. The server verifies the credentials. In case of a successful login, an authentication snippet is sent to the logged user and a presence snippet is sent to the server to notify all online contacts of the logged user with his new status.

```
Authentication authentication = new Authentication();

authentication.setUsername(username);
authentication.setPassword(password);
authentication.setResource(resource);
authentication.setPassword(password);
sendSnippet(authentication);
snippetWriter.sendSnippet(new Presence(PresenceType.AVAILABLE));
```

If the user was not authenticated, a presence snippet is sent to the user with an error specifying that he is not authenticated and the logging process is stopped. Once the

logged user receives the authentication snippet, the user creates a roster snippet requesting the retrieval of all his contacts and the status of each.

```
Roster roster = new Roster();

try {
      conn.sendSnippet(roster);
  }
  catch (XMPPException ex) {}
```

The server fetches the contacts of the user and replies back. Trivial logic approaches such as sorting the contacts with respect to status, is done at the client side. The second part of the protocol, is handling the messaging after the presence part was covered.
The Framework provides an easy method for developers to handle messaging, similar to handling presence. The developer will override the other method in the SnippetProccessor called processMessage. This function takes a message object as a parameter. The developer should provide the proper implementation for this method depending on his needs. For instance, when having multiple forms of chat, the function will forward the proper message to the intended chat form. However the same techniques can be utilized not only in text chatting. For example, the messages could be used to transfer a string that corresponds to an action in a game; thus the developer provides implementation to the function to take appropriate actions depending on each message. Furthermore, the message could be utilized to send voice messages, where the developer implements a function that will receive the bytes containing the voice message, decode it, and then play it as a voice message. Apparently the Framework will do all the work for the developer, who will only provide an implementation to a function to utilize the message received from the framework.

IV. Conclusion

In this article, we presented a framework that facilitates the process of building communication capable mobile applications. Mobile devices can transparently communicate with one another using selected communication technologies such as Bluetooth and WiFi, and any other available technology that can support connection to the Internet such as WiMAX, 3G, and GPRS. Furthermore, the framework supports the roaming of mobile phones between various physical locations without the need for applications to keep track of the actual locations of users. The detailed architecture and design of the framework is presented in this article, along with sample applications that demonstrate the feasibility of the framework.

Acknowledgement

A special acknowledgement is made to the following members of the American University in Cairo for their contributions in producing this effort: Karim Hamdan, Amr Gouda, Ahmad AbouShady, and Ahmad Afifi.

References

[1] "Extensible Messaging and Presence Protocol (XMPP): Core". The Internet Engineering Task Force. October, 2004. [online]. Available: http://www.ietf.org/rfc/rfc3920.txt

[2] "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence". The Internet Engineering Task Force. October, 2004. [online]. Available: http://www.ietf.org/rfc/rfc3921.txt

[3] M. Day, S. Aggarwal, G. Mohr, J. Vincent. "Instant Messaging / Presence Protocol Requirements". The Internet Society. Feb, 2000.

[4] Hiroaki Kikuchi , Minako Tada , Shohachiro Nakanishi. "Secure Instant Messaging Protocol Preserving Confidentiality against Administrator". 18th International Conference on Advanced Information Networking and Applications (AINA'04). Volume 2. March 2004, pp.27.

[5] Jerry Gao , Mansi Modak , Satyavathi Dornadula , Simon Shim. "Mobile Jabber IM: A Wireless-Based Text Chatting System". 2004 IEEE International Conference on E-Commerce Technology (CEC'04). July 2004, pp.337-341.

[6] Julie Rennecker, Alan R. Dennis, Sean Hansen. "Reconstructing the Stage: The Use of Instant Messaging to Restructure Meeting Boundaries". Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06) Track 1. January 2006, pp. 27a

[7] Lee-Sub Lee , KyungSun Choi , Dongwon Jeong , Soo-Hyun Park , JuHum Kwon." An Inter-Domain Authentication Mechanism for XMPP/Jabber". Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06). IEEE. Aug 06, pp. 268-275.

[8] Peter Saint-Andre, "Jingle: Jabber Does Multimedia". IEEE Multimedia. Vol 14, No. 1. January – March 2007 pp. 90-94.

[9] Peter Saint-Andre. "Streaming XML with Jabber/XMPP". IEEE Internet Computing. September 2005, pp. 82-89.

[10] Rick Hayes-Roth. "Puppetry vs. Creationism: Why AI Must Cross the Chasm". IEEE Intelligent Systems. September 2006, pp. 7-9.

[11] Sixto Ortiz. "Instant Messaging: No Longer Just Chat". IEEE Computer. March 2001, pp. 11-13.

[12] Stephanie L. Woerner , JoAnne Yates , Wanda J. Orlikowski.” Conversational Coherence in Instant Messaging and Getting Work Done”. 40th Annual Hawaii International Conference on System Sciences (HICSS'07). January 2007, pp. 77c

[13] Steven J. Vaughan-Nichols. “Presence Technology: More than Just Instant Messaging”. IEEE Computer. October 2003, pp. 11-13.

