ISSN:0975-9646

MANCAF: A Framework for Building Collaborative Applications in Mobile Ad Hoc Networks

Mr. Hardik S. Mehta#[1], Dr. Dhaval R. Kathiriya*[2]

#*Research Scholar, School of Computer Science, RK University, Rajkot*
**Director IT, Anand Agricultural University, Anand*

Abstract-- **This paper presents the MANCAF framework that enables programmers to develop mobile collaborative applications. The framework provides an API that is easy to adopt and capable of creating advanced collaborative applications. The framework was built to provide applications providing pure peer-to-peer network where all nodes have the same responsibility and services. Further, the MANCAF framework provides services to transparently manage the finding of new and lost peers. The message component of the framework makes it possible to exchange any kind of data between clients. The MANCAF has been implemented in Java 2 Micro Edition (J2ME) and runs on standard mobile phones. The framework supports management and communication of mobile ad hoc networks (MANETs) like Bluetooth. The paper describes the architecture, the API and the application developed using the MANCAF framework.**

***Keywords*— Collaborative Application, Framework Designing, Mobile Ad-hoc Network.**

I. INTRODUCTION

Collaborative applications are not an exception when we are developing an application which satisfies the user need through application functionality. We can easily develop an application by determining user's needs in advance and then developing a correspondent application but the experience proves that it is not the case always. Users will refuse to use such application that does not support their needs.

Most wireless devices support some kind of personal area network (PAN) technologies like IrDA and/or Bluetooth [1]. PANs are commonly used for transferring data between two mobile devices. A PAN can be seen as a digital sphere around the mobile device enabling a collaborative network for users within range. The digital sphere opens for mobile computer supported cooperative work (mobile CSCW) [2] [3]. In such environments, the support for mobile peer-to-peer is essential, and the support and establishment of mobile ad hoc networks (MANETs) are necessary. A MANET is a self-configuring network where devices can join and leave the network dynamically making the wireless network topology unstable and unpredictable. MANETs can be utilized in situations where persons with mobile devices meet and there is a need for exchange of data.

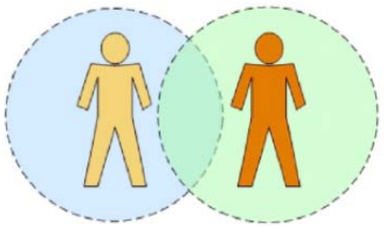
Figure 1: Collaboration in PAN Technologies

MANETs enable mobile users to interact in new ways. The interaction between users can either be explicitly initiated by the users, it can be automatically initiated by the mobile devices, or a hybrid of the two [4]. Such applications can be used for initiating collaboration between users of same interests, e.g., an application for finding people with same interest in a group meeting [5]. Further, MANETs can be used to create application for proximity chats and file exchanges, or simply for free time like games.

The MANCAF framework is introduced to enable rapid development of collaborative applications for mobile devices on the Java 2 Micro Edition (J2ME) platform. Main goal is to develop a high-level programming framework that made it possible for developers to only use simple primitives and methods to manage the complexity of MANETs. MANCAF framework is transparent and hides the network technology used for communication. The framework enables researchers and developers to explore utilization of MANETs and how MANET applications can provide collaborative support for mobile users. The main contribution of this paper is to describe the MANCAF framework and experiences us from building and using the framework.

The rest of the paper is organized as follows. Section 2 describes the architecture and the design of the MANCAF framework. Section 3 describes how the framework can be used in applications development. Section 4 relates MANCAF framework to similar approaches. Finally, Section 5 concludes the paper and defines future work.

II. MANCAF FRAMEWORK

This section describes the architecture, design and implementation issues related to the MANCAF framework.

A. MANCAF and J2ME

Sun Microsystems developed Java 2 Micro Edition (J2ME) [6] to provide a general execution platform for resource constrained devices. J2ME consists of various configurations, profiles and optional packages to support various kinds of equipment. For mobile devices like mobile phones and PDAs, the Connected Limited Device Configuration (CLDC) is the most common configuration that is tailored for devices with wireless network capacities. In the same way the Mobile Information Device Profile (MIDP) is the most used profile for such devices. The MIDP provides an environment for developing and managing applications, called MIDlets, including GUI libraries [6]. Most mobile phones sold today supports J2ME and MIDP 2.0. In addition, some mobile phone models support a variety of optional packages that that provides API for various purposes like location, 3D graphics, multimedia support, security, speech etc. Figure 2 shows how our MANCAF framework fits into the J2ME environment.

