Resource Discovery in Distributed System[1]

Jyoti Jacob
Computer Science and Engineering Department
The University of Texas at Arlington, Arlington, TX 76019
jacob@cse.uta.edu

ABSTRACT

The problem of network computing is made immeasurably complex by the proliferation of the computing devices and other hardware resources on large networks like the Internet. Configuring these resources supplied by different vendors is a major bottleneck for expansion and building networks on the fly like the Internet, ad hoc and wireless network. Discovery and auto configuration of resources on the network has become more important because of the mobility of these devices and the limitation on amount of resources, which can be installed, on the devices. Wireless network can be beneficial to the mobile users only if the "plug-n-play" feature is provided for using the resources on the network. This can be achieved by developing mechanisms, which can dynamically access and auto configure the resources on the network. Ideally, the discovery mechanism must be flexible enough to be used in large networks and independent of the network protocols. Protocols and standards have already been developed to provide such resource discovery tools. This paper discusses various protocols and tools with emphasis of the usability of such mechanisms in different networks. Interoperability between these tools to provide greater capability and flexibility has also been discussed.

1 Introduction

Resource Discovery protocol has become very important because of the proliferation of devices and the changing networking environment[1]. Until now, static configuration of resources was feasible as the network was small and administration was centralised. But manual configuration becomes very difficult in a large network such as the Internet. As long as the configuration remains a bottleneck, network administration will be difficult, tedious and expensive and users won't be able to take full advantage of the network capabilities. Hence, it is necessary to develop mechanisms for dynamic discovery and configuration of the resources on the network.

Dynamic discovery and auto configuration of the resources is difficult because of the varied specification and data formats of different vendor supplied resources [2]. Success of resource discovery can be achieved by availability of resource information in a universally recognizable format, availability of device- drivers and standard APIs for the ease of use by the end user. Added, the resource discovery mechanism must be designed to work in different network environments like large networks (Internet), ad hoc networks and wireless networks.

Different resource discovery protocols like Service Location Protocol (SLP), Salutation, JINI, Bluetooth and UPnP have tried to achieve dynamic discovery of resources on the network.

[1]This paper is part of the course work of cse6306 – Advanced Operating Systems.

SLP enables network application to automatically find the location and configuration details of services on the network. It allows for decentralised operations and can scale for small to very large networks [3]. SLP has advantages over other resource discovery mechanisms because of its scalability and design. Added, its non-dependence on any other language or mechanism makes it suitable to be used for bridging with other resource discovery mechanisms.

The paper mainly discusses the operation and features of SLP and compares it with the other protocols. Section 2 defines and describes Service Location Protocol in detail. Section 3 gives a brief introduction to the other protocols. Comparisons between these protocols and bridges for interoperability between them have been given in Section 4 and Section 5. The rest of the section discusses the features of these protocol and their improvements.

2 Service Location Protocol

The SLP is a standard developed by the Internet Engineering Task Force (IETF) for resource discovery on the network. SLP provides a flexible and scalable framework for providing hosts with access to information about the existence, location, and configuration of networked services [4]. It enables application on the network to advertise or automatically discover the location of a required service and get the configuration information. Thus, it provides dynamic configuration mechanism for applications on the network. SLP has been intended to service clients in enterprise networks under a cooperative administrative control and shared services. SLP is only used for service discovery. The actual communication between the client and the server after the discovery of the service is outside the scope of SLP.

2.1 Background

SLP is developed by the Service Location Protocol working group, part of the IEFT. SLP Version 1 was the first Proposed Standard Request For Comment (RFC) published in 1997[5]. In June 1999, Version 2 was published and its related documents were promoted to Proposed Standard RFC [4]. SLPv2 provided improvements over the SLP Proposed Standard like reducing the number of protocol features and messages that were mandatory to implement, better security, revise the query language to match Lightweight Directory Access Protocol (LDAP) semantics and allow service:URLs for address families other than IP (AppleTalk and IPX).

2.2 Protocol Overview

The framework of SLP consists of three agents [4]

- User Agent (UA) works on behalf of the user to retrieve service information based on service attributes and type.
- Service Agent (SA) works on behalf of the service to advertise service attributes and configuration.
- Directory Agent (DA) collects information from the service agents to present a single repository of service information

SLP uses Internet Protocol (IP) as the underlying network protocol.

