SERVICE DISCOVERY IN MOBILE ENVIRONMENTS

Nandini Ravi
Department of Computer Science and Engineering
University of Texas, Arlington
nravi@cse.uta.edu

Abstract

Mobile computing and mobile devices such as handhelds, laptops and even wearable computers, are changing the way we live and work. With numerous services available on the Internet, the problem of configuring the system for each service becomes a problem and is further compounded for the mobile user who moves into foreign networks that have unknown infrastructures. Service discovery aims to simplify the process of finding and also using services. Service discovery protocols enable automatic discovery of resources, device cooperation and the reduction of configuration hassles. This paper presents an in-depth analysis of two major service discovery protocols-Service Location Protocol and Jini while surveying the other protocols such as Salutation, UPnP and Bluetooth.

Key words: Service Discovery, Jini, SLP, Lookup services, Mobile Computing

1.INTRODUCTION

The emergence of the Internet has led computing into a new era. In the last few years we have seen a paradigm shift from a world dominated by standalone computers to one where the network is the norm. As Bill Joy of Sun Microsystems puts it, the Network is the Computer [1]. The world of computing has come a long way from the time printers had to be physically attached to the computer to be accessible and floppy disks were required to move data to the computer connected to the printer. With the advent of networks, shared devices began to move out of the computer room and across the network; sharing of hardware devices such as storage media and printers began proliferating. But today with an explosion of services available over the network it is becoming increasingly difficult for the average user to keep up with the configuration requirements for each such service.

"*Services*" do not denote software services alone, but any entity that can be used by a person, a program or another service [2]. This could be software, information access via the Internet, music on demand, classical services such as those offered by printers, scanners and fax machines or hardware. It would be simpler for the user if she/he did not have to manually search for the desired service over the network. This problem is especially significant in mobile computing scenarios. Resource constrained mobile devices such as PDAs and laptops depend on the network for several services such as storage, printing and computations. Mobility into a foreign network implies that the user still has to locate the services on the network before he/she can use them. For example, a user with a PDA and in a foreign network would probably experience low bandwidth because the device is not aware of a web-proxy nearby. Service discovery is a new area of research that focuses not just on offering plug and play solutions but aims to simplify the use of mobile devices in a network allowing them to "discover" services and also be discovered; enabling mobile devices to be automatically configured to use the desired services. Service discovery is taking on the role of critical middleware in mobile computing environments especially with the advent of location-based services and peer-to-peer computing. In the case of pervasive computing where numerous devices would have to act interact with each other, discovering services from a peer device with the help of service advertisements, helps achieve the desired level of intelligence [3].

SERVICE DISCOVERY PROTOCOLS

The chief protocols employed in SERVICE DISCOVERY are

1.UPnP- *Developed by a consortium of companies lead by Microsoft*
2. **Salutation-** f*rom the Salutation consortium*
3. Bluetooth-*Developed by the Bluetooth Special Interest Group*
4. Service Location Protocol (SLP)- *From Internet Engineering Task Force (IETF)*
5.JINI- *from Sun Microsystems*

With the exception of Bluetooth, which offers only service discovery and not delivery, the various Service discovery protocols are alike in their goals [3]

1. To search and browse for services
2. Choose the right service
3. Use the service

Although the goals of the protocols are common, they are different in their approaches. This aim of this paper is to discuss how these protocols work with a special emphasis on SLP and Jini. The paper is organized as follows: section 2 discusses the UPnP, Salutation and Bluetooth service discovery approaches, section 3 focuses on SLP while section 4 elaborates on Jini. Section 5 has a critique of Jini and SLP as well as a comparison of their approaches. Section 6 presents the Jini-SLP bridging protocol and section 7 presents a summary comparison of the major service discovery protocols. In section 8 areas for future research are identified and the paper concludes in section 9.

2. SUMMARY OF UPnP, SALUTATION AND BLUETOOTH:

UPnP:
UPnP [5] stands for Universal Plug and Play. It has been developed by a consortium of companies formed in 1999 lead by Microsoft. UPnP is designed to support zero-configuration, "invisible" networking, and automatic discovery for a breadth of device categories. It can be used with just about any network medium - radio frequency, phone lines, power lines, communications, Ethernet and IEEE 1394 to name a few. This vastly expands the range of devices that UPnP may be used on.

The main components of a UPnP network are **control points and services.** A control point in an UPnP network is a controller capable of discovering and controlling other devices. The steps involved in UPnP networking are ***1.Addressing 2.Discovery 3.Description 4.Control 5.Eventing*** [5].

