2009 International Symposium on Computing, Communication, and Control (ISCCC 2009)
Proc .of CSIT vol.1 (2011) © (2011) IACSIT Press, Singapore

A New Approach for Implementing RMI in Mobile Phone Platforms

Mohammadreza Razazi and Meisam Hejazinia +

Amirkabir University of Technology, Dept. of Computer and IT Engineering, Tehran, Iran

Abstract. This paper discusses challenges of implementing RMI over mobile phone platforms. Improving Distributed Computing capabilities in mobile phones has three emerging challenges: first, mobile devices have a limited processing power and storage, second, mobile devices have a limited electrical power, and third, there is a limited networking capability. Java RMI supports wireless environments, such as Bluetooth, General Packet Radio Service (GPRS), and Wirelesses LAN (WLAN). It provides a software infrastructure for Desktop and Server environments, but unfortunately there is a lack of support over limited devices such as mobile phones. Today, mobile phones have the same capability of Desktop computers in 90s. At that time, computers were used in distributed systems and the main challenge was optimization. But when Desktop's processing power and memory increased, the focus changed. Today, mobile phone platforms provide an opportunity for improving distributed processing, but due to their constraints, most of the previous solutions should change according to the general guidelines for optimization. This paper describes different network infrastructures for implementing RMI, and provides a guideline for suitability of the network medium, the underlying software layer of RMI. Moreover, the concept of Roaming layer is discussed and finally, an architecture style is proposed to facilitate Distributed Processing over mobile phone platforms.

Keywords: RMI, Distributed Computing, Mobile phone, Bluetooth, J2ME, Mobile Peer-to-Peer Applications, Constrained devices

1. Introduction

Mobile phones have recently become more common in human's life. Although there were many opportunities to use them as platforms to facilitate Distributed Computing, there is still a big hole in software infrastructure to improve distributed computing over this platform. Improving Distributed Computing capability in mobile phones has three emerging challenges: first, mobile devices have a limited processing power and storage, second, mobile devices have a limited electrical power, and third, there is a limited networking capability. Java RMI supports this type of computing in wireless environments, such as Bluetooth, General Packet Radio Service (GPRS), and wirelesses LAN (WLAN). It provides a software infrastructure for Desktop and Server environments, but unfortunately a lack of support over limited devices such as mobile phones exists [4].

There are many useful scenarios showing the use of RMI in mobile phones. Suppose John has an application on his mobile phone recommending him to choose the portfolio for the stock market. His mobile phone has some limitations in the dimension of the processing power. Fortunately his wife, Alice, is near him, and she has another application which needs only some inputs related to the market situations to estimate the price of each stock. In addition, they have two sons called Jack and Joe, who have this application installed on their mobile phones. If John's application communicates with three other devices, via RMI, he could use the processing and electrical power of their mobile phones, and as a result, he could easily decide to change his stock portfolio and call the stock broker to change it accordingly. The point is

+ Corresponding author. Tel.: +989376364895

E-mail address: hejazinia@aut.ac.ir

that John was in a car, with his family on a journey and it was hard and a little bit impossible for him to use his laptop.

Over the past years, there were some frameworks developed for Computer Supported Collaborative Work (CSCW)[8]. The focus of these frameworks was on writing a software layer to help developers to overcome complexities of network application writing. This is the same purpose of RMI, which is providing transparency by hiding the complexities of the underlying infrastructure. When we use RMI, we do not need to know anything about socket programming, device discovery and properties, and it is similar to invoke a method of an object on a local device. What CSCW frameworks try to provide, is to hide network programming complexities by decoupling the network layer from the application, and providing a high level interface.

The paper is organized as follows: Section 2, describes different network infrastructures, for implementing RMI over mobile phones; it also compares different options, and provides a guideline for the improvement path. Section 3, describes different options and challenges related to software infrastructure development, and provides a comparison between different options. Section 4, provides unique challenges exists on ME platforms, Section 5 provides a proposed architecture for the platform, and Section 6, concludes the paper.

2. Network Infrastructures

There are three options available for providing RMI, over mobile phones: Bluetooth, WLAN, and GPRS. GPRS is based on GSM. In the middle of these two technologies is the improved version of GSM, which cellular operators enhance their systems, to improve to GPRS. We included both GSM and the improved version of it in the third option.

