Portable Information Devices – Service Discovery

J. Kadirire
Anglia Ruskin University
Victoria Road South
Chelmsford CM1 1LL

Abstract

This paper looks at how portable information devices (PIDs) can automatically discover services online and use those services. A specific service example used in this research is a printing service, although the service can be anything that one cares to define. Quite often, PID users receive documents via emails which they can access directly or notification of some documents like word documents or via text messages. Their PIDs may not have the capability to download and display those documents and if they need to read those documents urgently, one possible way is to download and print them in their vicinity. For this to happen, the PIDs need to be able to discover services like a printing service and the Jini technology makes this possible by providing spontaneous networking.

I. INTRODUCTION

The Internet and the World Wide Web (www) have been growing at a phenomenal rate and the world is slowly turning into an e-World, where most things are done online. With 1.52 billion mobile users world wide, as of September 2004 [1], mobile devices like mobile phones and portable information devices (PIDs) are now ubiquitous and access to services online via PIDs is common place. PIDs and mobile phones are sometimes indistinguishable in their functionality and are being put into use in different aspects of our lives, not least making phone calls, accessing online resources, sending messages and even being used at conferences [2], and in education for e-Learning[3]. However, more needs to be done in terms of service discovery.

Although PIDs are ubiquitous, with many benefits to users, they present a new set of difficulties for developers [4]. One difficulty is connecting these devices into a network. Small consumer devices demand a flexible, resilient networking architecture so that consumers can use PIDs to exchange information with existing networks and other computers. Sun Microsystems have introduced a java runtime environment called KVM, [5] for embedded devices, and the Jini technology [6], which provides a flexible, resilient network architecture to easily link together small computer devices onto a network, making application portability and code reuse possible.

Quite often, PID users receive documents via emails which they can access directly or notification of some documents like word documents or html documents via text messages. Their PIDs may not have the capability to download and display those documents and if they need to read those documents urgently, one possible way is to download and print them in their vicinity. For this to happen, the PIDs need to be able to discover services like a printing service and the Jini technology makes this possible by providing spontaneous networking.

The idea of spontaneous networking with Jini is that if a Jini enabled printer is plugged into the network, it automatically joins the network. When a client joins the network, it automatically discovers the Jini enabled printer and can make use of that printer without installing any software or configuring the printer. If the client needs any software to use the printer, the software will be automatically downloaded from the printing service [7],[8]. Any network resources needed by the client to use the printer will be automatically provided and as soon as the client's job is printed, the network resources are again automatically reclaimed and freed for use by other clients.

Jini is making this need to easily access and use network resources from mobile devices realised, by introducing new network capabilities like spontaneous networking, automatic configuration via protocols and application programming interfaces (APIs) that enable more robust and reliable distributed systems to be built [9]. Here is a scenario for spontaneous networking. Currently, if one needs to print a document, a network printer needs to be connected to the network, device drivers need to be installed and configured and permission probably needs to be obtained from the network administrator to use the printer. The idea of spontaneous networking with Jini is that if a Jini enabled printer is plugged into the network, it automatically joins the network. When a client like a PID joins the network, it

automatically discovers the Jini enabled printer and can make use of that printer without installing any software or configuring the printer. If the PID needs any software to use the printer, the software will be automatically downloaded from the printing service [7], [8]. Any network resources needed by the PID to use the printer will be automatically provided and as soon as the PID's job is printed, the network resources are again automatically reclaimed and freed for use by other clients.

The rest of this paper is organised as follows: Section II describes the design of the Printing Service for the Palm V PID and how the service discovery is accomplished. Section III looks at the results of doing a service discovery and printing a document. Sections IV and V look at the discussions and conclusions, respectively.