Every device upon being plugged into the network gets an IP address either from a Dynamic Host Configuration Protocol (DHCP) server or using Auto-IP procedures (***Addressing)*** [5]. When a device or control point is first plugged into a network, it sends out presence announcements using the Simple Service Discovery Protocol (SSDP) (***Discovery)***. Control points(clients) searching for services also use this protocol. Each device hosts an Extensible Markup Language (XML) file called the Device description document that has device information such as device name, service type as well as a pointer (URL) to another XML file containing the service descriptions ***(Description*)**. Upon boot-up control points send out search requests for devices of interest as multicasts. Devices listening on a multicast port examine the search criteria and should they match with their own device and/or service descriptions, send a unicast reply stating the URL from where the Device Description file is available. The control point has access to the services through the service description file, which lists commands and parameters for controlling services. To implement remote procedure calls in order for services to be executed, a Simple Object Access Protocol (SOAP) using XML and Hypertext Transfer protocol (HTTP) is used **(*Control).***

Event Notification is also possible in UPnP by which control points are updated about changes in the service variables that it controls [5] ***(Eventing)***.

Analysis: UPnP is the newest among the set of service discovery protocols and by using open, standard protocols such as TCP/IP, HTTP and XML, interoperability between vendors is ensured besides making it very easily adaptable into existing networked environments. The use of XML allows powerful description of services, devices and eventing. But UPnP is applicable with TCP/IP networks and not UDP. An important problem that UPnP has is that since it does not have a central registry where services may register and which devices may use to query for services, as is present in Jini and SLP. This results in more traffic in the network. Security is another concern in UPnP. Finally the present SSDP specifications limit discovery to a single subnet. Several UPnP products are in the market today, but they have been limited to small office or home network scenarios only- not large enterprise wide networks.

2.1.2:SALUTATION

Salutation [7] has been developed by the Salutation consortium, formed in 1995, and is vendor-independent. The architecture consists of salutation managers (SLM) and Transport managers.

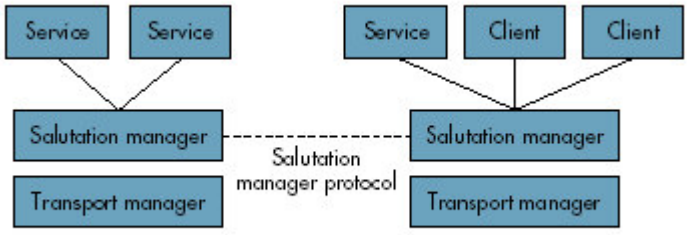


Fig 1: Salutation architecture: Salutation managers are service brokers, isolated by transport managers from the details of specific network transport protocols
Source: [6]

SLMs do the job of service brokering and services are required to register their capabilities with them and clients that need service query the SLMs. SLMs sit on the transport managers which are responsible for providing a reliable communication channel irrespective of the underlying network transports [3]. The transport-independent interface available to server and client applications (SLM-API) includes service registration, discovery and access functions. The interface between the SLM and the transport manager is called SLM-TI. While the SLM may have more than one transport manager, if it is attached to multiple networks, it sees the underlying transport through the SLM-TI and performs the above-mentioned tasks as well as session management. Each SLM also discovers other SLMs and their registered services. So when a client requests a service, all the managers coordinate to perform the search. Salutation lite is a lightweight flavor of Salutation targeted at mobile devices. It lends itself well to low bandwidth network such as IR and Bluetooth [8].

Analysis: Salutation is independent of programming language or network transport and its major advantage is that there already exist commercial applications. There are several salutation products available but most of them are office automation products. However salutation is confined to the service discovery protocol and session management and doesn't handle event notification. The salutation specification also does not handle security issues.

2.1.3:BLUETOOTH

Bluetooth is a new short-range wireless transmission technology released by the Bluetooth Special Interest Group (SIG) in 1999. The Bluetooth stack contains the Service Discovery Protocol (SDP), which

is capable of locating services provided by or available via a Bluetooth enabled device. The SDP is specified in the Bluetooth specification part E [9]. Unlike other service discovery protocols, Bluetooth does not provide access to resources, brokering of services, service advertisements, service registration, or event notification either [4]. SDP aids in the search for services by service class, service attributes and also helps in service browsing. Service browsing is used by the client when there is no prior knowledge about services in the vicinity of the client. Security in Bluetooth devices is ensured through unique 48 bit identifiers, 128 bit authentication keys and 8-128 bit encryption keys.

Analysis: Though the drawback of Bluetooth is that it cannot actually provide a means of using the services discovered, bridging of protocols is underway to use Bluetooth's services by another protocol that can broker and access the service. Salutation has proposed a mapping between its own service discovery protocol and Bluetooth's SDP. Bluetooth would be used at the transport layer while Salutation can take of brokering of services, access of services and session management.