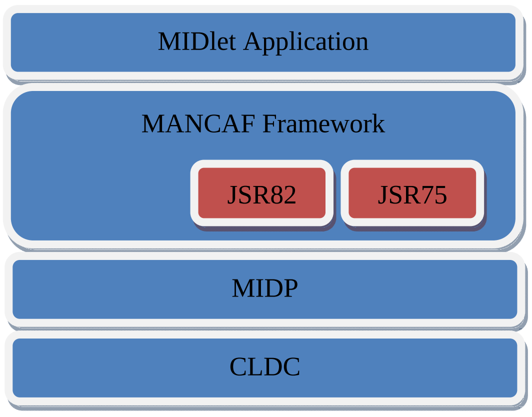

Figure 2: MANCAF and J2ME

Our framework is built upon MIDP 2.0. In addition, the MANCAF framework uses two optional packages:

• **JSR82**: J2ME API to access and manage Bluetooth networks.

• **JSR75:** J2ME API to access Personal Information Management (PIM). The JSR75 makes it possible to read and write to the file system as well as access data (both read and write) in the built-in address book and calendar of the mobile device.

The current MANCAF implementation only supports Bluetooth networks, but the architecture is made modular to also support other types of networks when they become supported in the J2ME environment.

B. MANCAF Framework Design:

The MANCAF design is based on a layered architectural pattern where each layer is assigned with its own responsibility, and one layer is based on the layer below. By using the layered approach, the architecture would gain positive characteristics like modularity and transparency.

The negative effect by using this approach could be slower execution if the applications often have to go up and down several layers to carry out the operations. As the MANCAF framework should be used on resource constrained execution platform, we decided to use few layers in the architecture. Figure 3 shows the compact design and Figure 4 shows the high-level architecture of MANCAF.

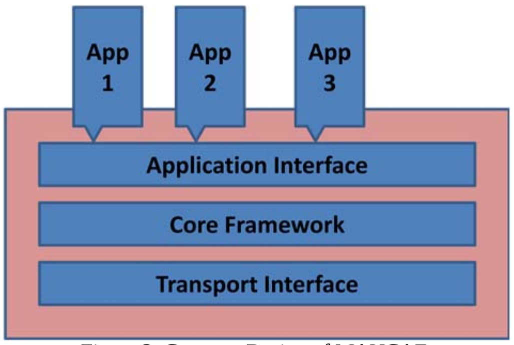

Figure 3: Compact Design of MANCAF

Node: A node is a logical representation of a peer, i.e., a mobile phone running the framework. Two or more nodes can form a mobile ad hoc network.

Group: A group is a collection of nodes that know of each other's existence. All the nodes in a group can communicate with each other.

Network: A network is an abstraction of the network layer representing the communication medium accessed by the framework instance. The network layer can consist of several network implementations that also can be run simultaneously.

Concurrency: This component is responsible for managing the conflict during the collaboration.

Session: A session represents the lifetime of all the communication between the nodes in a group. A session keeps track of known nodes, groups and available network mediums.

Framework: This component is responsible for whole collaboration. All communication between the user interface and the functionality of framework done through this. It receives message from connection manager and it forwards each message to appropriate component.

Document: This component is responsible for managing the document.

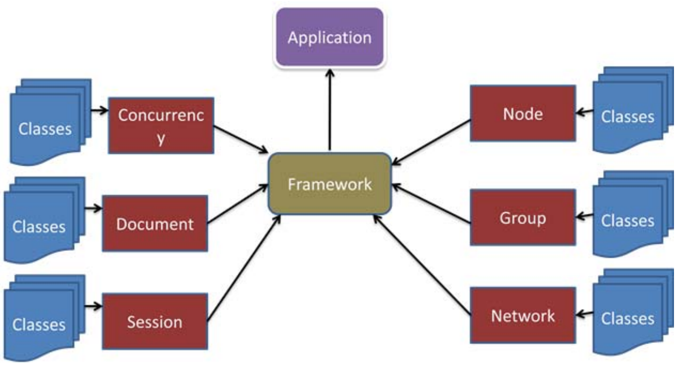

Figure 4: High-Level Design of MANCAF

Application: A collaborative application will be implemented as a MIDlet running on top of the MANCAF framework.

C. *Design and Implementation:*

This section describes design and implementation decisions made in MANCAF to provide a transparent high-level API.

