Use of distributed resources in mobile environment

Tommi Kallonen and Jari Porras
Lappeenranta University of Technology, Department of Information Technology
P.O Box 20, 53851 Lappeenranta, Finland
E-mail: {tommi.kallonen, jari.porras}@lut.fi

Abstract: In this paper an approach to utilize distributed resources in mobile computing environment is presented. Our approach is based on the use of previously implemented mobile connectivity solution PeerHood. PeerHood observes the neighborhood of the mobile device making it possible to discover and use services of other devices (i.e. distributed resources in this environment) and thus makes it possible for our new approach to utilize these resources for remote execution of various tasks. We created a service on a fixed computer which receives tasks from other devices and executes them. Remote image analysis is used as an application to show how the approach could be efficiently used.

1. INTRODUCTION

The number of different types of computing devices and communications networks as well as their capabilities has increased dramatically during the last decade. The world around us is full of processors taking care of different tasks and communication links delivering data from one place to another. Many researchers believe that this kind of development is making possible a new computing paradigm - pervasive or invisible computing - where computing technologies will recede into the background [1]. This new computing paradigm is based on developments in many separate fields. Tiny sensors and sensor networks as well as wireless network based environments can be used as the core of the paradigm. Sensor based information as well as services in the network form the content in this paradigm. The key issue in this whole paradigm is in the intelligent use of the information of the environment. The intelligence makes it possible to offer something unique for the users. The work of Wireless World Research Forum [2] supports these ideas. The ideas of WWRF are based on the user centric model presented in the first book of visions [3]. In this model user is connected to the services and to the pervasive environment through some personal device that allows the exploitation of the distributed resources.

The variety of mobile devices and their capabilities has increased enormously during the last few years. A variety of wireless networks, like WLAN and Bluetooth, have become common in these mobile devices. These devices may be small handheld devices like PDA's and mobile phones, or laptops with state-of-the-art processing capabilities. In laboratory conditions the laptop based approaches are working but in real life the personal device need to be small enough. Therefore mobile phones and PDA devices are the key elements of the future pervasive environment. Unfortunately the capabilities, e.g. processing power and battery life, of these devices are still limited. Therefore, these devices need to take in advantage of distributed resources available for them.

In this paper we present an approach where mobile devices utilize the distributed resources in their near vicinity. This approach allows remote execution of various tasks, e.g. analysis of images. The remote execution of tasks in mobile environment is discussed in chapter 2. Our approach is based on a previously implemented mobile connectivity solution PeerHood. This environment is shortly presented in chapter 3. We have tested this approach by implementing a barcode analysis application. This application and its implementation as well as some results are presented in chapter 4. Finally we conclude the paper by analyzing some of the results.

2. REMOTE EXECUTION

Remote execution is an approach where tasks are executed in some other resource than the origin of the task. Remote execution is the basis for the distributed systems and well supported by approaches like CORBA, GRID or just by RPC. Unfortunately the widely used methods don't work so well in the mobile environments due to connectivity and bandwidth issues. Remote Procedure Calls (RPC) are popular in workstations and servers using wired connections but because of their nature they wouldn't work so well in wireless environment where bandwidth may vary greatly and there may be temporary disconnections. RPC is synchronous approach, thus leaving the client in a blocking state while in execution. With wireless network prone to disconnections this is clearly undesired. The client might have to wait a long time for the response or if the connection breaks, the response might never arrive.

In this paper we consider remote execution in mobile environment. Although there are those existing approaches we are not going to modify them for the mobile environment but rather do the necessary steps with the mobile environment restrictions in mind. Therefore we consider the following steps sufficient:

1. Sending the task over wireless network to the server
2. Executing the task at the server
3. Sending the result back to the client

Mobile environment differs from the fixed counterpart with several aspects. The nature of mobile environment is more dynamic or ad hoc than fixed environment. However, by carefully designing the approach remote execution can benefit significantly. One of the reasons for using remote execution might be for saving time since the server usually has more powerful processor and more memory than the mobile device. This would extend the usability of the mobile device in the areas where the power of the mobile device would not be enough. Executing the task elsewhere also creates some overhead from communication between the mobile terminal and server so a decision has to be made where the task should be executed. This has been addressed in [4]. Remote execution can also be used to save portable devices battery life. Since executing a task consumes a lot of electricity, battery life can be increased if the task is executed elsewhere. This of course requires that the transfer of the task consumes less energy that the local execution. Wireless transfer takes a lot of energy and therefore in many cases it would take more energy to transfer a task than to execute it locally but in some cases we can benefit from transferring the task elsewhere [5]. One reason for performing remote execution could simply be that we might not have suitable software for mobile clients so that we really have to do the execution in another environment.

