Service Discovery

By

Ahmad Syazwan Shaharum

Dissertation in partial fulfillment of
the requirement for the
Bachelor of Technology (Hons)
Information Technology

JULY 2005

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

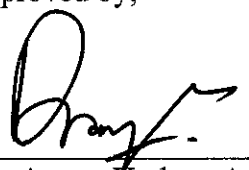
Service Discovery

by

Ahmad Syazwan Shaharum

A project dissertation submitted to the
Information Technology Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION TECHNOLOGY)

Approved by,



(Mr.Anang Hudaya A.Amin)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

July 2005

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

AHMAD SYAZWAN SHAHARUM

Abstract

Service discovery is necessary to enable computers to interact with each other through heterogeneous wired and wireless networks in a seamless way. To discover a service, a client uses one of these protocols to issue a query to a central server or an individual service provider. The service description in the query may contain a specific name and set of one or more attributes. The server or provider attempts to match the query's pattern with the pattern of a service its database contains, then it returns the appropriate response to the client. For this project the methodology use will be the Rapid Development Application methodology that is found most suitable assist in this project. Requirement analysis has been done in order to seek for the characteristics and areas that need to be put in action in getting the result. Next will be the design phase where the process to design the service-oriented and Jini architectures within the client/server architecture. For the implementation phase, both the architectures are going to be set up simultaneously using Jini Network Technology. Therefore with developing service discovery with Jini technology, the minimum of manual intervention which it is include in the Jini vision.

ACKNOWLEDGEMENT

Throughout this project entitled Service Discovery is done with the help of many people. I would like to express my gratitude to Allah, The Al-Mighty, for giving me the strength, ability and feasibility in carrying out all the research and tasks required.

I would also like to express my highest appreciation to my project supervisor, Mr. Anang Hudaya Muhd. Amin for the time he spent in providing me guidance and good advices throughout my final year project. Thanks to University Technology of Petronas in general and the Information Resource Center management staffs specifically for their helping hands in assisting me in conducting and finding the good research materials.

Not to forget, I would like to thank my colleagues, Suhana Suarni, Mohd Faitz Adam, Mohd Azizi Ahmad, Mohamed Ferdaus Abdul Wahab, and Khairil Anuar Othman for give me the energy and ideas to me in developing this project as the first in University Technology of Petronas.

Lastly, I would never forget and thanks too, to my family, all the lecturers involved, and all my friends who have been giving me continuously supports from the very beginning of this project kick-off upon the completion.

Thank you.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Broadly, a service discovery is a collection of protocols for developing dynamic client/server applications. Client/server is nothing new, and in fact many of the concepts in service discovery are not ground breaking. Instead, the advantages provided by service discovery arise because a number of important features are bundled and standardized.

Highly dynamic interactions between clients and services is the norm service discovery-enabled clients seek needed services based on their type (e.g. printing, file storage) and based on descriptive attributes that identify the manufacturer, the cost of using the service, or other interesting facts. These technologies allow services introduced into a network to be discovered, configured and used with a minimum of manual intervention. Enabled services announce their presence when they enter the network and (if possible) announce their demise when they leave the network. Applications which display a client in need of an IP address, for example might seek a DHCP server, communicate with the server, and finally be granted one already exist in networked environments.

Basically, these clients and services are typically developed from scratch, with little support for the developer other than "here's a C compiler. Get to work." Service discovery technologies generalize and standardize the environments in which client/server applications are developed and used, and implementations of the various service discovery protocols provide software tools that make the development effort much easier and interactions between clients and services more dynamic than usually seen in today's systems.

There is a lot of common group among the various choices in service discovery frameworks. All support the concept of client and service which are simply entities that need and offer some functionality (e.g. printing), respectively. Clients perform discovery in order to find needed services. A discovery attempt generally classifies the services by type, and may optionally include requirements such as a manufacturer, serial number or other services attributes. Whether services are sought directly or a catalog is consulted, a client needs a very little information about its environment which it can locate services dynamically, with little or no static configuration. When a service enters the network, it will perform service advertisement, either directly to clients or to one or more service catalogs. The advertisement includes necessary contact information, and will also include either descriptive attributes or information that will allow these attributes to be discovered.

1.2 Problem Statement

Connecting the needy clients and the providers' available services is the point of service discovery. Service discovery technologies directly attack the "I don't know where you are" and "I don't know how to talk to you" issues.

One problem user's encounter with increasing frequency and severity is the installation, configuration, and management of peripherals. This is a client/server problem in disguise. The complexity of this problem is becoming more serious as laptops, handheld computers, printers, scanners, wireless devices, external storage devices, digital cameras and other peripherals are integrated into networked environments in the home and office. Inexperienced computer users may become quite frustrated in dealing with the configuration and interaction of such a multitude of devices.

Introducing a new device can include physical installation, removing device drivers for a device being replaced, installation of devices drivers included with the new device, determining that the included device drivers are outdated or designed for a

different operating system, necessitating downloading and installing new device drivers, and unexpected interactions with existing devices. Concrete evidence of user's frustration with these situations can be seen in the red faces and angry tones of users. Configuration headaches can be compounded many times over in environments with numerous computers. Service technologies take an important step towards eliminating manually installed device drivers, relying instead on standard interfaces to put devices in touch.

1.3 Objective and Scope of Study

Following are the objectives of service discovery protocols:

- Discovery. The ability to find a service provider if there is a service in the network with the properties described in the request. In order to achieve this goal, protocols implement the following functions:

  - Use a description language. To facilitate the discovery process, services are semantically described following a certain description language. Service requests are also expressed using this description language.