2.2.1 Service discovery

The basic operation consists of the client's attempt to discover the location of a service. The User agent (UA) needs no configuration set-up for locating the service. SLP framework allows for direct communication between the SA and the UA. In small networks or where the UA has no knowledge of a DA, a service request (SrvRqst) is multicast by the UA. The Service Agent (SA) responds directly to a request from the UA, if it matches the requirements of the user with a unicast reply (SrvRply).

In a large network, SAs register their services with one or more Directory Agents (DA). The UA queries the DA to locate a service. Based on the service request, the DA responds with the location address and configuration information of the service. The discovery of DA by a SA or UA has been explained in the next section (see2.3.1)

If the UA already knows the location of a DA, the UA will unicast a request to it to retrieve service information. The DA will unicast a reply to the UA. If a DA is not able to service a request, it should respond with zero values or error code or else the UA will keep on requesting until it gets a request. Interval time is set for the UA to send the request to another DA. If the UA does not receive a response within the given time interval, it contacts another DA. The UA increases the waiting time exponentially between subsequent attempts (doubling the wait period each time). DA addresses may be cached from previous discovery attempts, pre configured, or by use of Dynamic Host Configuration Protocol (DHCP)[4].

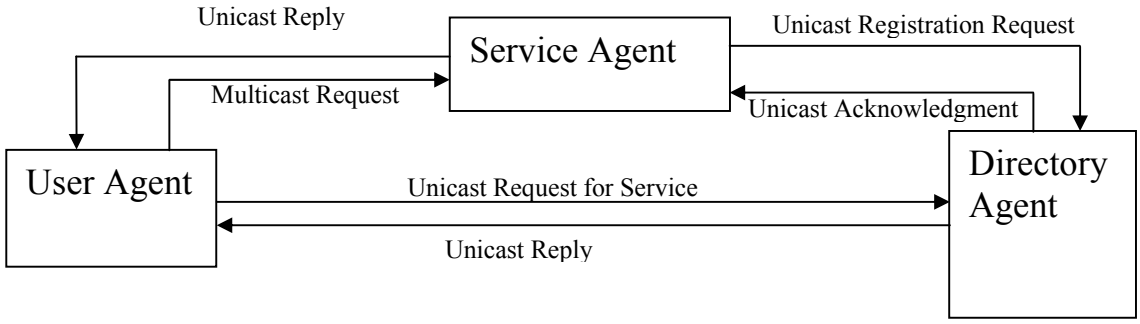

Fig1: Basic Communication between SA, UA and discovered DA

2.2.2 SLP messages

SLP messages are sent using UDP. SrvRqst is used to request for a particular service and the DA/SA respond with an SrvRply. In addition to these two messages, an attribute request and attribute reply is also used to query and obtain information about attributes of a given service. Other messages have been explained in the subsequent sections.

The reserved listening port for SLP is 427. SLP messages can be transmitted on an ephemeral port on which it receives the replies and acknowledgement. If the SLP message is too large to fit in a datagram it

can be truncated. The UA checks for overflow page and if it is set then it establishes a TCP connection with the DA or SA with the same transmission ID.

If the agents receive no response for a request, the messages are retransmitted. A time interval is set which is defined by CONFIG_RETRY wait period. After the initial retry, the interval is increased exponentially. The upper limit on retransmission of unicast request is set by the CONFIG_RETRY_MAX seconds. If no response is received, then the request is sent to another SA or DA. In case of multicast request, a time interval set by CONFIG_MC_MAX parameter. If the UA is satisfied with the first response, there is no need for retransmission. But if it needs to elicit response from more than one DA, a multicast convergence algorithm is used (see 2.3.1.1 below). The message will be retransmitted till there is no more response or if the request is too large to fit into a datagram or the CONFIG_MC_MAX seconds elapses.

2.3 Directory Agent (DA)

DAs are queried by the UA to obtain information about registered services based on type and attributes. SA registers their services with one or more DAs. The DA advertisement message (DAAdvert) message contains the service:URL, the scope of the DA (see 2.5.2 below), attributes and the digital signature (see 2.6 below).

2.3.1 DA discovery

The DA can be located by two different methods: active and passive [3].

Active discovery is a SA and UA initiated method. In this method, UAs and SAs multicast SLP requests to the network. In this case, the multicast convergence algorithm is used (see 2.3.1.1 below)

Passive discovery is a DA initiated method. DAs multicast advertisements for their registered services infrequently. In absence of multicast support, broadcast can be used. Apart from the above-explained method of DA discovery, DA can be pre configured statically or can be configured with DHCP[6].