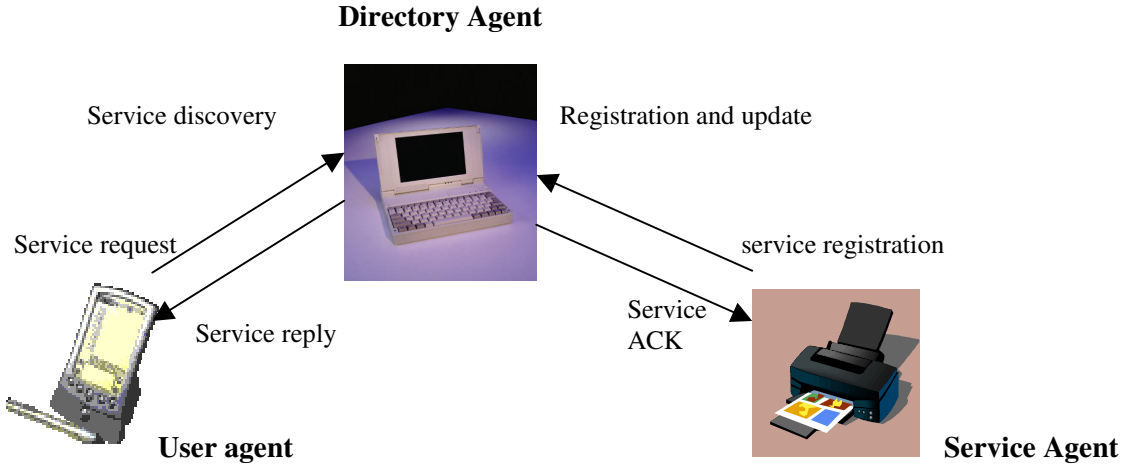
3. SERVICE LOCATION PROTOCOL (SLP):

SLP is being developed by the IETF Svrloc working group and is a popular service discovery protocol. This has been widely accepted in the industry as it is vendor independent and is scalable to large enterprise networks [4]. The first RFC was published in 1997 and what we have today is ver. 2 or SLP v2 [10]. It is primarily designed for TCP/IP networks. One of the motivations for developing SLP was that, all previous service discovery protocols such as AppleTalk and Microsoft Networking only listed all instances of a particular service for e.g.: instances of servers or printers. The Svrloc group wanted to develop a solution that allowed clients to look for a certain service based on its type and obtain the software without having to prompt the user for further inputs. Yet another problem with proprietary protocols was their lack of scalability. In this aspect SLP is scalable from small networks to large enterprise networks [11]. The impact of SLP on the network is minimized by using DHCP and multicast protocols to initialize the service discovery framework without having to configure each SLP agent. Since neither DHCP nor multicast scales the Internet, SLP too does not scale to the Internet.

3.1 SLP ARCHITECTURE:

There are 3 important components:
1.**User Agent** (UA) 2**. Service Agent** (SA) 3. **Directory Agent** (DA).

Fig2: The SLP architecture



Service agents advertise the location and service attributes in the form *service advertisements* on behalf of their services [11]. Directory agents maintain a directory of service type and its attributes and where each service can be found. User agents are those that need service and perform service discovery on behalf of the client software. Their functionality is depicted in the following diagram.

Service advertisements sent out by service agents have information about the service type, service attributes and the service access point. For example, a printer belongs to the service type name "printing" and may have attributes that denote if it is a color/ ordinary printer, laser/ dot matrix printer among other things. The most vital information is the service access point which enables the user agent contact the service agent. Typically this may be the machine's host name or Internet address and perhaps a port number. Other information may also be required in some cases, and these are also made a part of the service advertisement. User agents transmit queries stating the service type desired and other attributes describing the client's requirements to the DA or to the SA if no there has been no response to its queries from a DA [12].

3.2 SERVICE IMPLEMENTATION:

SLP has **two modes of operation**: 1.When the DA is present 2. When the DA is absent.
Upon initialization, the UAs and SAs first determine if there are any DAs in the network.

3.2.1. Methods of locating DAs:

There are four methods by which DAs can be found [11]
1.Using a **configuration file 2**.Using a **DHCP server**. **3.Active discovery 4.Passive discovery**
The first two are considered as **Static Discovery methods**.

> *Configuration file*: The configuration file would contain the address of the DAs and the UA would only have to send a unicast service request to the DA.
> *Using DHCP*: In the second method, UAs and SAs learn about the location of the DAs by using DHCP servers configured to distribute the addresses of DAs to requesting hosts. This address is used to contact the DAs. Both UAs and SAs contact the DAs through unicast.
> *Passive Discovery:* In passive discovery DAs multicast advertisements for their services and continue to do so periodically just in case UAs or SAs missed earlier announcements.
> *Active Discovery:* In the active discovery method, UAs and SAs multicast SLP requests to the network searching for the DA. The SLP multicast Convergence algorithm is used by UAs and SAs to discover DAs and also by UAs when they search for a SA upon finding no DA in the network. This algorithm will be explained in detail in the next section.