Some work for remote execution in mobile environment has been previously done in different environments. Bakos et al. [6] have studied the use of mobile phones for distributed computing through different mobile networks. Their test platform consisted of a server delivering work for a static network of slaves formed by several Nokia 9210 Communicators. These phones requested work from the server, which gave them parts of Mersenne prime number calculation. Different connection types were tested using HSCSD or SMS protocols. GPRS was also discussed but not tested. Although interesting similarities exist, this paper concentrates on slow connections and thus cannot be seen as a viable approach for our purposes. This approach is also of static nature due to fixed server and networking technologies used. We are mostly interested in the dynamic use of local, ad hoc, resources. Usage of wireless links to connect desktop computers has been under research e.g. in the MOWGLI project [7]. This kind of research gives background on the possibilities and limitations of wireless links with limited bandwidth (e.g. GPRS.) This research does not use Bluetooth, so only some parts of it are applicable. Several ideas of using mobile devices as a part of GRID computing have been presented [8], [9]. In most of the cases mobile devices act in the GRID as the management devices from where the operation can be controlled and monitored [8]. Mobile devices are hardly ever used as the computational platform but some of those ideas have been presented as well [9]. Although interesting, GRID itself is still quite heavy system for mobile environment. Therefore we have implemented our own remote execution in pervasive environment on top of our local connectivity solution PeerHood.

3. PEERHOOD ENVIRONMENT

PeerHood is an environment for wireless devices where devices act and communicate in a peer-to-peer manner [10]. The devices communicate directly with one another without any centralized servers. To achieve this devices monitor their neighborhood and gather information concerning the environment for further usage. PeerHood offers a library that enables the usage of different supported network technologies through unified interface. Currently PeerHood is implemented to Linux and Symbian environments. The Linux version supports Wireless LAN (WLAN), Bluetooth and General Packet Radio Service (GPRS) networking technologies. The Symbian version is more limited and currently supports only Bluetooth networking technology. In this remote execution study we only consider Bluetooth connectivity as the idea is to remotely execute a task in some local resource found by the PeerHood environment. The basic idea and elements of our environment is presented in Fig. 1.

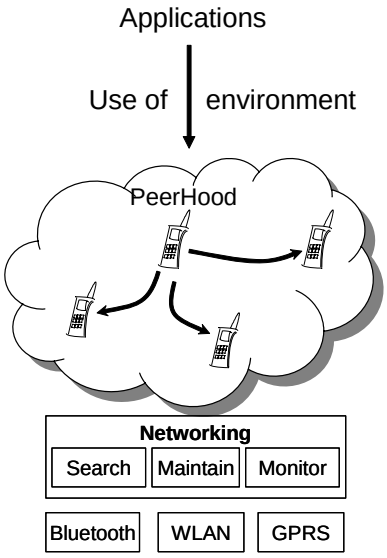

Figure 1. PeerHood structure and elements.

3.1 Functionalities of the PeerHood environment

The key functionalities implemented by the PeerHood system and necessary in this study include:

- *Device discovery* – PeerHood is able to detect other devices that support PeerHood that are in its

neighborhood. Different network technologies allow different sizes of neighborhoods.

- *Service discovery* – PeerHood is able to find out which services are available on the devices which it has detected. In this study this means that the image analysis service needs to be found from the distributed resource.
- *Service sharing* – PeerHood offers a mechanism for applications to register services. PeerHood then advertises these services to other devices. Part of the service discovery as service needs to be registered before it can be used.
- *Connection establishment* – PeerHood offers a way to connect to remote devices in the same neighborhood. Different network technologies can be used. In this study only Bluetooth is considered.
- *Data transmission between devices* – PeerHood supports data transmission between connected devices Abstract Connection interface. Used in this study to transfer task to the remote resource.
- *Active monitoring of a device* – It is possible to put a device in the neighborhood under active monitoring. PeerHood notifies the requesting application as soon as the monitored device goes out or comes to the range. Not so important in this study.
- *Seamless connectivity* – It is possible to change the used networking technology if the established connection weakens or breaks. Could be used in this study together with the previous functionality to ensure that results of the submitted task can be received even in mobile environment.