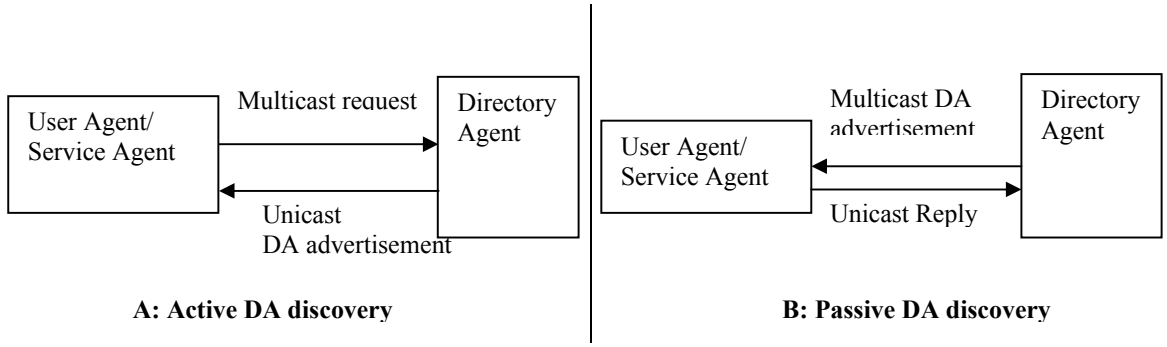

Fig 2: Active and Passive mechanism for DA discovery

2.3.1.1 Multicast Convergence algorithm [3, 4]

This algorithm is used to discover DAs. SA or UA agents, attempting to discover services or DAs, use multicast to send request on the network. DAs respond with unicast messages .One or two replies may be

received in response to the first request. After a wait time interval, the request is retransmitted. The request URL contains a "previous responders list"(PRList). When the request is retransmitted, it contains an appended PRList, including address of each SA or DA that has already responded. When SAs/DAs receive request, they first examine the PRList. They respond to this request only if they are not on the PRList.

For example, the UA1 sent a multicast request on the network containing DA1 and DA2. Both respond but response from DA2 is lost. The UA1 again sends the request. DA1 checks the request but does not respond as it is on the PRList. DA2 does not find itself on the PRList and hence responds to UA1.

The benefits of this algorithm are:

- This algorithm provides a semi-reliable multicast transmission. This is much better than using the basic multicast which is not at all reliable.
- More DAs can be discovered using this method as this algorithm allows more responders to reply than the standard multicast.
- The Requesting SLP agent will not be undaunted with implosion of responses on retransmission.

The only limitation to this algorithm is the length of the PRList. The PRList can be very long and when appended to the service request will result in a large message. Such SLP messages cannot be transmitted in a single datagram and hence cannot be multicast.

2.3.2 Service Registry with DA

SAs register configuration and location information with one or more DAs. SAs send service request (SrvReg) of all their services that they advertise to the DAs. DAs respond with an acknowledgement (SrvAck). A time interval of 18 hrs is set and these registrations have to be renewed within the stipulated time or else they expire.

To deregister a service or some attributes of a service, SA can issue a Service Deregister request.

2.4 Service Advertisement

SAs advertise their services using service:URLs via multicast. The attributes associated with service:URL are defined in document templates registered with the Internet Assigned Numbers Authority (IANA) known as Service Templates. The service:URL provides the client with all the information it needs to contact the service server.

2.4.1 Service:URL

The service:URL allows client/servers and peer-to-peer systems to make use of a standardized dynamic service access point discovery mechanism and is primarily used by SLP in order to distribute service access information. The URL is constructed according to the service schema definition and contains the following: access protocol, an address locating this service, url path information specific to the type of service, and attribute information encoded according to the URL grammar[7].

SLP uses service:URL to register and locate the services. The registration of services includes service:URL and possibly other attributes and digital signature. When a client needs information about a particular service, it sends a request and expects a service:URL specifying the attributes to be returned.