Once DAs have been discovered/located SAs respond immediately, registering their service access point addresses, service attributes and service type with the DAs. In the simplest case, if a DA has been located by the UA using one of the means above, the UA directs its queries to it and the DA checks to see if the any SA registered has matching service attributes. If they do match, the DA replies with the corresponding SA's service access point. SLP ensures that then UA do not lose any reply messages even when many of them are sent at nearly the same time. Checking the Time To Live (TTL) field of the multicast request, which effectively reduces the number of network hops from the UA, does this.

In the event that a UA fails to get any response from DAs either because the desired service is unavailable in the DA's scope or there are no DAs at all in the network, its multicast service requests are answered by SAs supporting the service advertisements matching the query [12]. It is to be noted that if a DA reboots, it is presumed to have lost all its service registrations and all the services would have to re-register.

3.2.2. SLP Multicast convergence algorithm:

This algorithm is used whenever the UA attempts active discovery and sends out multicast service requests to discover DAs or, in the absence of any responses from DAs, to discover SAs. This algorithm is very simple yet effective and it helps SLP agents receive replies from more responders than they would have using the standard multicast [11]. In this algorithm, the usual multicast request is sent out and this would yield one or more unicast responses. After a certain wait period following the multicast request, the request is re-issued with an appended list of "previous responders". Each SA/DA receiving this multicast would examine this list, check if it features in it, and if it does it would ignore it. However a new DA/SA that has received this for the first time and would like to send a service reply or a service advertisement (depending on if the reply comes from the DA or the SA respectively), sends a unicast reply. Reliable multicast is difficult to achieve, but this algorithm with its repeated multicasts, ensures a "semi-reliable" multicast transaction [11].

3.2.3. SLP for Larger Network Environments:

It was mentioned earlier that SLP is scalable for larger, enterprise networks. This section shows how.

1. **More DAs:**
Larger networks will need more DAs, as a single DA would be overloaded. Every SA registers with each DA it detects, so all DAs will offer the same service information without requiring any database synchronization. By having duplicate repositories of the same information, the load is equally divided over all the DAs. The other advantages are robustness, as the failure of a DA would not affect the network.

2. **Scoping:**
According to RFC 2165 [10], a scope is a collection of services that make up a logical group. Scoping helps scalability as they limit the number of requests that DAs or SAs need to handle. SLP agents can support multiple scopes. Every SLP agent must belong to a scope and the scope must be specified in each service request. If no scope is specified, "default" is used to specify the scope [12]. UAs grouped into a set of scopes issue queries only to DAs and SAs that support that scope or which support the "default" scope. Likewise, SAs only register with the DAs supporting their scope(s) or the "default" scope. Thus by separating the network into domains, SLP can be suitably scaled for an enterprise network without many problems. The concept of scoping is a separate topic by itself and is beyond the scope of this paper!

3.3 SECURITY ISSUES IN SLP

Security is very important for mobile users moving in a foreign network and for users that form an ad-hoc network [14]. It is important to ensure that only authorized users (perhaps belonging to a scope) access services. It is equally important to ensure that users do not communicate with bogus DAs. One way of ensuring security is by the use of digital signatures. SAs can include a digital signature with all their registration messages and advertisements. This way only DAs and UAs in scopes under which the SA's services are defined can verify them before letting them register (as in the case of the former) or in the latter case, by ensuring that verification is only possible by UAs belonging to the same scope, disallow unauthorized users from accessing services. To ensure the validity of DAs, DAs too can include digital signatures with their advertisements. Some new mechanisms are proposed in [14].

4. JINI

Jini is an extension of the Java programming language and was introduced by Sun Microsystems in 1998.Jini is a distributed system that aims to form a federation or a community of devices and resources**.** As stated in [2], the focus is to make a network a dynamic entity that better reflects the dynamic nature of the workgroup by enabling the flexibly addition or deletion of services. It relies on the existence of reasonable speed connecting the devices on the network, and also the presence of Java Virtual Machine (JVM) on each device that uses or offers services. Latency is also required to be small.

Since Jini uses Java, code and data mobility as well as security in executing downloaded code is ensured. Remote Method Invocation (RMI), the java-enabled extension for remote procedure calls, enables communication between the various services. RMI provides a means for finding, using and garbage collection of objects. This is responsible for Jini's ability to move objects around the network so easily and offer a no-driver-needed service [2].

4.1 COMPONENTS AND SERVICE ARCHITECTURE:

The Jini distributed system architecture is defined by 3 components:
1**.Infrastructure**- The components that help build a Jini community.
2**.Programming Model**-A set of an interface that helps in the construction of services.
3.**Services-**These are the entities in the Jini federation and can be added to or removed from the federation These three components are inter-twined and inter-dependent in making Jini the "system" that it is. To be more specific, the "system" is built with the particular infrastructure and programming models based on the notion of service [2].