3.2 PeerHood implementations

In this study both Linux and Symbian based PeerHood implementations are used. Symbian based PeerHood is used for the mobile device whereas Linux based implementation is used for the remote resource.

Linux implementation

PeerHood Linux implementation consists of two parts, a daemon and a library. The daemon is an independent process which advertises device's services and discovers other devices using network specific plugins. The library interface offers PeerHood functionality to the application.

Symbian implementation

The Symbian implementation is not as complete as the Linux version. It was developed to test if peer-to-peer solutions can be used on mobile phones at all. The most fundamental difference between Linux and Symbian implementations is that Symbian version doesn't have a daemon to advertise services and discover devices. The Symbian implementation consists of a single package which implements all PeerHood functionalities.

4. CASE: REMOTE IMAGE ANALYSIS

Our approach was further evaluated through an image analysis problem. User takes a picture with his mobile device. In our case we used a simple bar code but the target could be even more complex. The picture was then transferred to a distributed resource found from the neighborhood of our device. This remote resource analyzed the picture and returned the content to the mobile device. Mobile device then may use this information as it wishes. This scenario is presented in Fig. 2.

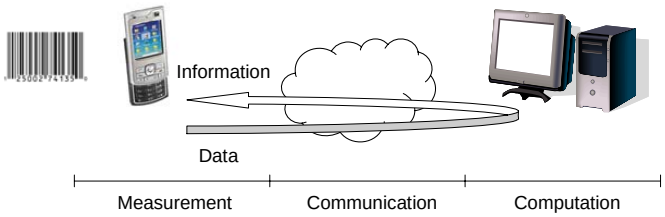

Figure 2. Bar code analysis through our approach.

The solution has two independent parts, the client running on Symbian/Series 60 based mobile phone and the server running on Linux-workstation. Both of these devices have PeerHood installed on them. The client uses the phone's camera to take pictures of barcodes. Then with a help of PeerHood it finds a server with correct service, connects to it and sends the picture there to be analyzed. After the server has analyzed the picture it sends a response containing the barcode's information content in text format. After the server is started it registers its service to the PeerHood daemon. The PeerHood daemon advertises this service in its neighborhood and server program just waits for incoming connections. After the client has created a connection with the service it sends a picture in jpeg format. The picture is sent in several packages. After the whole picture is received, the server program saves it and calls for *Bardecode* [11] program to analyze the picture. *Bardecode* program returns either the information content of the barcode presented in the picture or nothing if no content was found. Then the server sends its response to the client. It's either the content *Bardecode* found or text "No barcode found" if *Bardecode* didn't return anything.

The client

As previously stated the client software is running on a mobile phone with Symbian operating system and Series 60 platform. These phones also have cameras which can be used for example to take pictures of barcodes. The client uses PeerHood first to find its neighbors and their services and, after a suitable service has been found, to create a connection

with it. The most important classes of this solution are shown in Fig. 3 below.

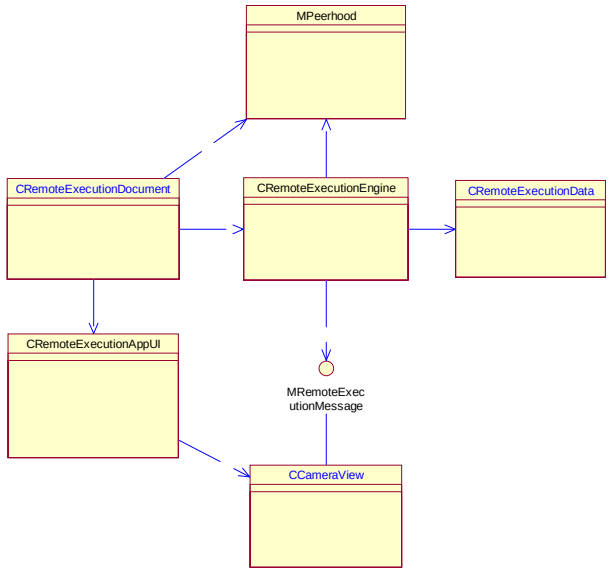

Figure 3. Structure of the client program.

The purpose of each element of the client is as follows:

- *CRemoteExecutionDocument* is a base class which starts the whole program and creates other objects.
- *CRemoteExecutionAppUI* holds all the views, which actually create the different user interfaces for this program.
- *CCameraView* is the most important view. It works as a user interface where user can take pictures and send them to the server to be analyzed.
- *CRemoteExecutionEngine* uses the PeerHood interface and handles connections and data transfers.
- *CRemoteExecutionData* holds all the data needed to transfer a task and handles packaging/unpacking before and after data transfer.