The current solution for RMI, on constrained devices, as we told is based on the servlet, so it is over http [9]. Although it is claimed that this solution is wise, for it decouples RMI from the underlying infrastructure, it has a huge shortcoming. On Desktop platforms, TCP/IP layers are developed over the Bluetooth layer, but there is no such implementation on constrained devices. The big problem is that it has a huge overhead, which makes it unsuitable for this platform. Having a low processing power and memory with a critical power consumption, makes constrained devices different from other platforms. Thus, in this platform, optimization is critical and performance is an important non-functional requirement, so providing RMI on a lower layer is a good performance decision, which has not been concerned yet.

On the other hand, we have another critical factor, which is the wireless network. We don't have any wired networks on mobile phones, and the wireless network consumes much of the electrical power of the device. Although there were some optimizations in the stream to be sent over the wireless network in constrained devices, such as the compression of streams to be sent, it is not suitable. The better decision is to optimize the current solution, and just walk a bit backward. Instead of using the good principals of software engineering, which is decoupling the layers, we try to combine them, and also try to implement RMI over the lower layers than TCP/IP. Another solution is to optimize the messages that are sent periodically, like pings and try to combine them in order to make RMI more efficient. The previous proposed solution to improve RMI over wireless networks, used proxy to facilitate compatibility with RMI specification, but while there is no implementation on constrained devices, this solution is not needed. Data compression especially on serialized data can improve performance on such devices. Moreover, to optimize the protocol a new mechanism for Registry lookup should be developed so the previous Registry mechanism is not suitable, especially for MANET. The other optimization to consider is the Distributed Garbage Collection, which should be provided, and has a huge overhead. Though this part could be omitted, for it is less meaningful in wireless environments. The link is a subject to sudden disconnections and can be handled at transport or similar layers.

As we can see there are many opportunities to provide RMI on a new platform, especially from the network point of view. Another challenge here is to choose the most suitable network infrastructure to provide RMI over that. Concerning these three options to compare, some criteria are needed to affect our

decision, including Bandwidth, Power, Range, Cost and Frequency. Table 1 provides information related to these criteria [1]:

To make a decision over these options, there is another criterion, which is the coverage of these networks over mobile phones. Most mobile phones equipped with Bluetooth, have access to GPRS, but a few phones such as E61, E60, E70 of Nokia, and P990i, W960i Sony Ericson are equipped with WLAN.

TABLE 1 NETWORK MEDIUM COMPARISONS FOR IMPLEMENTING RMI OVER MOBILE PHONE PLATFORMS

	Bandwidth	Power	Range	Cost	Frequency
Bluetooth	1 Mbps	1-10 mw	10-100m	None	2.4GHz
WLAN	11 Mbps	50-70 mw	100-200m	Low	2.4GHz
GPRS	115-117kbps	200-800mw	1KM	High	900/1800 MHz

To conclude the most suitable platform, Bluetooth shows up, for it is free, but has some shortcomings upon its limited range, after that we have WLAN which is also free, and suitable, and the third one is GPRS, due to having a high cost of transfer; it makes it difficult for a normal application to communicate. In section 5, we deeply go through this issue and according to the suitability, without interruption of application working on this layer we propose a software layer that supports, roaming between the networks.

3. Frameworks for the Underlying Network Layer

In this section, we are going to analyze different options for the underlying network layer. The good point of these frameworks is that, they decouple the network layer from the application, and provide a qualified and usable framework. Additionally, these platforms could be extended, to provide access to other network mediums. Also for roaming they could be a very suitable platform to enhance RMI over heterogeneous wireless networks on RMI. As we previously mentioned the main point is, to go further and create Grid Computing with the required services on mobile phones.

The options for this software layer are, JXTA, JXME, JXBT, PROEM, OBEX, SyncML, BlueCove, BlueFramework, Jadbas-CLDC, Ergon-J2me, and Peer2me. We are going to provide some information about each of them and then we will compare these options, to find the most suitable one for developing RMI over mobile phones.

JXTA is short for Juxtapose that means side by side. The JXTA project started by some researchers at Sun Microsystems. Its goal is to explore a vision of distributed network computing using peer-to-peer topology, and to develop basic building blocks and services that would enable innovative application for peer groups. It is an open source project under Apache Software License and has three objectives: Interpretability, Platform independence, and Ubiquity. The protocol is specified as a set of XML messages. This means heterogeneous devices with completely different software stacks can interoperate with the JXTA protocols. The Peers in a JXTA based network can be advertised and discover other resources. The problem is Java binding of JXTA, JXTA Java SE, requires Java 2 Standard Edition (J2SE) to run. This makes it unsuited for most mobile devices on the market; also there is no support for Bluetooth as a network medium which also makes it unsuited for mobile devices since Bluetooth is the most common mean of communication on such devices [8].