The chief entities that participate in Jini are **services, lookup servers (LUS) and clients.** A Lookup server offers **lookup service**, which is the central bootstrapping mechanism for Jini and acts as an intermediary between the clients and services [2]. The lookup server is the Jini equivalent of the DA in SLP and the service provider is the equivalent of the SA in SLP. Services need to register their respective service objects and other service attributes with the lookup service. Service objects are basically Java programming language interfaces along with methods that will be invoked for implementing the service and any other service attributes. Service Objects in a lookup service may include other lookup services and objects that encapsulate naming or directory services. The former aids hierarchical lookup while the latter helps bridge Jini lookup services with other types of lookup services [3].

4.1.1 Component Description [2]

Now that the entities of Jini have been identified, each of the components will now be elaborated upon to lend perspective.

1. Infrastructure:

Infrastructure encompasses

1. **Distributed security systems**, part of the RMI, and which extends the Java security model.
2. The **discover/join protocol**: This allows both hardware and software services to find or "discover" other services in the network, become a part of the Jini system and advertise themselves to other seeking services. Discovery is the first thing that happens when a device is plugged into the Jini system.
3. **Lookup services**

2. Programming Model

The interfaces that comprise the programming model are

1.**Leasing Interfaces-** Services are leased to users and is negotiated between the user and the provider. They can be exclusive or non-exclusive leases, thereby ensuring that access to the

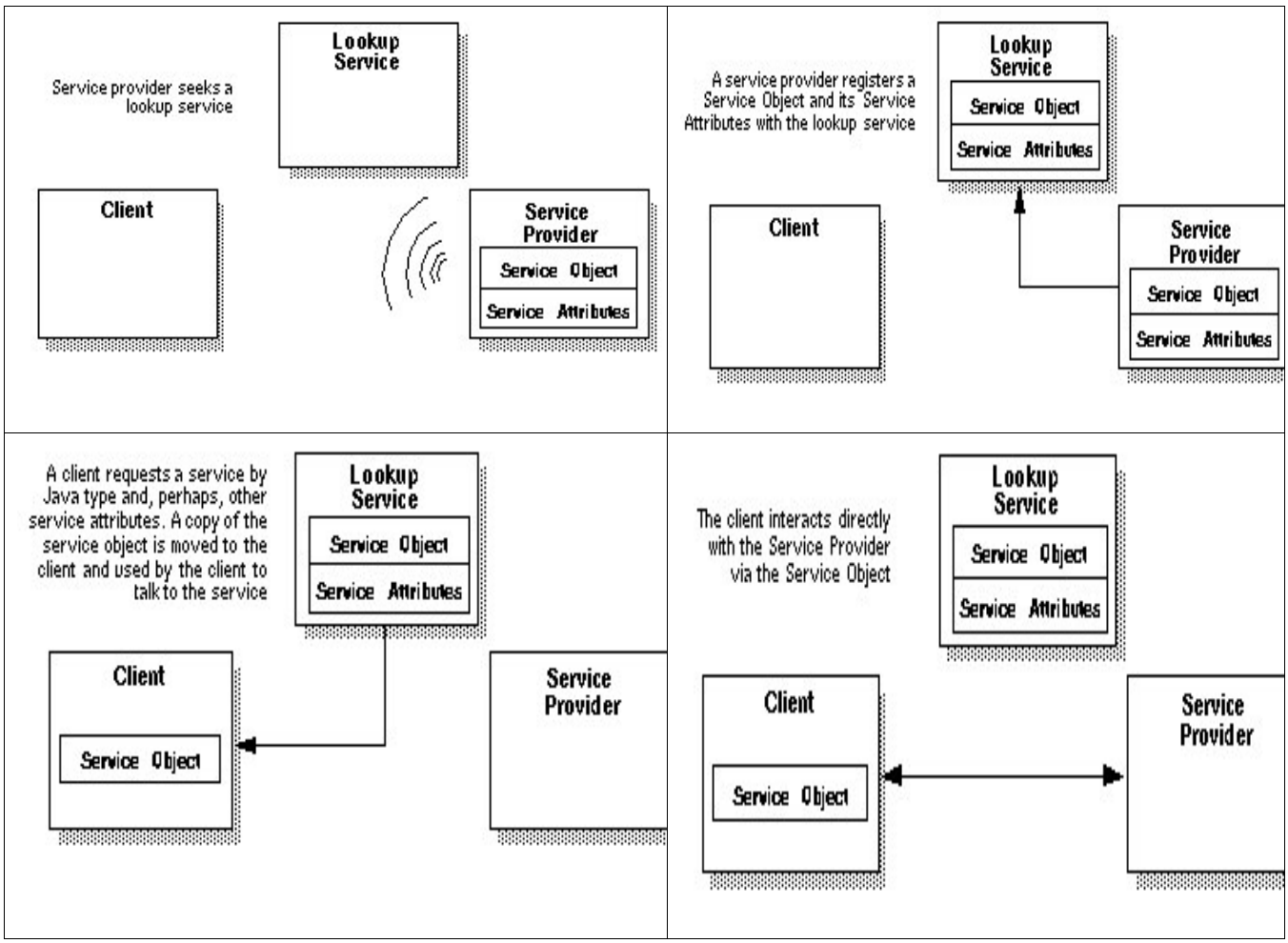
resource requested is either restricted to the requesting user at a time or can be shared with other requesters. Leases need to be renewed for continuing use. If not renewed, the resource is freed. Leases can also be non-renewable.