Based on the attributes defined in the set of service:URLs, a client can distinguish the most preferred one among them .The client uses the service:URL to communicate directly to a service

service:URL is defined of the form[4]:

“service:”<servicetype>”://”<addrspec>

- <servicetype> denotes a particular network protocol or abstract service associated with a number of protocols. Abstract service type is an addition in SLPv2 and may not be compatible with SLPv1.
- <addrspec> is the address which is the hostname followed by the port number

For example: 1. “service:http://hostname:80” will indicate a http service on port 80

2. “service:printer” will match both the URLs like “service:printer:lpt1://myhost” or “service:printer:http://hostname” .

Some attributes are not included with the service:URL and can be retrieved using other mechanism. URLs without address specification should not be advertised by the SLP

SLP provides a mechanism for service:URLs to be obtained dynamically, according to the service's attributes. Also service:URLs can be obtained from documents and using other protocols. In this case, the URL may not be accompanied by the service attributes. The context in which the URL appears should make it clear when the service is appropriate to use [7].

e.g.: a document might include a service:URL as a pointer to a service, describing in text what the service does and who is authorized to use it.

2.4.2 Service Templates

Service Templates provide a formal description of the service attributes and the syntax of the service: URL associated with a particular service type.

Service Templates are encoded in a simple form using UTF-8 character set. Though coded in UTF-8 character set, they can be translated into any language and character set.

The following items are included in the Service template[7]:

- <template-type> defines the service type name and an optional additional authority name.
- <template-version> defines the version number of the service type specification.
- <template-description> gives a description of the text provided as part of the template.
- <template-url-syntax> gives the syntax of the service:URL.
- <attribute-definitions> is a collection of zero or more definitions of attributes associated with the service.

Registrations by the SAs are checked against the formal attribute specification defined in the Service template.

2.5 Scalability

2.5.1 Increase DAs

More DAs can be activated in the network to serve more number of clients and servers. SAs register their services with all the DAs, which are discovered by them, resulting in registration of the services in all the DAs. Hence the repositories are sort of replicated without database synchronization. With more DAs in the network, the load will be shared and clients can get quicker response.

2.5.2 Scope

Services are grouped together using 'scopes'. Scopes are strings used for grouping resources by location, proximity in the network topology, or administrative category. DA and SA are always assigned a scope string. A UA may or may not contain a scope string. If no scope is specified then the user can find all services in the entire network. But if a scope is specified then UA will be able to discover services only with that particular scope. UA can either cache service advertisements in a particular scope or can multicast to create the list.

2.6 Security

SLP provides security by authenticating the source of information. Hence it prevents false service information to be propagated.

SAs produce the digital signature using public key cryptography. This signature is included with the registration messages. The DA can verify the signature while registering. These signatures are forwarded in reply messages to the UAs . UAs can reject unsigned or incorrectly signed service information. Additionally, DAs can also include digital signature with their advertisements. Based on these signatures, UAs and SAs can avoid unauthenticated DAs.

In order to verify the digital signatures, additional configurations are needed by the SLP agents. The SLP vendors can deploy new cryptographic algorithms within the SLP's existing protocol. One added feature in SLP is the deployment of new keys gradually without simultaneous reconfiguration of all the systems. Unless the old keys are totally replaced, SAs include digital signatures generated by both old and new keys, in their messages. Thus, the old keys are gradually phased out.

2.7 Additional Features

SLP API allow applications and services to access SLP's functionality. It provides both synchronous and asynchronous operations and features both C and Java bindings.

SLP interoperates with DHCP and LDAP. Interoperability with DHCP helps in configuring the location of the DA or SA by the UA. Interoperability with LDAP allows services with DA to be automatically registered in an LDAP directory.

2.8 SLP Implementations

The following companies and services have adopted SLP [8]:

- Axis, HP, LexMark, Xerox and Minolta support SLP for their network enabled printers.
- IBM, WRQ, Zephyr, Novell all support telnet server discovery and load balancing [9]

- Apple and Opendo or support discovery of Apple application resources.
- Sun Microsystems, Caldera International, Novell, Apple, Salutation Consortium protocols provide platform support .
- Axis and emerging standards work in the IETF IPS WG for Internet SCSI auto configuration support discovery of storage devices.
- Caldera International and emerging standards work in the DMTF (The DMTF is the industry organization that is leading the development, adoption and unification of management standards and initiatives for desktop, enterprise and Internet environments) support management applications.

3 Other Discovery Protocols

3.1 Salutation

Salutation, developed by the Salutation Consortium, provides a framework and APIs for resource discovery. A lightweight version of Salutation called Salutation-lite is also available; primarily to cater to resource limited devices.