Pure versus hybrid peer-to-peer model: Pure peer-to-peer approach where all the peers have the same responsibility for managing resources and communication is not supported directly in Bluetooth. The Bluetooth technology is based on a master-slave communication protocol, where the Bluetooth master will search for nearby Bluetooth devices that will become slaves when communicating. The master-slave configuration is essential the same as client server and suffers from the same problems of single point for failure. A possible solution to avoid this problem could be to make the master node delegate the server responsibility to a new node in case of a failure, but this would require a very complex protocol without getting a high availability of the system. In MANCAF we implemented a pure peer-to-peer protocol avoiding the difficulties of the master-slave paradigm. The master-slave connections are established dynamically when there is a need for one node to send a message to another node or nodes. This means that in MANCAF all the nodes know all other nodes, and every node have the same responsibility.

Communication protocol: Bluetooth provides two different protocols for implementing communication between peers: RFCOMM and OBEX. RFCOMM emulates a RS-232 serial connection, which provides a stream-based interface. The advantage using RFCOMM is that it is very straightforward to implement in J2ME, but it has some major disadvantages. RFCOMM only supports one session at a time between two devices, and the maximum amount of active RFCOMM services a Bluetooth device can have is 30. When RFCOMM is used, the receiving Bluetooth device must read the stream and later parse the stream. OBEX is much more versatile than RFCOMM and provides support for setting up a session of two communicating devices. The information is sent between the devices in the form of a put or a request pattern. In contrast to RFCOMM, OBEX also fully supports headers (also user defined) to describe the context of the message. MANCAF uses the OBEX protocol that enables the support for a variety of types of messages including serialsable objects. It also makes it possible to only send the headers, making it possible for the client to decide to receive the data or not.

Detecting new nodes: MANETs are characterized by a very dynamic network topology where nodes dynamically added and removed from the network. The discovery of new nodes in MANCAF implemented using the Bluetooth discovery protocol provided in J2ME that searches for all nearby Bluetooth devices. It then filters and performs a service search for all mobile phones detecting all devices running a specific MANCAF service. If the discovery detects any new nodes, references to the new nodes are created so that a connection can be established later on. After a completed search, the node shares the result with all the notes it has found. This process is illustrated in Figure 5. In 5A, node A searchers and discovers B and C. In 5B, node A sends messages to B and C, which contain information about all the nodes in the vicinity. A search for new nodes is initiated when a MANCAF application is run. How often the search for new nodes is run during the execution of an application is up to the application developer to decide.

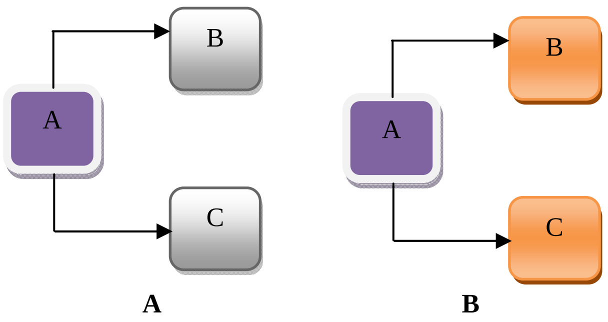

Figure 5: Detecting a new node

Initialization of the framework: The first time a MANCAF application is run, the framework has to be initiated from the MIDlet (the application) by calling the method initialize(). The initialization will be performed through the layers of the framework by initializing the session layer, the network layer and the Bluetooth implementation. The last step of the initialization process is to create a network listener (in current version a Bluetooth listener). The network listener makes it now possible for other nodes running the application to perform a service discovery and find this node. After the framework has been initiated, the MIDlet can start a search for other nodes as shown in Figure 6. The search is performed through the layers of the framework and the MIDlet is alerted when a node is found.

Other design considerations: When testing J2ME applications in the actual execution environment, there was no support for a console for test output on the mobile devices. Even though this is available on simulators running on PCs, it is essential to be able to track errors on the mobile devices. The applications running fine on simulators do not necessarily behave the same way on the physical mobile device. To catch and manage runtime errors, the framework provides a strong handling of exceptions that catches typical common errors like lack of initialization of framework, file, group and node not found etc. In addition to the exception handling, we also provide a log package that can be used by all classes in the framework that stores and manages runtime-messages. The log is useful for debugging as well as getting real-time information of the framework's progress and well-being.