  - Search for services. Service requests are expressed using the description language and are addressed to the directory nodes or disseminated in the network.

Service discovery can make things more convenient by allowing the types of available services to be discovered easily. It is the needed services may be discovered on demand, with minimal prior knowledge of the network.

Typically, to discover a service, a client uses one of the protocols to issue a query to a central server or an individual service provider. The service description in the query may contain a specific name and set of one or more attributes. The server or provider attempts to match the query's pattern with the pattern of a service its database contains, then it returns the appropriate response to the client.

CHAPTER 2

LITERATURE REVIEW AND/OR THEORY

Service discovery is a field which that need an exploration with the purpose to view the ability of its function in the network world. There are many researches which applied the concept of service discovery in their project such as in ubiquitous computing.

This sounds very visionary and will probably remain a dream for a long time to come. On a more modest scale, [8] says that:

"One of the visions of ubiquitous computing is the ability to use arbitrary devices such as cell phones and hand held computers to interact with arbitrary remote networked appliances such as TVs, printers, and EKG machines."

This is more close to the project; this is where service discovery comes in. An article with good coverage of this subject, [9], says:

"Future computing environments will consist of a wide range of network based appliances, applications and services interconnected using both wired and wireless networks. In order to encourage the development of applications in such environments and remove the need for complex administration and configuration tasks, researchers have recently developed a range of service discovery and interaction platforms. The key function of such service location and device interaction technologies is to allow users and applications to deploy, discover and interact with the services provided by devices and software components on the network. This interaction is required to occur without the users, the applications, or the service providers needing detailed knowledge of the local network configuration."

Due to this, it gives all the rationale needed to motivate investigations into service discovery where it shows that the author current understanding about the topic is in the right path. This is actually motivates to continue the research and explore this topic further.

The tools that available for service discovery are many and one of them is Jini Network Technology. Each of the tools available has their own specialty and vision.

From the official Jini architecture specification, Jini is defined as:

"A Jini system is a distributed system based on the idea federating groups of users and the resources required by those users. The focus of the system is to make the network a more dynamic entity that better reflects the dynamic nature of the workgroup by enabling the ability to add and delete services flexibly."

Jini is a service discovery technology based on Java, developed by Sun Microsystems. It has platform-independent which it can rely on mobile code as well. Jini Network Technology can be phrased as a powerful tool for this topic and the usage of this tool might help to the successful product in the end of the day.

CHAPTER 3

METHODOLOGY/PROJECT WORK

3.0 Rapid Application Development

Rapid Application Development is the methodology chosen for this project due to the time constraints and level of difficulties which might happen in the middle of accomplishing it. Extreme programming will be applied simultaneously. Each phase in this methodology will be explained in detail below:

3.1 Requirement Analysis and Planning Phase

This phase is where research about this topic takes place. The idea is to get the knowledge and improve the understanding about service discovery in order to continue to develop this project. The task while doing the research starts with identifying and listing down all the possible entities, applications or technologies such as Corba, Soap, Jini and others in order to select the most suitable entities that are going to be used to develop this project. Besides, the most important thing is to study and understand well about service discovery architectures.

Service discovery architecture is actually a little bit different with other network architecture. Inside the architecture, consist of a unique entity which varies with other network architecture. The reason to study and grasp the idea of the architecture will lead to the development of service discovery. After all the architectures from all the application such as Corba, Jini and others have been identified and analysed, only the most suitable architecture will be selected for this project. The next task that needs to be done is to identify and list down all the possible and available tools that can be used in service discovery. The most preferable will be tools which are free and easy to download from specific websites. The final task is to select the most suitable tools that is going to be used in next phase.

3.2 Design Phase

Design phase will focus on the service discovery architecture. The concept of network that will be applied in this project will be the client/server architecture. This concept will also be applied simultaneously with the architecture mention before. The tools chosen for this project is a tool from Sun Microsystems, Jini. The decision to use Jini as the application for this project is based on the comparison that has been done on several other protocols and application available. The comparison can be viewed in the Appendix A.

The reason to select Jini as the application or middleware for this project is because the central vision of Jini is the realization of a distributed computing environment that can support rapid configuration that will allow the configuration of devices and software being amended using a simple "plug and play" model. Essentially, the goal is to allow any device or a software component to be connected to a network and announce its presence. Other components that wish to make use of it can then locate it and call it to perform tasks. The reconfiguration takes place invisibly, reducing the administration load of the programmer developing the system.

Three main concepts are brought together in the formation of a running jinni system. Each of these different classes of component is important to the realization of the Jini model of distribution.

- A **Service** is a piece of independent functionality that is made available to the others users and can be accessed remotely across the network. Services include devices (e.g. printers and cameras) and software components (e.g. file systems). Jini services are routinely managed as a co-operating set known as a Jini community.
- A **Client** is a device or software component that would like to make use of a service. Jini seeks to support a very heterogeneous selection of clients embracing a wide variety of hardware devices and software platforms.

- A Lookup service helps clients find and connect to services. The lookup service acts as a broker between the needs of the client and the services it knows about across thee network.

These central components combine with simple principles to form the foundation of Jini. These three principles ensure that Jini services can spontaneously interconnect with each other without cumbersome administration. These principles are embodied within Jini in term of how Jini makes use of the concept of client, service and lookup service.

The first architecture that needs to understand is the service-oriented architecture. This architecture takes the existing software components residing on the network and allows them to be published, invoked and discovered by each other. SOA allows a software programmer to model programming problems in terms of services offered by components to anyone, anywhere over the network.