II. PRINTING SERVICE DISCOVERY BY THE PALM V PID

Increasingly, online services are becoming more personalised and mobile. Fig. 1 shows the architecture of the Jini Printing Service for the Palm V PID. It consists of a distributed Jini network (Djinn) which uses a lookup server, a Palm V PID coupled with a proxy device to form a client, and a printing service. The Palm V has a small limited java virtual machine called the KVM, running on it and this KVM only supports basic sockets and input/output streams. This necessitated the need to develop the proxy which sits on the network and is Jini capable.
So, the printer service defines an interface which it registers with the lookup server. The proxy receives commands from the Palm V PID to print some data using a basic private

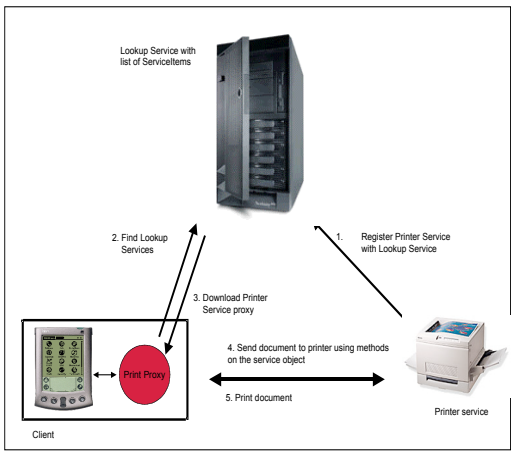

Fig. 1. Network for the Palm V PID Printer Service

protocol. The idea of a proxy that's Jini capable is very powerful as it allows mobile or indeed any network devices, with or without a java virtual machine to use the Jini protocol and APIs. When the proxy receives the data to print, it stores the data into a file on its local file system, and then proceeds to do multicast discovery [9] to look for lookup servers on which the printing service is registered. When it finds the lookup server(s), it downloads the service proxy which enables it to talk via RMI [10], to the backend printer service. This proxy object has methods defined on it, e.g. print() and by invoking this print method, the file is downloaded and printed on the service side using the default printer on which the printing service is running. When the printer service proxy is downloaded by the Palm V PID proxy, the lookup server is effectively out of the picture and communication takes place directly between the Palm V PID proxy and the Printer Service via RMI. The printing service is always registered on the lookup servers by making sure that a lease [11] is granted at the point of registration and constantly renewed before it expires.

III. RESULTS

To start the Printer Service, a number of things had to be done in the following order.

1. The RMI daemon was started first.
2. The web server from which the java classes were served when requested by the lookup service, was also started.
3. The Jini Spaces browser or just the basic Jini browser which enabled one to detect what services had been registered with reggie, the lookup service, was also started.
4. The Sun Microsystem's implementation of the Jini Lookup Service called reggie was then started. Reggie was needed to enable other services to do a discovery, join and lookup.

When reggie, the lookup service was running, it was detected by the Jini Spaces browser. Because of the way Jini does its multicast discovery, it sometimes took a few minutes before the reggie lookup service was detected by the browser.

After reggie was running on one's local host and perhaps other hosts on the network, the Printer Service was then started. After compiling the java source code for the Printer Service, the code for the Printer Service and the Printer client was kept in two different directories. This was necessary to avoid

problems with dynamic code downloading of the printer proxy, at runtime. The command to run a Jini service can be quite long and difficult to remember. Therefore, all the commands were put in a batch files. Below is an example of how to run the Printer Service.

```
C:\APACHE\HTDOCS\packages\service-dl>
java -classpath c:\jini1_0_1\lib\jini-core.jar;
c:\jini1_0_1\lib\jini-ext.jar;
c:\jini1_0_1\lib\sun-util.jar;.;
c:\apache\htdocs\packages\service
-Djava.rmi.server.codebase
=http://kings.ima.bt.co.uk:8090/
 -Djava.security.policy=c:\java.policy
uk.co.bt.ima.jiniPrinterService.PrinterService
```

When the Printer Service was started, it went off and did discovery to locate the lookup services on the network and registered its printing service. Below is an example of the Printer Service running.

```
Discovered Lookup Service: jini://kings:4160,
Service ID = bb69a534-c187-41a3-bb6e-
9d4c98b2c624
PrinterService:
got 1 LookupLocator(s)
Registered the Printer Service with the reggie
lookup service.
Printer Service ServiceID:  f25d0293-3be3-
46c0-999e-dc4bf12c66cf
```

The Printer Service is now running.