2.**Event and Notification Interfaces-** This is based on the event-based model of Java and extends that used by Java Beans. Objects can register interest in events on other objects in the network. For e.g.: A device seeking a particular service and upon its unavailability in the network can choose to register interest in being notified once such a service becomes available.

3.**Transaction Interfaces-** A series of operations executed as either part of a single service or spanning multiple services is termed as a *transaction.* Jini employs a transaction protocol to coordinate a two-phase commit. The Jini transaction protocol does not associate itself with how transactions are actually implemented or their semantics, all it provides is the infrastructure for the objects to interact among themselves for a transaction.

3.Services: As stated in [2], "services appear programmatically as objects written in the Java programming language, perhaps made up of other objects". The interfaces of a service define how and what operations may be requested of that service. Some interfaces are to be used by programs and others to be run by the receiver for service-user interaction.

Fig 3:Jini Implementation: 3a:Discovery 3b:Join 3c:Lookup 3d:Service Invocation



Source: [2]

4.2 SERVICE IMPLEMENTATION:

The heart of the Jini system lies in the three protocols- discovery, join and Lookup. Service providers use the discovery process to locate lookup servers to register their services with, as depicted in the diagram (fig 3.a, b). Clients also use this to locate the lookup service when they are plugged into the network. Discovery is done by multicasting a request on the local network for lookup services. Upon locating a lookup service, the service provider would register by transferring a service object to the lookup service (fig 3.b). The client then connects to the lookup service to locate a service by its service attributes and upon finding one, the service object is loaded at the client (fig 3.c). The last step involves the client invoking the service using the interfaces provided with the service object (fig 3.d)

5. ANALYSIS OF SLP AND JINI:

Jini has its advantages from the fact that it offers code mobility, which other protocols don't. Because of this, device drivers can be downloaded easily whereas in SLP only the service's URL is provided. Most mobile devices have very limited memory and power and so cannot be expected to support JVM or RMI. For such devices to be used in Jini, the use of proxies or *surrogates,* which support Java and RMI, have been proposed. The device just sends the data to a surrogate on the local network, which executes it on its behalf. [2] gives a detailed description of how this is done. Since Sun chose speed over storage, it aimed to deliver faster access to services through a lookup service at the expense of using extra storage to maintain efficient data access structures. Also, the lookup server is assumed to be a part of any network a mobile device may want to join, which in reality may not be the case for highly mobile devices. For e.g.: specially two PDAs meeting each other in a foreign network may want to exchange information but cannot due to the absence of a look-up server [14]. However, the good news is that **PsiNaptic Inc** [14] is working to extend the Jini technology and has produced **JMatos**, which frees mobile devices from their dependence of LUS. One can refer [14] for more information on this technology.