To overcome these limitations a side project called JXTA-J2ME (JXME) is started. The purpose of JXME is to provide JXTA compatible functionalities on devices using the Connected Limited Device Configuration (CLDC) and the Mobile Information Device Profile 2.0 (MIDP), typically a mobile phone or a PDA. JXME was first designed as a proxy based peer-to-peer solution, relying on a central device acting as a proxy between the peers. This prevents “real” peer-to-peer operated, ad hoc networks. In the newest version, however, this proxy is removed. The main disadvantage of both solutions is the lack of Bluetooth Support [3].

PROEM is a mobile middleware for ad-hoc networks based on WLAN within J2ME. Mobile Chedar is similar to PROEM, but it provides support for peers with fixed P2P network connections. It uses Bluetooth as the underlying communication channel for supporting Chedar P2P protocol [7].

Peer2me framework enables developers to create mobile peer-to-peer applications. It is the result of many projects in Norway University; more than 6 master’s theses have been defined over it. The last version, which is optimized, is the most suitable one for mobile phone platforms. The framework provides an API that is easy to adapt and capable of creating advanced peer-to-peer applications. The framework was built to

provide applications providing the pure peer-to-peer network, where all nodes have the same responsibilities and services. Furthermore, the framework makes it possible to exchange any kinds of data between peers including Java objects. The Peer2Me has been implemented in J2ME and runs on standard mobile phones. The framework supports the management and the communication of mobiles and ad hoc networks (MANETs) like Bluetooth. The positive point is that it is an open source framework [2].

To choose between the options, we need to have some criteria, such as: being open source, suitability for mobile phones, supported network technology, and usability; Table 2 provides a comparison between the options.

TABLE 2 COMPARISONS BETWEEN DIFFERENT SOFTWARE LAYER UNDERLYING RMI

	Open source	Mobile phone support	Supported network tech.	P2P Model	usability
JXTA	Yes	No	Exc. Bluetooth	Hybrid	Medium
JXME	Yes	Yes	Exc. Bluetooth	Hybrid	Medium
JXBT	Yes	Yes	Bluetooth	Hybrid	Medium
PROEM	Yes	Yes	WLAN	-	-
OBEX	Yes	Yes	Bluetooth	-	-
SyncML	Yes	Yes	-	-	-
BlueCove	No	No	Bluetooth	Hybrid	Medium
Blue Framework	No	Yes	Bluetooth	-	-
Jadbas- CLDC	No	Yes	Bluetooth	-	-
Umbrealla	No	No	Bluetooth	-	-
Ergon- J2me	No	Yes	Bluetooth		
Peer2Me	Yes	Yes	Bluetooth. extendible	Pure	High

As we can see the best option for the software layer, underling RMI, is peer2me. Additionally, peer2me has some advantages, like providing connection with more than 8 devices. This is done through the concept that it connects to the other devices only when it wants to send a message, so in this sense, there is no limitation, in the spec of 8 devices. Although peer2me version.2 did a very good optimization, still there are opportunities, like combining the layers to improve the performance of the platform. Some of the classes could be combined, while each class file has a special overhead. We could also move listeners to MIDlet and reuse them. Moreover, using preinstalled devices more, making the naming smaller so as to make the size of the class file shorter, and try to combine the hierarchy could help to improve the performance of this framework. These methods might violate the usefulness property which was the main goal of this framework, but there is a tradeoff between these two qualities: usefulness and performance.

4. Special RMI Challenges for Phone Platform

To know the challenges encountered in implementing RMI in the new platform we should talk about what is going on behind the scene. RMI has five components: first, the client object that invokes a method of the server object, second, the server object, third, Stub, and forth, Skeleton, and the last component is Registry. There is an interface which Stub, Skeleton, and the remote object implement in different ways.

When a method of the server object is invoked, from the client object, the request will invoke a method from Stub with sending metadata. Stub has the same interface as the server object, but the content of method is the network code, to marshal, send the request to server, receive and unmarshal the response. On the other side on the server machine Skeleton is presented. This object has the same interface and does the same thing as Stub but on the server machine it has the network code for marshal, sending responses, receiving and unmarshal requests, it will deliver the request to the server object, and after receiving the result when the method is invoked, it will marshal it and send it to Stub. Stub and Skeleton are produced using Rmic compiler, and their codes are hidden from the user, and it seems the user is invoking a method on the local devices, but to know where the object and the method actually are, Registry component is used. When the remote objects extend RMIRegistry, they are tagged “Remote”, and by having special attributes, they could be registered in Registry, and further be used as a reference to the invoked object. While the client doesn’t know where these object and method are, it just contacts Registry that has the specific default port and the address, it fetches them from the device that the server object is listening to, and calls them [9].