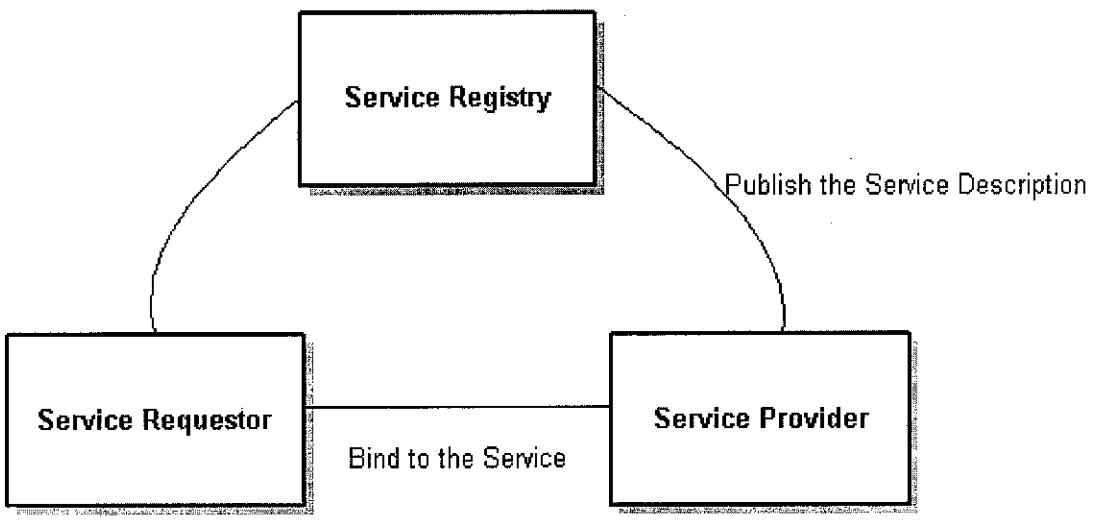


Figure 1: Service-oriented Architecture

Below are the functions of each entity:

Service Provider

The service provider is responsible for publishing a description of the service to the service registry. Normally, the service provider hosts the web service.

Service Registry

The service registry is a repository that provides the capability of discovering services by the service requestors.

Service Requestor

A software application that is responsible for discovering and invoking the service. The service requestor binds to the service obtained from the service registry.

Next architecture is the most important architecture to apply in developing the project as it been called as Jini architecture. As Jini sytem is been implemented in this project, it needs to use its own architecture which comprised of Lookup service, Jini service and Client service. It may notice the diagram of the Jini architecture closely resembles the architecture of Web services. This is another reason why the Jini architecture has a foundation that is rooted in the dynamic principles of a SOA. Below are the functions of each entity together with the architecture.

Lookup Service

The lookup service keeps track of the Jini services and provides the proxies to communicate with the service. In addition, the lookup service is a Jini Service as well. The Sun reference implementation of the Lookup Service is reggie.

Jini Service

Jini Service are registered with the lookup service and are capable of being invoked through their public interface which is defined via a Java remote interface. The underlying system that allows Jini services to communicate is RMI.

Jini Client

The Jini client is software that requests the proxy from the lookup service in order to invoke the Jini service.

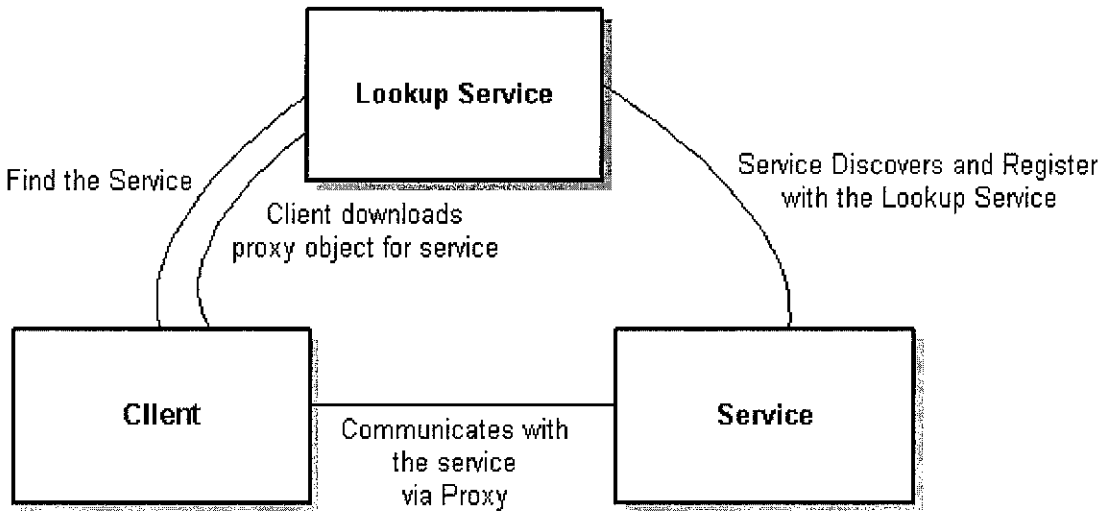


Figure 2: Jini Architecture

3.3 Implementation Phase

This is the phase where both the architectures are going to be set up simultaneously using Jini as the tool selected in the earlier phase. Programming part will also take place. It involve in programming a program which will act as a service in this project. The programming language used is Java so that the program is compatible with Jini which also a product of Java Microsystems.

The development starts with required Jini services are up and running. It has its own setup and configuration to run Jini. In order to make sure that the environment has been setup accordingly, here is a checklist for getting setup generally.