Salutation has three important components: salutation managers (SLM), functional units and transport managers. The functional unit defines a service. The specifications of Salutation define attributes that characterize a service. Services are registered with the SLM. Clients query the SLM to obtain information about services. The communication between functional units, clients and salutation manager take place with API calls. API calls are also available for version checking and communication between clients and services.

SLM act as service brokers discovers the services on the network and manage sessions[2]. SLM can operate in native, emulated and salutation personality[1]. In native personality, SLM are used only for discovery. In emulated personality, SLM do the discovery process by establishing the connection and they transfer native data packets encapsulated in Salutation manager protocol format, thus providing a bridge when no common message protocol exists between the client and the server. In salutation personality, SLM not only establish the connection but they also mandate the specific format of the data transferred.

SLM may be hosted on the same device or remotely located. SLMs coordinate and exchange service registration with each other using Sun's ONC RPC [10]. Clients can only query their local salutation manager. SLM contact the other SLMs and direct the service request. Services register with the local SLM or nearest SLM.

Transport independence is achieved by the transport manager. A new transport layer can be written without modifying the SLM. Salutation requires a network transport layer, which supports reliable, stream-oriented communication. Transport managers locate SLM on the network either by multicast, static configuration or reference to a centralized directory.

Salutation currently does not address security issues [1].

3.2 JINI

Jini is developed using Java and relies on the concept of mobile code. Jini consists of three components: services, lookup servers and clients[11]. Client can also be a service. Services register their services with a lookup server and clients query the lookup server to discover a specific service. Service or client use multicast request protocol to locate a lookup server. The lookup server responds with a unicast reply. Once the server is located, the communication takes place with unicast discovery protocol. Like SLP, in Jini also, the discovery process can be initiated either by the lookup server or the client/service. The lookup server uses a multicast announcement protocol to announce itself.

Jini uses remote method invocation (RMI) to interact between the client or service or lookup server. The service can be implemented entirely in the software or can be downloaded by the client, after discovery and implemented on the client.

Jini associates a proxy or remote control object with each service instance. A service registers its object in one or more lookup server. A unique identifier identifies each service. This object may either implement the service entirely or provide methods for accessing the service over the network. Lease duration defines the duration of maintenance of the registration by the lookup service. At the end of this duration, the server should renew the lease. By querying the lookup server, a client can discover a service. Clients must first secure an instance of the proxy object. The object contains the location of the service and the protocol necessary to operate it.

The clients or services need to either run on the Java virtual machine (JVM) or associate itself with a device that can execute a Java virtual machine. Jini depends on Java's security model like digital certificates, encryption and control over mobile code activities for security.

3.3 Bluetooth

Bluetooth, developed by the Bluetooth Special Interest Group (SIG), is a low-power, short range wireless radio system. SIG was formed in 1998 by Erricsson, IBM, Intel, Nokia and Toshiba[12].

Bluetooth employs frequency hopping system and operates in the 2.4GHz band. It has a range of 10 meters and can provide upto 1megabit/sec links to other Bluetooth devices. It supports both synchronous and asynchronous services Hence it can automatically discover wireless network connections and can perform automatic synchronization of data between several Bluetooth devices[1]. Groups of bluetooth devices form ad hoc networks called piconet to communicate, share services and synchronize data.

Bluetooth has its own Service Discovery Protocol (SDP) in which services can be located by polling devices in the piconet. A universal identifier identifies each service. Bluetooth SDP provides simple APIs for enumerating the devices in range and browsing available services. Stop rules are defined to limit the duration of searches or the number of devices returned. Services are identified either by service classes or by the descriptive attributes. Service class defines broader types of devices categories while attributes define the characteristics of the device.

Once a service is discovered, then communication must be handled by higher protocol or other discovery protocols like SLP or Salutation etc. As Bluetooth employs frequency hopping, snooping is difficult. Added, it provides data security through unique 48-bit identifiers, 128-bit authentication keys

and 8 to 128 bit encryption keys. But the Bluetooth devices must be paired to provide them with matching secret keys that will support authentication.