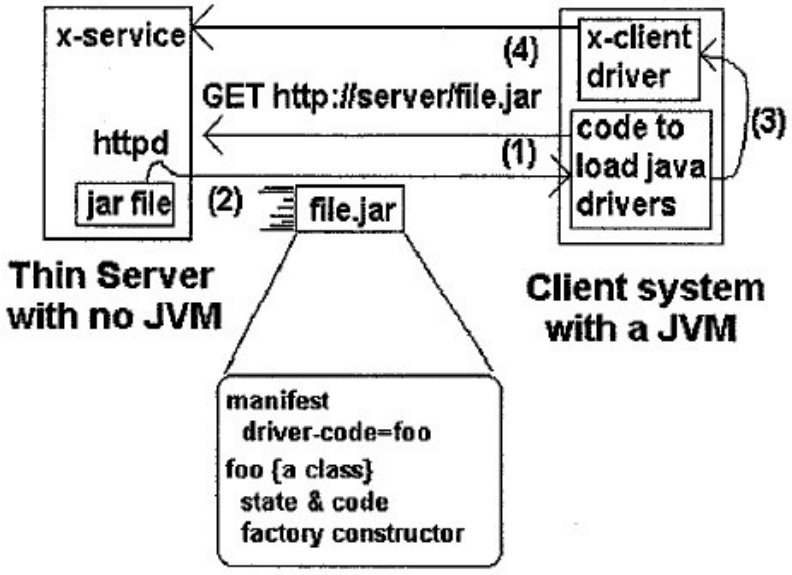
SLP is similar to Jini in that it uses DAs, the Jini-equivalent of lookup servers. However while Lookup servers are required for Jini, DAs are optional. This fact coupled with scoping procedures available for SLP make it flexible for networks of all sizes from home networks to enterprise networks. SLP supports string matching for service attributes using substring comparisons as well as the use of query operators such as AND and OR operators. This gives SLP an advantage over other protocols like Jini or UPnP, which only check using equality. The one problem that SLP faces is its scalability for the Internet. If code mobility is Jini's advantage, the other side of the coin that SLP only offers to locate but not deliver it actually works to its advantage as it is does not incur the overhead of code mobility! Developers perceive SLP as the minimum requirement protocol for service discovery. SLP is considered the best protocol over TCP/IP and a completely UDP based SLP protocol is not common unless there minimal SAs and UAs are implemented. In such cases the service attributes, URLs and authentication blocks are limited in size.

To better evaluate the suitability of Jini and SLP for mobile devices, [15] discusses in detail, experiments conducted to analyze the bandwidth usage of Jini and SLP. It deduces that SLP consumes lesser bandwidth than Jini and that the infrastructure required by Jini is more complex than the one by SLP.This can be attributed to Jini's use of RMI, which causes an overhead. This makes SLP more suited for mobile environments. Nevertheless, Jini is considered a more sophisticated system.

6. BRIDGING OF PROTOCOLS: JINI-SLP BRIDGE

With so many service discovery protocols in the market today, the need for interoperability between devices running different protocols is soon gaining importance. One such is the Jini-SLP bridge [16], which helps the discovery of thin servers. Thin servers are described as "an emerging class of devices that offer services over IP networks without requiring network expertise or infrastructure to deploy". They can

be built into consumer appliances and need to be lightweight to minimize the cost of the base appliance [16]. Some of them cannot be expected to support JVM. [16] proposes methods by which such servers can still offer their services and be accessible in a Jini system. This is made possible through the use of "Jini Driver Factory" object. The SAs that do not support JVMs advertise this capability. According to [16], a special UA, which is the SLP-Jini Bridge, is used whose job is to discover all such SAs. Each such advertisement is turned into registration process of the services, that the thin servers offer, at all appropriate LUSs. The Driver Factory object is a self-contained object in that it contains the code to load drivers upon download and instantiation by the client. This enables a system with no JVM to supply objects to a system with a JVM. The client is then used by the client software to drive the services offered by the thin server. Driver loading can be used by Java clients that employ SLP as well. The same factory code is called to create a factory object and the service object driver.



Source:[16]

Fig 4: (1) The client system requests the Jini Driver Factory jar file from the thin server. (2) The File has a manifest, which includes the name of the Driver Factory class. The client uses this to instantiate the driver object (3). The client is then used by the client software to drive the service offered by the thin server (4). Any additional classes needed to driver factory class has to be present in the jar file.

Analysis: This mechanism is also helpful as it helps download driver software from a thin server. Such a scheme can help SAs based on the SLP/Jini protocols to easily download the drivers that encapsulate the network protocol needed to use the service. The disadvantage of such a system is that the state of the driver factory cannot reflect the current state of the server as the Driver Factory object has been preloaded and stored in the read-only storage of the server.

7. SUMMARY COMPARISON OF THE FOUR MAJOR SERVICE DISCOVERY PROTOCOLS

	SLP	JINI	SALUTATION	UPnP
Developer	IETF www.svrloc.org	SUN MICROSYSTEMS www.sun.com	Salutation Consortium www.salutation.org	Consortium lead by Microsoft www.upnp.org
License	Open Source	Open license, but fee for commercial use	Open Source	Open (Only for Members)
Version	2	1.0	2.1	1.0
Network Transport	TCP/IP (predominantly)	Independent	Independent	TCP/IP
Programming Lang.	Independent	Java	Independent	Independent
OS and Platform	Dependent	Independent	Dependent	Dependent
Code Mobility	No	Yes (Java RMI)	No	No
Service Attribute Searchable	Yes	Yes	Yes	No
Central Cache Repository	Yes	Optional using SLP	Yes (Optional)	No
Operation without directory	Yes	Lookup Table Required	Yes	-
Leasing Concept	Yes	Yes	No	Yes
Security	Authentication Security Feature	Java Based	Authentication	IP Dependent
Main Entities	DA, SA, UA	Lookup Service, Client, Service	Salutation Manager, Transport Manager, Client Server	Control Point, Devices(services)
Service Repository	DA	Lookup Service	A set of SLMs	No
Service Anouncement	Service Registration	Discovery/Join Protocol	Registering with local SLM	Advertisement
Service Discovery	Active, Passive, Static	Query to lookup service	Query to local SLM and cooperation among SLMS	Contact control point/ listen to advertisement
Access to Service	Service protocol for the discovered service	Service Proxy Object based on RMI	Service Session Management	Invoking action to the service (SOAP), query for variable state
Service Registration Lifetime	Lifetime in service registration	Leasing	No	Cache Control header in alive message
Service Group	Scope	Group	No	No
Service Description and Scoping	Service type and attribute matching	Interface Type and attribute matching	Functional units and attributes within it	Description in XML