- **Run** the **RMI** activation daemon. It should run this on the same machine that will be running the lookup service on, and the activation daemon should be started before the lookup service.
- Run a Jini HTTP server which will supply downloadable code for the JIni lookup service; this server can be run on the same machine as the lookup service. This ensures that the code is able to download the classes in the reggie-dl.jar file, which contains the code needed to use the lookup

service. In addition, all the programs wrote before need to be able to supply code to Jini services and to each other.

- Run the Jini lookup service. The service should run on a machine on the same network subnet on which it intend to run the programs. This is because Jini's multicast discovery protocols are by default, only configured to find lookup services running 'nearby' the object doing discovery.

This configuration of services is sufficient to run most Jini services. While all these programs, lookup services, HTTP servers, activation daemon, and the clients and servers can be run and need to develop on the same machine, and Jini applications are intended to be distributed and run across a number of machines.

Next step is Jini deployment where it has three Jini-aware applications running:

- A Jini lookup service that helps clients locate service
- The Jini service under development that have been tested
- A Jini client program that uses the service under development

There are few steps to follow in order to run the project and the aim is to illustrate the principles involved in developing a Jini service. Firstly is to define a service interface where it defines what the service will do. It is essentially the way in which services are described to Jini and made available to potential users. The service proxy object implements this interface and the client uses it whenever it searches the lookup services.

Secondly, publishing the service interface with a lookup service and to make it available to the Jini framework in order to allow potential clients to find and use the service. A program is needed wwhich it handles finding a lookup service and publishing the proxy. This is normally called the 'wrapper' process and takes care of the interactions with Jini needed to publish the proxy.

Next is discovering a service via a lookup service where a thing needs to be done is to find a jinni lookup service. It means to make a request to the Jini framework for lookup service. The principle way of doing is to use the LookupDiscoveryManager class. This class will aalow to find instances of the lookup service that are nearby the network. Run the lookup service that is configured to support that group name somewhere in the network is a must.

Continue with the next step which registering with a lookup service. The process of registration with a lookup service which will call two arguments which are the service items and second is the requested duration of the ease. After registering the service proxy, the wrapper checks to see if this is the first time has been registered. If it is the first time, it writes the service ID that has been returned into the service item so that future registrations use the same ID. It also stashes the registration result into a hastable so that it can retrieve it in the future. Final part is to compile and run the programs and see whether it show success or its failure.

At the same time, error checking will also takes place in order to check the functionality according to the requirements. After examined the prototypes, final product will be released.

3.4 Cutover

The final phase in this methodology consists of the product is ready to be launched. Then it ready for handover the project to the university, besides report preparation and presentation.

3.5 Tools /Equipment Required

In this project, the main tool that going to be used for the coding is using the application that are available on the internet and it is free to download. Below are the list tools and equipments that have been used for this project development process.

3.5.1 Software Requirements

- Jini Network Technology by Sun Microsystems
    - Jini 1.1
    - Jini 1.1.2
    - Jini 2.1 Beta 2
- Java Development Kit version 5 (JDK-5)
- Java2 Enterprise Edition Software Development Kit version 4 (J2EESDK-4)

3.5.2 Hardware Requirements

- Personal Computer / Desktop
    - Window XP
    - Pentium 4 processor with 2.4 GHz
    - 256 MD RAM memory
    - 1 GB Hard Disk space
- Network Interface Card (NIC)
- Network Cable

CHAPTER 4

RESULT AND DISCUSSION

4.1 Findings

This is the critical and the most important part of this project. All research and product are presented here, in this section. Both research and product must achieve its objectives as to announce as a successful project.

There were several problems faced during the development of this project. The problems might be trouble to deploy programs on multiple machines. This is because the distributed nature of their deployment means that Jini applications cannot rely on many of the assumptions which could take for granted when developing Java programs within a single machine environment. In fact, developing Java applications in the same way probably develop "standalone" Java applications by doing everything on one machine with the same CLASSPATH.

During the deployment stage, because of convenience or necessity which it may develop and test the code on only one machine. In such cases, it is easy to run into problems because by running everything on one machine. It is not exercising parts of the code that may cause things to break when the components of distributed system run on different hosts.

It is the great danger when testing only 'locally". The apparent early convenience of taking the easy way out can result in greater headaches down the road. It is a good idea to get in the habit of thinking about multi machine environments from outset. The best way to avoid future deployment problems is to mimic the multi machine setting which will eventually be expected to run.

Many of the problems encounter in developing Jini programs arise from the integration of code and the need to download software across the network. Often errors occur because the code that has been anticipated is not being downloaded correctly. If fortunate, then no code is downloaded and the error will immediately visible. However, it is more than possible that it still can be downloaded from the wrong location and the errors will not be as immediately visible. These situations made the development process delayed and to avoid introducing subtle and problematic errors, attention to configuring the services that support the downloading code is demanded. The setup and configuration was not as easy as it see when there were many things to consider and understand so that all parts are working especially those servers. It is impossible to configure without any reference because if Jini was configured wrongly, no services available in the end of the day.

In addition, during the configuration and setup the Jini 1.1 environment, the main problem occurred which was the reggie, the Jini build in function to lookup for services was not functioning although those three servers which were RMI activation daemon, Jini HTTP server and Jini lookup service. These three servers were interrelated so that the function of lookup for service can be applied accordingly. In this case, the Jini HTTP server was not functioning accordingly as it could not supply the downloadable code for Jini lookup service. So the result was no service available although the end result supposed to be the other way round.

After few testing and investigation, the problem mentioned above was because of the version of Java Development Kit (JDK) used. Jini 1.1 was only compatible with JDK 1.3 which could not be downloaded in Java websites as this JDK already in the stage of end of life (EOL). The only JDK available to download was the latest version JDK 5.1. This new JDK which been installed was not compatible with Jini1.1 and as the result, problem mentioned before produced. The reason why Jini 1.1 selected for the development tools was because the tutorials that managed to get and available were focusing on using this version only and no other versions tutorial available.