3.4 UPnP

UPnP, developed by Microsoft Corporation, uses XML to discover services and communicate. Virtual web servers in the device act as entry points for interacting and controlling[13]. UPnP support device addressing, service advertisements and discovery, device control, eventing and presentation through the following protocols:

- Simple service discovery protocol (SSDP): the mechanism for service discovery and advertisement.
- Simple object access protocol (SOAP): helps in device control after discovery by supporting remote procedure call based on XML and HTTP.
- AutoIP: allows devices to dynamically claim IP addresses
- Generic event notification architecture (GENA): an HTTP based subscription event notification service.

When devices are introduced in the network, they send an "alive" message to the control points. Each service contains service type, service name and location. Service advertisements are multicast on the network. The client does a multicast request to discover services on the network. Matching services respond with unicast messages. The response contains URL, which has the description document, describing the service. After retrieving the description document, an UPnP component called the "rehydrator" at the client's control point extract the definition of the device's service control protocol (SCP). SCP allows APIs to be converted to device specific commands. Thus, the application level does not have to bother about details of a particular device. The "rehydrator" sends device specific commands via the device's control URL with the help of SOAP.

4 Comparison

Protocol	**SLP**	**Salutation**	**JINI**	**Bluetooth**	**uPnP**
Controlling Organization	IETF	Salutation Consortium	Sun Microsystems Inc.	SIG	Microsoft Corp.
Source code/ Specification	Open	Open	Company controlled	SIG members	Company controlled
Intended application	Internet	General	General	Wireless Telecommunications	Personal computing, small office
Requirements	None	IP	JVM	Blue tooth environment	IP, HTTP, XML
Network Protocol	IP	Any	Any	Wireless	IP
Service advertisement	Yes	Yes	Yes	No	Yes
Repository of services	Yes	Yes	Yes	No	No
Interoperability	Yes	Yes	Yes	Yes	No
Security	Yes	No	Yes	Yes	No
Any: Reliable stream oriented network protocol, XML: extended Markup language, HTTP: hypertext transport protocol.					

5 Building Bridges

From the comparisons, it is apparent that the discovery mechanisms do not have all the necessary capabilities to provide a flexible resource discovery mechanism to the users. Added, it is difficult for the various service vendors to accommodate the specifications by the different protocols. The solution is to establish bridges between the various protocols to allow for interoperability. Below are description of some bridges built between SLP and other protocols.

The SLP-JINI Bridge provides a mechanism for thin servers without JVM to provide service information and service object for registration to JINI [14]. Servers equipped with SLP SAs advertise the Jini driver factory (jar files). The bridge acting as a SLP UA finds all the advertised services and turns it into a JINI service registration and registers with the Jini lookup server. The services are registered in the lookup server-using driver loading. The java driver factory is preloaded in the thin server's (without JVM) read-only storage. This static jar files will be downloaded on the client system which has a JVM. The driver object is instantiated by the client [4]. Hence a system with no JVM can supply objects to a system with JVM.

An SLP-Salutation Bridge has been proposed and reviewed by the Salutation Consortium. This proposal will enhance the Salutation Architecture to support directory based service discovery by utilizing SLP [15]. The SLM will have a SA and a UA to support SLPv2 [16] The SLM will try to discover a SLP DA through multicast, broadcast or manual configuration. If a DA is discovered, the SLM will use SLP protocol to request, register or un-register functional units. If no DAs are found, the UAs will multicast the service request and search for all functional units registered with SLMv2.1. The discovery process will be made transparent to the users by the Salutation API.

6 Discussion

• Architecture: Resource discovery protocols should be able to provide the so called "plug-n-play" feature in different environments like small LAN networks, ad hoc networks, Internet and wireless networks. These protocols must have flexible architecture such that they can be adopted by services and applications in distributed, pervasive and embedded computing. The most important requirement is the ability of the communication protocol to function atop any communication protocol either wireless or Ethernet link. SLP does not offer network flexibility as it has been designed over IP. Salutation and Jini are better options for network flexibility as they offer sort of network protocol independence.

But SLP offers flexibility in terms of its non-reliance on any language unlike JINI and UPnP. SLP is only used for service discovery. The service information and configuration are communicated in strings via messages. Hence, the SLP architecture is very useful in embedded systems with scarce resources. SLP architecture allows for discovery of more resources because of its modified multicast algorithm. The usage of multicast is not necessary for DA/SA discovery. DHCP can also be used, which allows for centralised administration. SLP allows interoperability with LDAP. Additionally the flexible architecture of SLP allows it to be used by other resource discovery tools like Salutation and Jini.