8. FUTURE RESEARCH:

The present service discovery protocols do not support context awareness. For e.g.: there is no support to service routing and selection based on the client's location. Other contextual information, besides distance to service, that can be used are time, service load and quality of service instances. Some research projects have already begun work towards this end. There is a need for enabling DA-DA cooperation, as in Salutation, for handling service requests in SLP. Wide area SLP (wasrv) is also under research.

9. CONCLUSION:

This paper discussed the motivation behind developing service discovery protocols and explored the methods by which it can be deployed for mobile devices and mobile environments. While Salutation, UPnP and Bluetooth were analyzed in brief, SLP and Jini were focused upon. The two protocols were analyzed on the basis of their individual advantages, shortcomings, similarities and differences. Each of the protocols has its own advantages and disadvantages as they give different weightage to different parameters. There are bound to be more service discovery protocols in the future, hence there will be a greater need for bridging of protocols for interoperability among various devices.

REFERENCES:

[1] K. Arnold et al., ***The Jini Specification***, Addison-Wesley Longman, Reading, Mass., 1999.

[2] "**The Application of Jini Technology to Enhance the Delivery of Mobile Services**", white paper (december 20001) . Available at http://wwws.sun.com/software/jini/whitepapers/index.html

[3] S. Helal, "**Standards for service discovery and delivery IEEE Pervasive Computing**," IEEE Pervasive Computing, vol , 1 no. 3 , 2002 ,pp. 95 - 100

[4] Bettstetter,C., Renner,C., "**A comparison of service discovery protocols and an implementation of the service location protocol**.", Institute of communication networks:Munich,Germany, 2000

[5] **Universal Plug and Play specification v1.0**; available online at http://www.upnp.org/.

[6] G.G.Richard," **Service advertisement and discovery: enabling universal device cooperation**", IEEE Internet Computing , Volume: 4 I Issue: 5 , Sept.-Oct. 2000 Page(s): 18 –26

[7] **Salutation Architecture Specification***;* available online at http://www.salutation.org/specordr.htm.

[8] S.Helal, C.Lee, "**Protocols for service discovery in dynamic and mobile networks**", International Journal of Computer Research, vol 22, no. 1, pp. 1-12.

[9] **Specification of the Bluetooth System***;* available at Http://www.bluetooth.com/developer/specification/specification.asp

[10] E. Guttman, C. Perkins, J. Veizades, and M. Day, "**Service Location Protocol, Version 2**," IETF, RFC 2608, June 1999; available at http://www.rfc-editor.org/rfc/rfc2608.txt.

[11] E. Guttman, "**Service Location Protocol: Automatic Discovery of IP Network Services**," IEEE *Internet Computing*, vol. 3, no. 4, July/Aug. 1999, pp. 71-80.

[12] J.Kempf, P.Pierre, ***Service location protocol for enterprise networks***,Wiley and sons,N.Y,1999

[13] M.Vettorello et al, "**Some notes on security in the service location protocol version 2 (SLP2**)", Proceedings of the Workshop on Ad hoc Communications, held in conjunction with 7th European Conference on Computer Supported Cooperative Work (ECSCW'01), Bonn, Germany, 09/2001

[14] S.Hashman, S.Knudsen, "**The Application of Jini Technology to Enhance the Delivery of Mobile Services**", Dec 2001,white paper, Available at http://wwws.sun.com/software/jini/whitepapers/index.html

[15] C.E Perkins,"**Service Location Protocol for mobile users**", Personal, Indoor and Mobile Radio Communications, 1998. The Ninth IEEE International Symposium on , Volume: 1, 1998 ,Page(s): 141 –146.

[16] E. Guttman and J. Kempf, "**Automatic Discovery of Thin Servers: SLP, Jini and the SLP-Jini bridge**",Proc. 25th Ann.Conf. IEEE Industrial Electronics Soc. (IECON 99), IEEE Press, Piscataway, N.J. 1999.