[14] Wei Li , Fredrik Kilander , Carl Gustaf Jansson. “Extending Instant Messaging to Provide Pervasive Personal Communication”. International Workshop on System Support for Future Mobile Computing Applications (FUMCA'06). September 2006, pp. 43-50.

[15] Zhen Xiao , Lei Guo , John Tracey. “Understanding Instant Messaging Traffic Characteristics”. 27th International Conference on Distributed Computing Systems (ICDCS '07). June 2007, pp.51.

[16] Scott Ellison, William Stofega, Elisabeth Rainge. Neustar. “Mobile Instant Messaging: The Next Global Messaging Opportunity”. Nov, 2006.

[17] “Beyond Instant Messaging, Jabber XCP: Presence and Routing Engine for Real-Time Internet”. Jabber,inc.[Online]: http://www.jabber.com/media/Jabber_Inc_Beyond_IM_White_Paper.pdf Jabber, inc. 2006.

[18] “The Power of Presence”. Jabber, inc. [Online:http://www.jabber.com/media/JabberIncPresence. White_Paper.pdf 2006.

[19] “iMobile EE: an enterprise mobile service platform”. Wireless Networks. Volume 9, Issue 4. July 2003, pp.283-297.

[20] Saleh Almanei. “Secure Instant Messaging: The Jabber Protocol” Oregon State University. June, 2003.[Online].Available: http://islab.oregonstate.edu/papers/03Almanei.pdf

[21] J. Rosenberg, H.Schulzrinne, G. Camarillo, A. Johnston. “SIP: Session Initiation Protocol”. The Internet Engineering Task Force. June 2002. [Online]. Available: http://tools.ietf.org/html/rfc3261

[22] A.J.H. Peddemors, M.M. Lankhorst, J. de Heer. “Combining presence, location and instant messaging in a context-aware mobile application framework” Telematica Instituut. March 20, 2002. [Online].: https://doc.telin.nl/dsweb/Get/Document-21982/PLIM_d28.pdf

[23] A. Abou Shady, A. Afifi, A. Sabry, A. Gouda, K. Hamdan, M. Hamza, “Beep Software Design Description Document”, Report Submitted to The Department of Computer Science and Engineering, The American University in Cairo, 2007.

Mohy Mahmoud received his B.S. and M.Sc. degrees in Electrical Engineering from Cairo University in 1975 and 1981 respectively. He then received his Ph.D. from Bath University in 1984. He is currently an Associate Professor at the American University in Cairo and has more than 25 years in teaching and research in computer hardware and software systems courses as well as advanced formal training and practical experience in Computer Simulation. His accumulated years experience included computer design, programming, testing and computer modeling His work experience included the primary responsibility for the selection of hardware, design and programming of bilingual software for computerized information system for 18 rural Egyptian governorates participating in the Basic Village Services Project, and ongoing Arabization of generic software to suit a wide range of local development policy-making and planning needs.His research interests include multi-lingual interfaces, ad-hoc networking, image processing, and data encryption. Dr. Mahmoud is a member of both IEEE and ACM.

Sherif G. Aly received his B.S. degree in Computer Science from the American University in Cairo, Egypt, in 1996. He then received his M.S. and Doctor of Science degrees in Computer Science from the George Washington University in 1998 and 2000 respectively. He worked for IBM during 1996, and later taught at the George Washington University from 1997 to 2000 where he was nominated for the Trachtenberg prize-teaching award for his current scholarship and scholarly debate. He spent two years as a guest researcher for the National Institute of Standards and Technology at Gaithersburg, Maryland from 1998 to 2000. Dr. Aly also worked as a research scientist at Telcordia Technologies in Morristown, New Jersey, in the field of Internet Service Management Research, and as a Senior Member of Technical Staff at General Dynamics Network Systems. He also consulted for Mentor Graphics and taught at the German University in Cairo. He is currently a faculty member at the Department of Computer Science and Engineering at the American University in Cairo, and recipient of the Egypt National State Prize for research. Dr. Aly published numerous papers in the area of distributed systems, multimedia, digital design, and programming languages. His current research interests include pervasive systems, programming languages, multimedia, directory enabled networks, and image processing. Dr. Aly is a member of IEEE.

Ahmed Sabry received his B.S. degree in Computer Science from the American University in Cairo, Egypt, in 2007. He is a graduate student in the M.S program at the American University in Cairo. He worked as a graduate assistant at the American University in Cairo. He is currently working for IBM. His research interests are social networks, grid computing, pervasive systems and mobile computing.

Mostafa A. Hamza received his B.S. degree in Computer Science from the American University in Cairo, Egypt, in 2007. He is a graduate student in the M.S. program at the same university. He worked for LinkDev during 2007-2008 as a solution developer working on online business applications. His research interests include software product line architectures, pervasive systems and mobile computing.