The server

The server is running on a workstation with Linux operating system and running PeerHood daemon. When the server software is started it registers its service to the PeerHood. PeerHood daemon then starts advertising this service to all PeerHood enabled devices in its neighborhood. Clients can now connect to the server and after a connection has been established, they can send their tasks, i.e. their pictures to be analyzed, to the server. The task data is sent in several packages (due to the PeerHood implementation) and after all the packages have been received and unpacked, the server does the work client required. After the work has been completed, the server sends a response to the client containing the result of the task. Structure of the server is shown in Fig. 4.

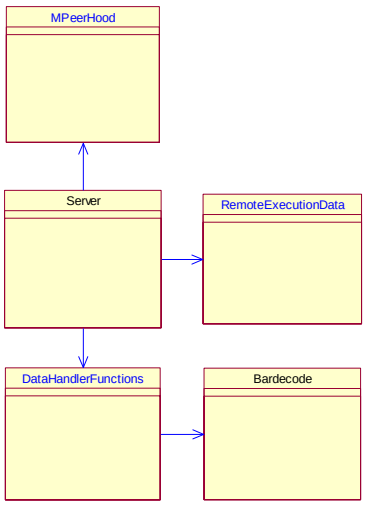

Figure 4. Structure of the server program.

The elements of server are defined as follows:

- *Server* is the main program from where the server is started. It uses the PeerHood interface to manage connections.
- *RemoteExecutionData* holds the data needed for a certain task just like in the Symbian implementation. It holds all the data and it takes care of packaging and unpacking the data for/after a transfer.
- *DataHandlerFunctions* is a set of functions used to do the actual tasks for the received data. One of these functions is used to call Bardecode to read barcodes from image data.
- *Bardecode* is software for reading barcode information from image files. It can interpret several types of barcodes from image files in jpeg or tiff format. In this system we used the command line version of *Bardecode*. It reads an image file and returns the barcode information content it possibly finds from it.

Testing

The system was tested by using Nokia 6630 mobile phone as a client and a workstation running Ubuntu Linux as operation system. *Bardecode* software comes with a program to create images of barcodes with desired information content. Different barcodes were created with this program and then printed onto a paper to use as test barcodes. The steps needed to read barcode data using our system are shown on picture 5.

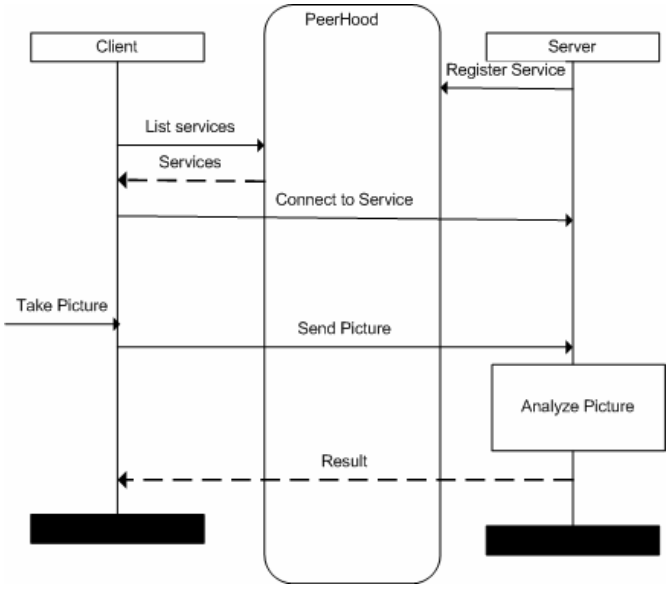

Figure 5. Steps needed to analyze a picture.

1. Server program is started
2. Server registers service to PeerHood
3. Client program is started
4. Client starts PeerHood and searches for neighbors and their available services
5. After a suitable service is found, a connection to it is created
6. A picture of a barcode is taken
7. Picture is sent to server
8. Server uses *Bardecode* to analyze the received picture
9. Server returns the information content *Bardecode* possibly found

The most time consuming part of this process is the searching for devices which can take up to several minutes using PeerHood over Bluetooth. This is mostly due to the Bluetooth inquiry process that takes time. This was a problem with the Symbian implementation of the PeerHood. Since the Symbian implementation doesn't use a daemon, the discovery operation starts after the client program is started. If a daemon was used, like in the Linux implementation, it would always have a list of neighbors from its previous search and we wouldn't have to wait for the result. After a connection to a service has been established, the transferring of picture, analyzing it, and returning the result happens fairly quickly. Transferring a 640*480 picture in jpeg format takes on the average about 670ms, the analysis of the picture takes just 50ms and returning the result 100ms.