The main requirement of RMI is serialization, in which some metadata related to the class name, class version, class hierarchy, class attribute type and class data are converted to a single string so that it could be sent over the network. In CLDC, sterilization is not built in, and each application should implement it according to their needs. the whole challenge is how to convert a complex object to a byte stream so that it is optimized for constrained devices. The complex object has a cross reference between the objects, and we should just store each object once. There are some guidelines to do so, for example, using Class.newInstance() instead of Class.forname() method. Experiments show that using nonrecersive methods can improve performance on constrained device platforms. In this way, the serialization method, uses two parameters: DataInputStream/DataOutputStream and stack. It uses them to read/write the simple data types and to maintain the references of the objects associated with the actual object. These two parameters could be avoided using a public attribute in another class; also, using some compression algorithm over serialized data could improve the performance of sending over wireless networks, and electrical power consumption in constrained devices [6].

The second problem that is encountered in this environment is a place to publish methods. We could have two proposed solutions: first, we could try to provide a standard service notification. In this sense, there is a global service signature for each service that is provided on mobile phones. When two devices become neighbors, then simply the device finds the related signature, by searching and then does Remote Method Invocations, over that. However there could be millions of methods with different naming, and this constraint makes it difficult for the service implementers. The second solution is to provide a diary on the web. But again we need to have a universal naming space in order to avoid any conflicts. This could be done by a hash function which maps the name of a person, location and other related information, to a special signature; this is a facilitator for the problem. The person who is using these services should check the results to make sure, that it is in the expected domain of output.

According to what we said previously Stub and Skeleton are produced using Rmic compiler. Thus, in order to provide RMI over mobile phones, Rmic compiler should also be implemented according to the underlying network layer. Here, we have a good opportunity for improvement. Though the previous version of Rmic was developed on Desktop and Server devices, it is not suitable for constrained devices. So it leads us to change the implementation according to the general guidelines for optimization on constrained devices [5].

The other improvement we could make for developing RMI over constrained devices is to provide persistency. This is very important in this scenario: your mobile phone needs a special service from the other phones, but phones in your proximity don't provide that. There are two solutions for this problem: first, you can use GPRS, but in this case, the cellular operator will charge you. We try to avoid paying, until we are compelled to. The solution is discussed in Roaming solution in section 5. Let's have another solution: your child and wife are going out for two hours, then your request for the service is stored on their mobile phones, and when they are walking in the park, or going shopping, their mobile phones will automatically do remote method invocation on a mobile phone in their proximity, and store the output for you, after two hours when they return home, your mobile phone will automatically fetch the output and will use it in your application. This mechanism that provides persistency needs a queuing mechanism and some independent threads in your application, to read queues and sends request to a remote object, and store the results. In order to fetch the results, also it should have a time out period to discard the outputs which are not useful anymore. We tried to discuss the challenges occurred in this new platform, each of the proposed solutions needs to be worked on, in order to check their validity.

5. The Proposed Architecture

So far, we have discussed three issues on the enhancement of distributed processing over mobile phones. The main challenge today, for developing applications over mobile phones is that, there is not enough software infrastructure, for distributed processing over mobile phones, with two qualities, usefulness, and performance simultaneously. The three main software layers that we discussed in this article could be integrated into a platform so that it helps developers to develop applications on this platform easily, and with high quality. Figure 1 shows our proposed architecture for the distributed processing enhancement.

The main goal of our architecture is to make the application independent from the underlying network technology; even roaming should be done transparently. This will also reduce the required effort to migrate the framework to a new network medium. The main goal of this approach is to use the tested and mature software layers to provide the functionality. Thus, the Network Interface, the Network Module, the Domain,

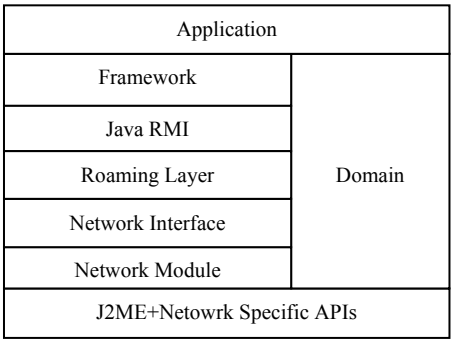

Figure 4 The Proposed Architecture for Distributed Processing over Mobile Phone platforms

and the Framework, are the layers of peer2me framework, which is a mature framework for P2P computing over mobile phones. It has also two important qualities of usefulness and performance, simultaneously. We provide two new layers, Roaming and RMI over them. This framework was previously developed for Computer Supported Collaborative Work, but we think we could change our insight, and look at mobile phones as a mature platform for distributed processing.