When the Printer Service was running, it was detected by the Jini Spaces browser. Fig. 2 shows a snap shot of a Jini Spaces browser, which has detected 2 services running i.e. the lookup service called Reggie and the Printer Service.

By clicking on the PrinterService line in the browser, more information i.e. the printer

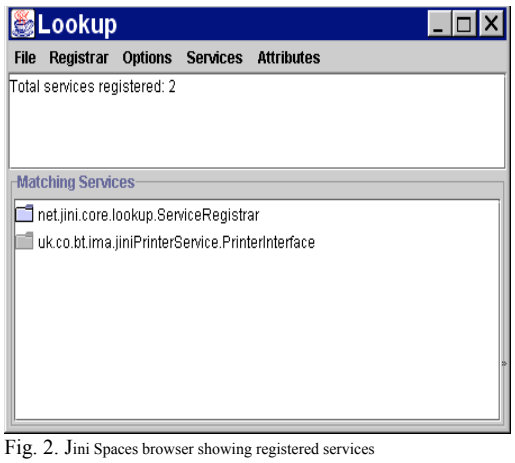

Fig. 2. Jini Spaces browser showing registered services

Fig. 3. A snap shot of the Printer Service Attributes

service attributes can be seen as shown in the Fig. 3.

Fig. 3 shows some of the printer service attributes. Some of the attributes are the service name, manufacturer, model, serial number, location, etc. If there is more than one printer service registered with a lookup server, then by browsing the attributes, a client like a PID can make an informed choice as to which service to use. For example, the location of the printer might be quite important as to which service is chosen.

Fig. 4 shows the interface on the Palm V PID that the user saw. A connection had to be established first, by clicking on the "Connect to Proxy" button. The data was loaded using the load button and sent to the Proxy for printing by clicking on the print button.

Fig. 4. A snap shot of the Palm V with its GUI for talking to the Printer Service

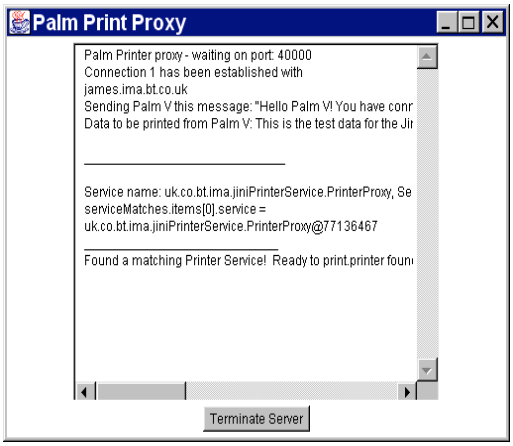

Fig. 5. A snap shot of the Printer Proxy as it interacts with the Palm V

Fig. 5 shows some of the interaction between the Palm V PID and the Proxy, and between the Proxy and the lookup service. When the Proxy received a print request from the Palm V PID, it initiated the discovery process to find the required service and printed out in the text area, the services found.

Fig. 6 shows the status of the Palm V PID GUI after the job was printed successfully. A message was sent back from the proxy to signal successful or otherwise, printing of the submitted job. At this point, the user can walk to the printer and collect his/her printout and pay for whatever charges are levied by the service provider. That's assuming one's perhaps at an airport or in a shopping mall that has internet access.