The biggest problem with reading the barcode has been the picture quality of the mobile phone's camera since the camera doesn't take good quality pictures from a short distance. While taking a picture of a normal barcode the picture would have to be taken from just a few centimeters distance for the barcode to be large enough in the picture. Unfortunately the camera doesn't focus on targets that close very well and the picture becomes unclear. And when the picture is unclear, *Bardecode* software can't interpret the barcode from the picture.

To solve this, large test barcodes were printed in sizes up to A4 sheet of paper. When the barcodes were this large their reading worked quite well. The barcodes were read correctly as long as the pictures were bright enough. With poor lighting conditions the pictures became dark and *Bardecode* failed to interpret barcodes from the pictures.

5. CONCLUSION

Remote execution in mobile environment is a true opportunity as mobile devices in general are limited in processing power and other resources such as memory. The use of currently available broadband technologies like Bluetooth and WLAN make it possible for the mobile device to locally connect to distributed resources, fixed or mobile, and utilize the capacities of the environment. This allows the usage of the pervasive environment in an efficient way.

In this paper we have presented one approach to use the distributed resources for remote execution of tasks. Our approach is based on the use of PeerHood - a networking solution for mobile and fixed devices. The application, i.e. image analysis, is relevant although simple as only barcodes are analyzed. PeerHood is used on the server side to create and advertise a service and on client side to discover the service and create a connection with it and finally to use the service. The remote execution works quite nicely. Image analysis takes time depending on the algorithm we are running. The communication through PeerHood and especially through Bluetooth interface in not the most effective as it was not defined to move big files. The most problematic part is definitely the picture quality. It seems that in our case the camera of the mobile device wasn't of the best quality and thus affects to the conditions where pictures can be taken. A better camera or some preprocessing of the image should solve this problem.

Remote execution in mobile environment is working as this study shows. However, there are lots of aspects that need to be improved and added to make this approach a successful one. First of all the quality of mobile cameras need to be improved. As a matter of fact the quality has already improved as new phone models have been shipped. Secondly the communication layer of PeerHood needs to consider this type of application better. Finally, the interfaces to the remote resources and their algorithms need to be defined in more

general way. In order to use remote resources in ad hoc manner, services need to be defined unambiguously.

REFERENCES

[1] Weiser M.: The Computer for the 21st Century, Scientific American 265, No. 3, 94-104 (September 1991).

[2] Wireless World Research Forum homepage, http://www.wireless-world-research.org/

[3] The Book of Visions, Wireless World Research Forum, pp. 275, 2001.

[4] Gitzenis S. and Bambos N.: Mobile to base task migration in wireless computing. Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications, 2004.

[5] Rudenko A, Reiher P, Popek GJ and Kuenning GH. Saving portable computer battery power through remote process execution. ACM SIGMOBILE Mobile Computing and Communications Review January 1998.

[6] Bakos B., Fodor S. and Nurminen, J. K.: Distributed Computing With Mobile Phones: An Experiment with Mersenne Prime Search, Pervasive 2002.

[7] Alanko, T., Kojo, M., Laamanen H., Raatikainen, K., Tienari M: Mobile computing based on GSM: The Mowgli approach. IFIP'96 World Conference - Mobile Communications, Canberra, Australia, September 2-6, 1996.

[8] Karppinen J., Niemi T., and Niinimäki M.: Mobile analyzer - new concept for next generation of distributed computing. 3rd IEEE/ACMInternational Symposium on Cluster Computing and the Grid, (CCGrid 2003), Japan, 2003.

[9] Sang-Min Park, Young-Bae Ko, and Jai-Hoon Kim: Disconnected Operation Service in Mobile Grid Computing, First International Conference on Service Oriented Computing(ICSOC'2003).

[10] Porras J, Hiirsalmi P, Valtaoja A. Peer-to-peer Communication Approach for a Mobile Environment, 37th IEEE Annual Hawaii International Conference on System Sciences, 2004, ISBN 0-7695-2056-1.

[11] Bardecode program homepage, http://www.bardecode.com/