Therefore the next step was to change Jini version as to find the one which support or compatible with JDK 1.5. Jini 2.1 Beta 2 was the version selected and this new version has additional functions as it was the enhancement product from Jini 1.1. The new environment needed to be setup and configured and the author managed to encounter the problem mentioned above which the server functioning accordingly. Next problem occurred was to locate the reference or the address of services in order for user to communicate directly with services. Service registry do not able to provide the proxy to user so as a result, user could not able to execute or use the service. This was the problem which the developer unable to settle by schedule. The services can be view by service browser but the proxy not provided as it suppose to.

4.2 Result and Discussion

The final product for this project will be a product that enables to show the functionality of how service discovery is actually works by using the Jini Network Technology by Sun Microsystems. The ability to search for services can be proved when user able to contact the Jini community which implements a particular Java interfaces and download code which implements that interfaces. The key factor for the product was by implementing two architectures which were service-oriented architecture and Jini architecture.

Based on the final product developed, some of the objectives were achieved and this was due to the problems occurred and the developer level of expertise. The product able to show the ability to search for a service by using the service-oriented architecture is to show that it is already achieve the project objectives and already prove it. Therefore the product do achieved its objectives which mainly focus on the ability to search for a service in a different way from usual. The services introduced in the network are actually to be discovered, configured, and used with a minimum manual intervention.

The end result for this product was the product was able to show those services available to user but the user does not have the ability to execute the service they desired. In other words, the user able to search and discover the available services but they could not use the service. Here are the steps in searching for service which firstly, user will search a specific service from the lookup service. Then, the lookup service will do an 'advertisement' and announce about the service that users looking for and the lookup service itself will tells the users the specific proxy to download the service. Previously, the service itself must register with the lookup service so that the service will visible in the lookup service perspectives. Next step was users will use the proxy given by lookup service to download the service which in other words, users can communicate with the particular service after the users have been given the proxy for particular service by lookup service.

Based on the steps stated before, the part which not functioning was the one user will use the proxy given by lookup service to download the service which in other words, users can communicate with the particular service after the users have been given the proxy for particular service by lookup service. The problem faced here was about the user could only view the services available by viewing them at service browser but do not able to communicate and execute the services user desired. User could not get the proxy to communicate with the services. The user only could search and view the services listed by service registry but could not execute the services desired.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Relevancy to the Objectives

As stated in chapter 1, the main objectives which is the ability to find a service provider if there is a service in the network with the properties described in the request is achieve and can be run by the product to prove that the objectives are implemented. The idea of service discovery is actually different with the usual type of network which available outside. Achieving the objectives acts as prove that service discovery is actually can be implemented in network and it brings together with its advantages which to show that service discovery itself is special type of networking concept.

Service discovery which implemented in this project leads to the minimum of manual intervention in network. The significance of lookup service is where its role is to identify the services which are available in the network and provides the proxies to user in order the user to communicate with the services that they would like to discover and use. This is actually minimizing the manual intervention compare to the usual concept of network. This is suitable for low level of user where the complexity of installation, configuration and management of peripherals. With service discovery technologies, clients and services are designed to support dynamic integration, of which the peripherals installation problem mention above is a specific case.

5.2 Recommendation

Currently in this project, service discovery is implemented within the concept of client /server architecture. In other words, the service-oriented and Jini architectures are subset of client/server architecture. As a recommendation, service discovery can be implemented in mobile or wireless concept. This is actually level up the implementation of service discovery in the network world.

Diversify the usage of service discovery into mobile concept will made service discovery more usable in the real world where user can search for services at anywhere they go. This specifically can be used by the auditors and other travelers which might needed the services during their travel. This will help the user to discover any services available which they can use specifically in their work.

In addition, when service discovery is implemented in mobile, Jini should make the tool in object-oriented so that the software or tools being used is attractive, interactive and effective. This is the ideal for mobile user in the future.

REFERENCES

1. Adrian Friday, Nigel Davies, Elaine Catterall, Supporting Service Discovery, Query and Interaction in Ubiquitous Computing Environments, Second ACM international workshop on Data engineering for wireless and mobile access, 2001.

2. J. Allard, V Chinta, L. Glatt, S. Gundala, and G. G. Richard, III, "Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability,"submitted for publication.

3. Salutation Service Discovery Architecture. www.salutation.org.

4. Javasoft RMI documentation, www.javasoft.com/rmi.

5. Robert Flenner, Jini and Javaspaces Application Development, Sams, 2002.

6. Golden G. Richard III, Ph.D, Service and Device Discovery: Protocols and Programming, McGraw-Hill, 2002.

7. Olufisayo Omojokun, Prasun Devan, *Directions in Ubiquitous Computing: A Position Statement*, The Second IEEE Workshop on Internet Applications, 2001.

8. Adrian Friday, Nigel Davies, Elaine Catterall, *Supporting Service Discovery, Query and Interaction in Ubiquitous Computing Environments*, Second ACM international workshop on Data engineering for wireless and mobile access, 2001.

9. W. Keith Edwards , Tom Rodden, *Jini Example by Example,* Prentice Hall PTR, 2001.

10. *Superstring: A Scalable Service Discovery Protocol for the Wide Area Pervasive Environment*, Sydney, September 2003. Proc. of the 11th IEEE International Conference on Networks.