IV. DISCUSSION

Portable information devices are revolutionising our lives from making phone calls, to playing music on them, to using them as cameras, remote control devices in the home as well as for internet access. The

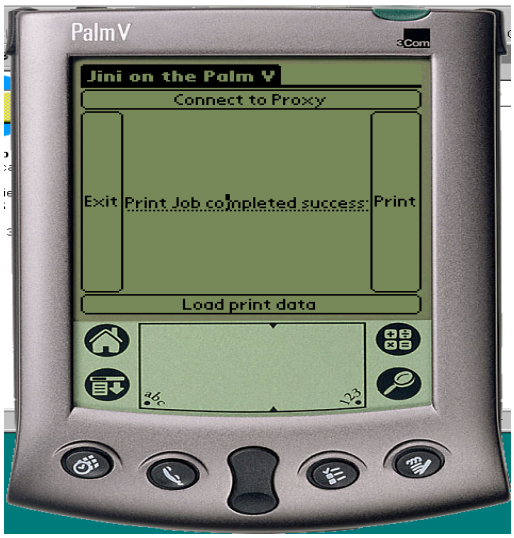

Fig. 6 Palm V PID GUI showing status of the print job

results in this research have shown that they are not only being used to access and download information online, but they can also be used to discover federated services wherever they may be defined and registered on the network, without prior knowledge of their existence. A service can be anything that sits on the network and is ready to perform a useful function. Hardware devices, software, communications channels, etc, can be services.

A basic PID can now be used to discover a whole range of location based services like downloading a menu in a restaurant, downloading a train/air line timetables, bank account details like a bank statement, etc, etc. In the case of the printer service discovery in this research, there is no need to install printer device drivers on the PID, and in any case that would not be possible because of the PID's limited capabilities and resources like memory, screen resolution, etc. The underlying Jini technology ensures that the network resources are only used when needed by using the idea of leasing, after which time, the resources are freed and made available for other users.

V. CONCLUSION

This research has demonstrated how online services like the printing service can be discovered and used by a Palm V PID using the Jini technology, to add to the increasing wealth of applications that can be used on PIDs. This is achieved via the use of the Jini technology on a PID which has very limited Java capabilities, memory, screen resolution, etc, by utilising the idea of a proxy. Documents like emails, faxes, web pages, etc, can be downloaded on the move and printed to a local printer without having to do any printer configuration.

REFERENCES

[1] "Latest Mobile, GSM, Global, Handset, Base Station, & Regional Cellular Statistics" http://www.cellular.co.za/stats/stats-main.htm

[2] Kadirire, J. The Short Message Service(SMS) for Schools/Conferences, Recent Research Developments in Learning Technologies (2005), (Vol. 2, pp. 856-859). Retrieved June 27, 2006, from http://www.formatex.org/micte2005/4.pdf

[3] Kadirire, J. Learning with Mobile Devices - A Microportal Design Experience, Recent Research

Developments in Learning Technologies (2005), (Vol. 2, pp. 792-797). Retrieved June 27, 2006, from http://www.formatex.org/micte2005/7.pdf

[4] Venners B., Objects and Java Seminar – Objects and Java Spaces. Retrieved September 15, 2006, from http://www.artima.com/javaseminars/modules/Jini/index.html

[5] K Virtual machine. Retrieved September 15, 2006, from http://java.sun.com/products/kvm/

[6] The Jini Technology Glossary, Retrieved September 15, 2006, from http://www.sun.com/jini/specs/

[7] Edwards W. K., Core JINI ISBN 0-13-014469-X, published by Prentice Hall

[8] Horstmann C. S. and Cornell G., Core JAVA Volume II – Advanced Features. ISBN 0-13-081934-4, Chapter 5, Remote Objects pages 255-317, published by Prentice Hall.

[9] The Jini Discovery and Join Specification, which can be obtained online from http://www.sun.com/jini/specs/

[10] The Java Remote Method Specification which can be obtained online from http://java.sun.com/products/jdk/1.2/docs/guide/rmi/spec/rmiTOC.doc.html

[11] The Jini Architecture Specification, which can be obtained online from http://www.sun.com/jini/specs/