Applications use the interface provided by the core functionality of the framework. The framework uses a generic interface to control the technology of specific Network modules. The framework entity is the core entity and is used as an interface between the application and the rest of the system. It manages resources such as known peers and available network mediums. The technology specific network module can be implemented with the technologies such as BT, WLAN, SMS or GPRS. The Network Module and the Network Interface layers are what make the framework the chosen network technology. The bottom layer is J2ME itself along with the specific network technology APIs. The Roaming Layer provides roaming between heterogeneous wireless networks, without interrupting applications over it. In addition the application could configure the qualities in order to switch between different wireless networks, and The Roaming Layer does it transparently. The RMI layer will provide a remote method invocation, between different mobile phones.

The rightmost part of the figure labeled “Domain” contains the abstractions of the domain concepts such as: node, group, service, and messages. A node is a logical representation of a peer, in our case a mobile phone running the framework. A service is connected to a specific application and supported by zero or more nodes. The service is used to identify the part of the application shared by several users. A group is a collection of nodes providing the same services and communicating using a homogeneous network. A message is the entity that can be exchanged between nodes connected in a group. A message can either be sent to a specific node or to a group as a whole. Of the concepts described above, two are dependent on the network technology: The network and node entities. For those two, the framework uses the abstractions independent from the issues of the application and framework themselves. The group entity will in some cases require a network specific implementation, depending on the capabilities of the underlying network. Since there was no such a architecture previously developed on mobile phone, we couldn’t compared it with similar systems, our architecture provides flexibility and extensibility. It provides usability for both developer and end-user.

6. Acknowledgment

This research was created through a financial grant offered by Iran Telecommunication Research Center.

7. Conclusions

Today, mobile phones have the same capability of Desktop computers in 90's. At that time, computers were used in distributed systems and optimization was the main challenge. Today, mobile phone platforms provide an opportunity for improving distributed processing, but due to the existing constraints, most of the previous solutions should change, according to general guidelines for optimization. Although these guidelines violate some of the general principals of software engineering, we need to obey them to achieve performance on constrained devices' applications. The current solutions for RMI over the mobile phones are proxy based, and don't provide you the ability to invoke the methods of the objects on the other mobile phones. Also, the developed solutions for RMI on Desktop platform are not suitable for mobile phone platforms, while the solutions on mobile phones have different functional requirements and concerns.

In this paper we discussed emerging challenges for implementing RMI over mobile phones. There is a general guideline for optimizing applications over mobile phones; we looked at them thoroughly to concern them in our new implementation approach. Different network technologies were discussed, and finally Bluetooth was identified as the most suitable technology for distributed processing, and after that WLAN and GPRS were suitable. We also discussed different software infrastructures for facilitating the underlying layer in RMI implementation. Peer2me is identified as the most suitable platform. We discussed the challenges related to the implementation of RMI, and provided some solutions. Then we mentioned the concept of the Roaming layer, and how it could be optimized for the new platform, and finally, we provided an architecture style to facilitate distributed processing, over mobile phones.

8. References

[1] C.K.Chen, C.W.Chen, C.T.Ko, J.K.Lee, J.C.Chen, Mobile Java RMI support over heterogeneous wireless networks: A case study, Journal of Parallel and Distributed Computing, 2008.

[2] A.I.Wang, M.S.Norum, A Peer-to-Peer Framework for mobile collaboration. In The 2007 International Symposium of Collaborative Technologies and Systems (CTS 2007).

[3] Sun Microsystems, jxme: JXTA Java Micro Edition Project. http://*jxme*.jxta.org/, 2006.

[4] L.Sarjokoski, Challenges of Mobile Peer-to-Peer Applications in 3G and MANET Environments, Seminar of Internetworking 2005.

[5] Sun Microsystems, Wireless Tech Tips, Optimizing J2ME Application Size, *www*.sun.com, 2006.

[6] A.J. Sierra, Recursive and non-recursive method to serialize object at J2ME, *www.w3.org/2006/02,* 2004.

[7] T.Bjornsgard, K.Saxlund, The Improved Peer2me Framework for Mobile collaboration, Master Thesis, NTNU, 2006.

[8] S.A.Hetnes, T.Vatn, Redesign and optimization of the peer2me Framework, Master Thesis, NTNU, June2006.

[9] S.Campadello, O.Koskimies, K.Raatikainen, H. Helin, Wireless Java RMI, Enterprise Distributed Object Computing Conference, 2000. EDOC 2000. Proceedings. Fourth International.