11. Sun Microsystems. Jini architecture specification version 2.0, June 2003.

12. Sun Microsystems. Jini technology surrogate architecture specification, October 2003.

APPENDICES

Appendix - Comparative table on service discovery protocols

	Type of network	Architecture	Storage of service information	Search methods	Event notification	Service description	Service selection and usage	Fault tolerance and mobility support	Network scalability	Security
Jini [19]	Enterprise network	Centralized	On Lookup Service	Both active and passive discovery for finding the Lookup Service	Distributed events	Java proxy objects	Usage - Java RMI via proxy objects	Lease mechanism for granting access to services	-Service grouping -The use of federation	- Authentication - Authorization (access control lists) - Confidentiality -Integrity
UPnP [11]	Enterprise network	P2P	On every control point	Both active and passive discovery for finding services	Eventing mechanism	XML description, based on UPnP template language	Usage - messages encapsulated in SOAP	-Expiry time for advertisements -"Device unavailable" notification	-	-
SLP [13]	Enterprise network	1. Centralized (with DA) 2. P2P (without DA) *DA = Directory Agent	1. On DA 2. On UA and SA *UA = User Agent SA = Service Agent	1. both active and passive discovery 2. active discovery of services	-	Service templates registered with IANA	-	- Lifetime for service registrations	-More DA -Scope mechanism for service grouping	Optional authentication of DA and SA (using digital signatures)
Bluetooth SDP [9]	Small, maximum 8 devices that can be low-cost and low-power	Client-Server	On every server	Request / Response messages between client and server	-	Service attributes (ID and value)	-	Implicit (no caches are maintained)	-	Determined in the Bluetooth connection negotiation phase
Salutation [10]	Any network	Flexible (can be P2P or centralized)	Service Registry on every Salutation Manager	Salutation Manager Protocol between two SLMs	Long-term requests	Service description records	Usage - through RPC	Periodic availability check of services	-	User authentication (username and password)
INS/Twine [5]	Large and dynamic environments	Overlay network of resolvers which form a DHT	Each resolver holds a range of keys and their values	Service discovery messages are routed in O(log N) hops	-	Hierarchies of attribute-value pairs	-	-Each strand is stored on multiple nodes -Hybrid state management scheme	Hash-based partitioning of resource descriptions among resolvers	-

Continued on next page

	Type of network	Architecture	Storage of service information	Search methods	Event notification	Service description	Service selection and usage	Fault tolerance and mobility support	Network scalability	Security
SSDS [15]	Wide-area	Hierarchical structure of SDS servers	SDS servers	- passive discovery of SDS servers - client queries are routed through the hierarchy	-	XML; aggregation of service descriptions using Bloom-filtered crossed terminals (BCT)	-	Soft state of service announcements	-Multiple shared hierarchies -Shedding server load by spawning on a nearby machine	Authentication of endpoints via digital signatures, privacy via encryption and access rights via capabilities
CSP [1]	Wide-area	Three-tier: proximity, domain and global; hierarchical structure of servers	Servers	Exploring the hierarchy of servers	-	Attribute-value; aggregation of service descriptions using Centroid method	Selection – through context attribute	Frequent updates for intra-domain movements; no updates in the global network	-Hierarchical structure -Minimal overhead associated with unpopular services	-
INS [3]	Dynamic and mobile	Spanning-tree overlay network	Overlay network formed by INRs	-Domain Space Resolver for INR discovery -Passive discovery for services -Early and late binding	-	Name-specifiers, consisting of attributes and values	-	-Replication of service information among resolvers -Soft-state name exchanges and updates	Spawning INRs to candidates resolvers	-
Superstring [2]	Wide-area	DHT overlay network	Resolution hierarchy that matches the service description	Queries are resolved through the routing process	-	Hierarchical description model	-	-	Distribution of the storage and queries among several nodes	SuperstringRep: trust problem addressed by means of a reputation system
JXTA [28]	Wide-area P2P	Virtual network overlay	Rendezvous peers	-Loosely-consistent DHT combined with a limited-range rendezvous walker (used for out-of-sync indices)	-	XML	-	-Index replication among resolvers -Periodic random updates and heartbeat among rendezvous	-Scalable distributed hash indexing - no frequent updates as DHT is loosely consistent	-Certificate authority-based trust models –Credential mechanism for message validation -Encryption -Client and server authentication via digital signatures

Continued on next page

	Type of network	Architecture	Storage of service information	Search methods	Event notification	Service description	Service selection and usage	Fault tolerance and mobility support	Network scalability	Security
CDS [12]	Hundreds or thousands of mobile devices	Hash-based overlay network	Small set of rendezvous points for each content name	Applying the hash function to AV pairs in the query	-	Attribute-value pairs	Selection - query optimization algorithm: node with the smallest response time or the node with smallest database is selected	Names are stored in a soft state fashion	Load Balancing Matrix to share registration and query load	-
GloServ [4]	Local-area and wide-area	Hierarchical	Name Servers manage information of services	UA gets the service hierarchy and queries the name server using RQL	Event subscription through event agent	RDF schemas and descriptions	-	Registration is periodically renewed	Hierarchical structure	Authentication of service providers
One Ring to Rule Them All [6]	Large-scale P2P	Universal ring	Persistent store through a DHT overlay network	-Discovery of contact node at bootstrap -Distributed search engine for finding services	Persistent queries for notification of new services	Textual description	Usage - service implementations stored on the overlay network	File replication over multiple nodes	-Scalable distributed hash indexing -Caching used to avoid overloading nodes	- Authentication with digital signatures
Splendor [34]	Mobile ad-hoc networks – public environments	Four components: clients, services, directories and proxies	Directories	-Active and passive discovery of directories -Clients query directories for services	-	-	-	-Soft state storage of service information -Hard-state storage of services represented by proxies	Aggregation and filtering of service registrations	- authentication - authorization - confidentiality -integrity -non-repudiation
Service Rings [17]	Mobile ad-hoc networks	Overlay structure of hierarchical rings that groups services which are geographically and semantically close	Service Access Points present on every ring	Queries are routed through the ring hierarchy	-	-Languages that support "dist" and "sum" functions - Descriptions are summarised at SAP points		-RingCheck message periodically checking for consistency -Algorithms for broken rings and network partition and reintegration	-Scaling through the hierarchical structure -Algorithms for ring restructuring	-