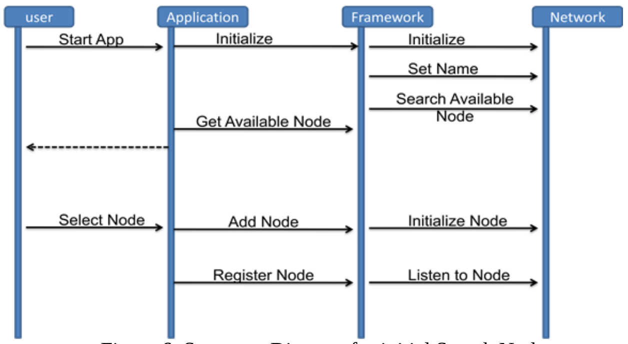

Figure 6: Sequence Diagram for initial Search Node

III. MANCAF APPLICATION DEVELOPMENT

As mentioned in the beginning of this paper, a main goal of the MANCAF framework is to create a framework for collaborative application development of mobile peer-to-peer applications. In this section, we will describe through code examples the main parts of the MANCAF that an application developer has to use.

A. Initialization of Application

Before you start to create any application, you need to import all the necessary parts of the framework

```
public class MANCAFChat extends MIDlet implements
FrameworkSubscriber, Commandlistener {
private Framework framework ;
framework = Framework . getInstance ( ”MyGroup” , ”
Chat2Me ” , new Bluetooth ( ) , this ) ;
framework.initialize( ) ;
framework.search( ) ;
```

The MIDlet created must implement the FrameworkSubscriber interface from the MANCAF framework. The CommandListener is a J2ME interface to catch events from the user interface. Then a framework variable must be created. Then initiate the framework with the parameters for name of group, name of service, the network used and a reference to the MIDlet itself. The device running the application is now available for service discovery from other devices running MANCAF. Next task is to search for nearby devices running the same MANCAF service.

B. Event-handling in MANCAF

After the framework has been initiated, the application need to implement the four methods, nodeFound, nodeGone, FinishSearch and messagePart provided by the FrameworkSubscriber interface. The nodeFound is invoked when a new node is found in the network. Note that there is no automatic search for new nodes in MANCAF after the initial search has been completed. The application programmer must include a search for new nodes in the application herself. The methods nodeGone is invoked when the background ping/echo mechanism has detected that one of the nodes in the group is not reachable anymore. Detection for lost nodes is run in background by the framework in a separate thread. The FinishSearch method is called when the initial search has completed. Following Listing show an example of how nodeFound, nodeGone and FinishSearch can be implemented. In the implementation of the first two methods, the node.getNodename() method is used to notify the user through the GUI that the node has joined or left. For the latter method, the application sets the GUI in focus and refreshes the GUI.

```
public void nodeFound (Node node )
{
dialog.append ( node.getNodename ( ) +” has joined ” , null
) ;
}

public void nodeGone (Node node ) {
dialog.append ( node.getNodename ( ) +” has   left ” , null )
;
}

public void FinishSearch ( ) {
display.setCurrent ( dialog ) ;
}
```

C. Manage Messages

The management of messages in MANCAF is flexible in terms of what data can be sent between peers. In addition, the management of message interface is easy to use and understand for the application programmer. Following Listing shows an example of how two different types of messages can be created and sent to another node.

```
Message message = new Message ( ) ;
message.addElement( 1 , ” type ” ) ;
message.addElement( ” Hello world ! ” , ” info” ) ;
message.addReceipt(node);
framework.sendMessage(message) ;

Message message = new Message ( ) ;
message.addElement( 2 , ” type” ) ;
message.addElement( ” Hello world ! ” , ” info ” ) ;
message.addReceipt(node);
framework.sendMessage(message) ;
```

Listing 4 shows how a message can be received. In this example, the message element is used to identify what type of data is sent in the message element called ”info”.

```
public void messageReceived (Message message )
{
int type = message . getInt ( ” type” ) ;
switch ( type )
 {
case 1 :
String info = message.getString ( ” info” ) ;
break ;
case 2 :
boolean info = message.getBoolean ( ” info ” ) ;
break ;
default :
break;
}
}
```

IV. RELATED WORK

There are several projects that have developed frameworks for developing collaborative application in MANETs. In this section we will present the most prominent projects.

JXTA [7, 8] is an open-source framework for developing P2P applications. JXTA provides a set of protocols and APIs for general-purpose, computer-to-computer communication and is platform and network independent. JXME is JXTA for Java 2 Micro Edition (J2ME) and is a lightweight implementation of JXTA for mobile devices. It is specifically aimed at devices without sufficient computation and/or communication resources to participate in the network on their own. The JXME implementation provides full JXTA functionality through the use of a relay host. There is also a JXME proxiless initiative, but there is currently no stable implementation. As JXTA does not have a pure peer-to-peer version working for J2ME, it cannot be compared to MANCAF.