The Salutation architecture also has greater flexibility. But clients can communicate with only their local SLM and the local SLM do the discovery of remote SLM. This bottleneck affects the scalability of the protocol.

• Scalability and Security: Most of this resource discovery protocols are useful in large network centric environment like the Internet or in wireless networks. As the network becomes large, the number of users and services also increase. Hence, resource protocol should be able to scale to a larger network, catering to huge number of users and services. SLP and JINI are scalable to large environment. SLP has been designed for Internet and hence is scalable to cater to increase in users and services. Increasing the number of DAs and using features like scope, SLP can be made scalable to larger networks. It can be also used for small network without DAs unlike Jini, which needs a mandatory lookup server.

Security is a main issue in large networks because of malicious attacks. SLP offers other security options like digital signatures but there is no way in to prevent access to the advertised data. JINI offers better security options than SLP.

• Bridges: It is difficult to obtain all the features in a single resource discovery protocol. A more universal standardised method of resource discovery can be achieved by bridging these protocols. The SLP-Salutation bridge allows Salutation to be used in wider geographic areas, spanning large networks. The bridge between Bluetooth and Salutation allows Bluetooth to be used in large networks. But interoperability has been achieved between only some protocols. More efforts are needed to establish interoperability between all the protocols.

• Compatibility between service and discovery protocols: The information discovered by the resource discovery protocols can be useful only if the service discovered is compatible with the client support for that service. Sharing of information about different services is only possible, if the client can interpret the information properly. Hence, a universal method of labelling devices /services digitally with a common way of describing their capabilities is a necessity. Different organisations like the Salutation Consortium and IETF standard –RFC 1759 are working to achieve this goal. But co-ordination should be achieved between these two standards to provide a universal service description.

7 Conclusion

Resource discovery mechanisms allows for discovery of various hardware/software resources dynamically. Dynamic discovery of resources and their auto configuration allow more flexibility for users to work in large networks. SLP offers more flexibility in terms of language, usage and operations but its reliance of IP, as the network protocol is a drawback.

But success of these discovery mechanisms will depend upon the cooperation of the service vendors and interoperability between them.

8 References

1. Richard, G.G.III, *Service Advertisement and Discovery: Enabling Universal Device Cooperation*, in *IEEE Internet Computing*. September-October 2000. p. 18-26.
2. Pascoe, R., *Building networks on the fly*, in *IEEE Spectrum*. March 2001.p. 61-65.
3. E.Guttman, *Service Location Protocol: Automatic Discovery of IP Network Services*, in *Internet Computing*. July/August 1999. p. 71-80.
4. E.Guttman, et al., *Service Location Protocol, Version 2*. June 1999, IETF, RFC 2608.

5. J.Veizades, E.Guttman, and C.Perkins, *Service Location Protocol*. June 1997, IETF, RFC 2165.

6. Droms, R., *Dynamic Host Configuration Protocol*. March 1997, IETF, RFC 2131.

7. E.Guttman, C.Perkins, and J. Kempf, *Service Templates and Service:Schemes*. June 1999, IETF, RFC 2609.

8. *Service Location Protocol Project*. available at www.srvloc.org

9. Naugle, J., K. Kasthurirangan, and G. Ledford, *TN3270E Service Location and Session Balancing*. January 2001, IETF , RFC 3049.

10. Kvalvaag, E.L. and D.V. Thanh. *Providing facilities to the visiting mobile user*. in *Service Portability and Virtual Customer Environments, IEEE*. 1 Dec. 2000. San Francisco, CA, USA.

11. Arnold, K., *The JINI Specification*. 1999: Addison-Wesley Longman, Mass.

12. *Specification of the Bluetooth System*. available at http://www.bluetooth.com/developer/specification/specification.asp

13. *Universal Plug and Play specification version 1.0*.available at http://www.upnp.org

14. E.Guttman and J. Kempf. *Automatic discovery of thin servers: SLP, Jini and the SLP-Jini Bridge*. in *I25th Annual Conference of IEEE Industrial Electronics Sociey (IECON 99)*. 1999. New Jersey: IEEE Press.

15. Pierre, P.S. and T. Mori, *Salutation and SLP* available at http://www.salutation.org/techtalk/slp.htm

16. *Expanding Salutation Architecture*, in *Results of the Tenth Salutation Consortium Technical Committee meeting,* available at http://www.salutation.org/techtalk/tc10.htm