Continued on next page

	Type of network	Architecture	Storage of service information	Search methods	Event notification	Service description	Service selection and usage	Fault tolerance and mobility support	Network scalability	Security
LANES [16]	Mobile ad-hoc networks	Loosely-coupled lanes overlay structure	Nodes within a lane share the same service information	Discovery messages are transmitted from lane to lane using anycast routing	-	-	-	-Proactive maintenance -Algorithms for node login/logoff, broken connections, network partition and reintegration	Algorithms for optimizing the lanes structure	-
Kozat and Tassiulas [18]	Mobile ad-hoc networks	A subset of the network nodes form a dominating set (virtual backbone)	Distributed directory located on the virtual backbone	Source based multicast tree algorithm	-	-	-	-Periodic service registration -Algorithms for backbone maintenance	Through the use of multiple service broker nodes	-
FRODO [27]	Home appliances -3C(poor), 3D(medium) and 300D(powerful) devices	Centralized	Central acts as a repository for service information	Clients ask the central for services	Subscribers are notified about any change in the state of a service	-	Selection - Devices can request for the best matched service Usage - through XML parsing	-Central election and re-election -Recovering from Central/Backup failure -Soft state for 300D devices, periodic poll of 3D devices	-	-
DEAPspace [21]	Wireless ad-hoc single-hop	P2P	Entire world view on every device	Modified passive search	-	Includes a TTL and the address	-	Nodes broadcast their entire view - rapid convergence	-	-
Konark [14]	Wireless ad-hoc multi-hop	P2P	Service registry on every device	Both active and passive discovery	Clients may subscribe to events	XML	Usage – SOAP	Servers periodically announce their services	-	-
Allia [23]	Ad-hoc	P2P, nodes are organized in alliances	Every node caches advertisements from nodes in its alliance	-Passive discovery -Active discovery by multicast or broadcast service requests	-	-	-	Adjustment of advertisement rates and alliance diameter based on mobility of nodes	-	Policies for access rights and credential verification

Continued on next page

	Type of network	Architecture	Storage of service information	Search methods	Event notification	Service description	Service selection and usage	Fault tolerance and mobility support	Network scalability	Security
GSD [7]	Mobile ad-hoc	P2P	Every node caches advertisements to maximum N hops away (advertisement diameter)	-Passive discovery -Active discovery by selectively forwarding the request to a set of nodes based on semantic information	-	- DAML+OIL -facilitates grouping of services based on service functionality	-	Adjustment of the advertisement time interval and advertisement diameter based on mobility of nodes	-	-
Wu and Zitterbart [32]	Mobile ad-hoc	P2P; based on DSR routing protocol	Every node caches advertisements	Both active and passive discovery	-	-	-	Reactive: -implicit service confirmation -reception of a service poll reply -reception of a "no service" indication	-	-
Varshavsky et al. [29]	Mobile ad-hoc	P2P; based on DSR and DSDV routing protocols	Each nodes maintains a service table	Both active and passive discovery	-	-	Selection - Clients choose the service with the lowest hop count	-Reselection: none, route breaks, any change - Rediscovery: proactive, reactive (route breaks, no route to server, no route to any server)	-	-
Cheng and Marsic [8]	Mobile ad-hoc appliances	P2P; based on ODMRP routing protocol	On every node which is interested in the service	-Active discovery -Passive discovery in the bootstrap	Nodes subscribe for receiving service updates	-	Usage - service invocation through mobile objects	Implicit through active discovery	-	-