Mobile Chedar [9] is a middleware being an extension to the Chedar peer-to-peer network allowing mobile devices to access the Chedar network and communicate to other Mobile Chedars. The goal of the Chedar software is similar to Peer2Me: To provide a convenient API for peer-to-peer application developers. The Mobile Chedar is implemented in J2ME and Bluetooth is used for communication. In contrast to MANCAF, the Mobile Chedar is based on a hybrid peer-to-peer model that uses a Mobile Chedar gateway node as the master in the network. The Mobile Chedar gateway node is run on a PC that also provides an Internet gateway for the mobile nodes. However, this approach suffers from having a single point of failure like client-server solutions.

Proem [10] is a framework for developing and deploying P2P collaborative applications in a mobile ad-hoc networking environment. The main objective of Proem is to provide a common framework for rapid development of applications for ad-hoc network environments. The framework is implemented in Java, and can be run on various wireless mobile devices. Proem is designed to be independent of underlying network transport protocols, and can be implemented on top of TCP/IP, HTTP, Bluetooth and others. The original Proem was based on a Java Standard Edition, limiting the devices to run Proem to powerful PDAs. There have been attempts to create a J2ME version of Proem that have not succeeded.

MOBY [11] provides a network for mobile peer-to-peer exchange of services and data. MOBY offer a dynamic service location and client mapping to achieve an adaptive network optimizing performance and reliability. MOBY uses heavily JINI functionality and can there for not be run in a J2ME environment.

V. CONCLUSION & FUTURE WORK

In this paper we have presented the MANCAF framework for rapid development of mobile collaborative applications. Despite the limitations in J2ME with a stripped down Java class-library and the limited resources on mobile phones in terms of CPU and memory we have managed to develop a full-fledged framework that transparently manages a MANET. The MANCAF API is very simple to learn and consists only of a few lines for framework initialization and implementation of four methods that will be invoked for events.

The Future work is to develop as many as collaborative application using MANCAF framework. The framework is then tested to determine with checking that specified technical and not technical.

REFERENCES

[1] Miller , B. A. and Bisdikian, C. (2004), *Bluetooth Revealed*, Addison-Wesley, 2 edition

[2] Wiberg, M. and Grönlund, Ä. (2000), Exploring Mobile CSCW: Five areas of questions for further research, In Proceedings of IRIS23 (Information Research in Scandinavia), Trollhättan, Sweden.

[3] Papadopoulos, C (2006), Improving Awareness in Mobile CSCW, IEEE Transactions on Mobile Computing, vol. 5, no. 10, pp. 1331-1346, Oct.

[4] A. I. Wang, M. S. Norum, and C.-H. W. Lund. Issues related to Development of Wireless Peer-to-Peer Games in J2ME. In First Conference on Entertainment Systems (ENSYS 2006), page 6, Guadeloupe, French Caribbean, February 23-25 2006.

[5] A. I. Wang, C.-F. Sørensen, and T. Fossum. Mobile Peerto-Peer Technology used to Promote Spontaneous Collaboration. In The 2005 International Symposium on Collaborative Technologies and Systems (CTS 2005), page 8, Saint Louis, Missouri, USA, May 15-19 2005.

[6] S. Microsystems. Java 2 Platform, Micro Edition (J2ME). Web: http://java.sun.com/j2me/, 2005

[7] J. Brendon and J. Wilson. JXTA. New Riders Publishing,2002

[8] N. Maibaum and T. Mundt. JXTA: A Technology Facilitating Mobile Peer-To-Peer Networks. In International Mobility and Wireless Access Workshop (MobiWac'02), pages 7–13, Fort Worth, Texas, USA, 12 October 2002.

[9] N. Kotilainen, M. Weber, M. Vapa, and J. Vuori. Mobile Chedar A Peer-to-Peer Middleware for Mobile Devices. In Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'05), pages 86–90, 2005.

[10] G. Kortuem. A methodology and software platform for building wearable communities. PhD thesis, University of Oregon, December 2002

[11] T. Horozov, A. Grama, V. Vasudevan, and S. Landis. MOBY - A Mobile Peer-to-Peer Service and Data Network. In 2002 International Conference on Parallel Processing (ICPP'02),pages 437–444, 2002