Appendix B

Interfaces

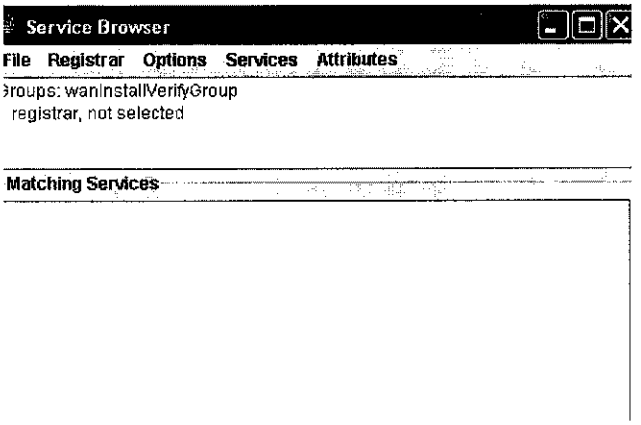


Figure a: Service Browser

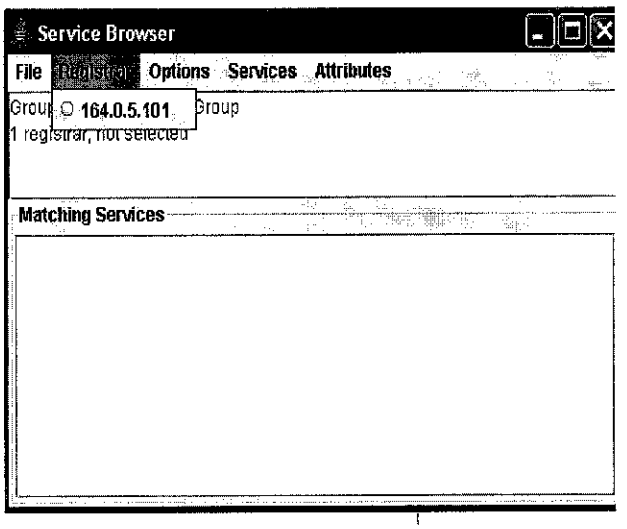


Figure b: Service Browser Host

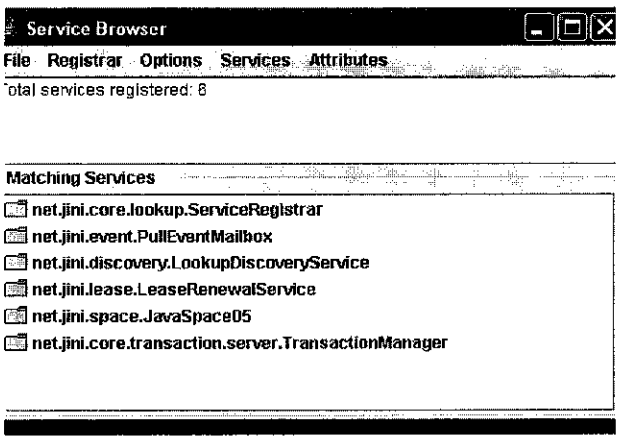


Figure c: Service Browser Main Services

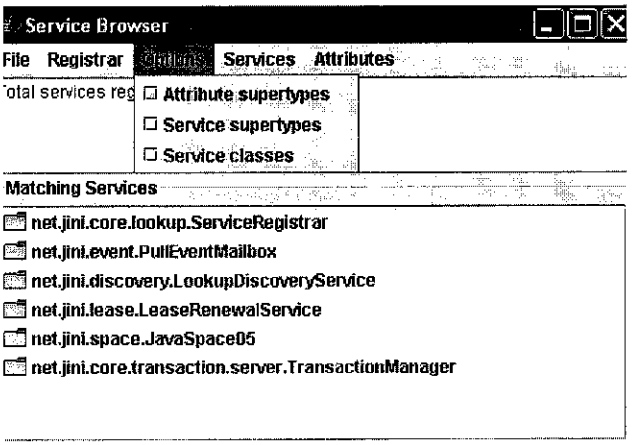


Figure d: Service Browser Option

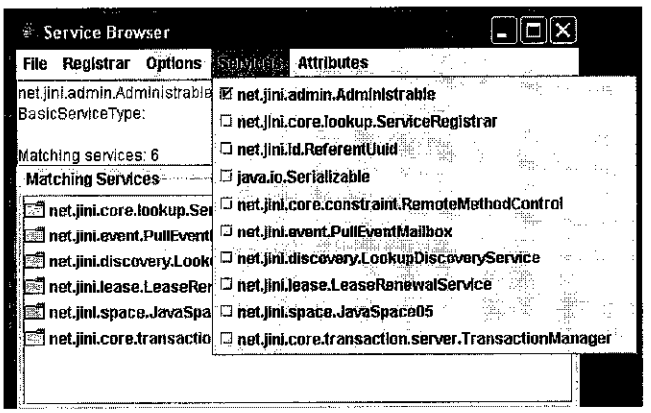


Figure e: Service Browser Service List

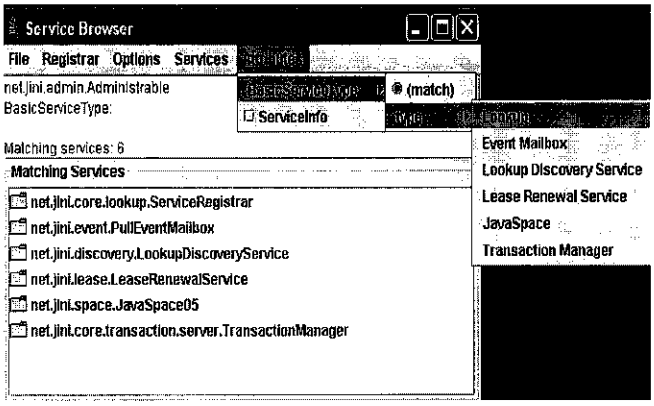


Figure f: Service Browser Attributes

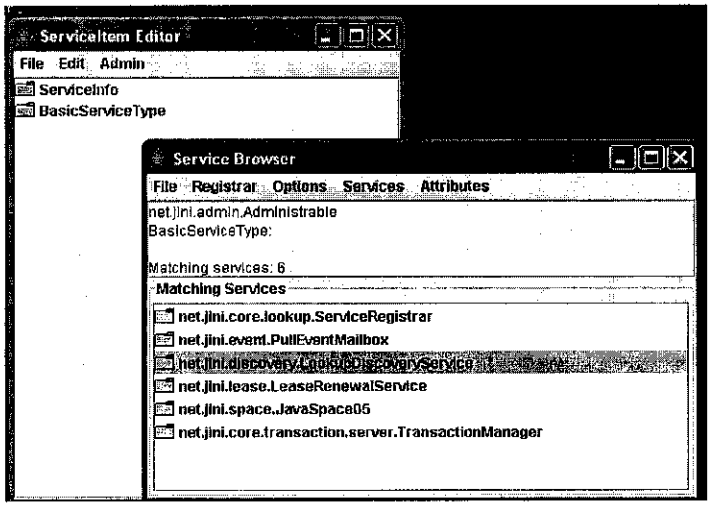


Figure g: Service Browser & Service Item Editor