SERVICE DISCOVERY IN MOBILE ENVIRONMENTS

Ritesh Mehta
Department of Computer Science and Engineering
The University of Texas at Arlington
mehta@cse.uta.edu

Abstract

Advent of mobile and wireless computing has changed traditional way of computing and communication based on wired networks. Users in mobile environment expect the same computing luxury which they used to enjoy in fixed computing environment. With a plethora of wireless hardware and software services present, the challenge is to determine how best we can utilize them. Service Discovery aims to simplify the use of mobile devices and services in a network by allowing them to be "discovered", "configured" and used by other devices with a minimum of manual effort. Also since there are number of service discovery standards, it will be beneficial if they can interoperate. This paper discusses various service discovery protocols along with the bridges available to make them interoperable.

Keywords: Service Discovery, SLP, Salutation, Bluetooth, UPnP, Jini, SLP-Jini, Salutation-Bluetooth, SLP-Salutation.

1. Introduction

"One billion connected devices" this is a citation made by two Java Developers (Mark Sears and Gregory Lewis). This vision is quickly becoming a reality due to a tremendous growth in the field of mobile computing and increase in deployment of Internet enabled devices in diverse environments. Users in mobile environment expect and demands same kind of computing luxury as in wired network. The problem is that devices in mobile environments are generally peripheral/resource poor. Therefore they need to cooperate with each other in order to complement the missing part of other device. Also there are number of services available on internet which a user can access. It is not possible for the user to keep up with the configuration requirements for each such service. For this purpose several service discovery protocols have been proposed.

Service discovery is comparatively a new area of research. The term "services" not only means software service but any service that can be used by a person, program or another service such as access to hardware or information [16]. Service discovery protocols (SDP) aims in re-shaping the way software and network resources are configured, deployed, and advertised, all in favor of the mobile user. The focus is not only to provide a plug and play solution but to provide an environment in which a device can automatically discover services including their properties and services may advertise their existence in a dynamic way. Among emerging SDP are Service Location Protocol, Salutation, Bluetooth, Jini and Universal Plug And Play. The goal for most of the service is to look for service, in case of many services offering same basic function browse services based on some specific attributes, choose the service and use the service. Even though they have same goal, all SDPs have different origins, underlying technologies, flavors, and audiences. Since they see the problem at different angles and take different approaches to it, they have pros and cons, especially compared to others.

For service discovery to become pervasive either a single service discovery technology must dominate or the most commonly used technologies must be made interoperable. Bridging is one of the way in which interoperability can be achieved.

The aim of the paper is to give a summary of different SDP and explain proposed bridges for them. The paper is organized as follows: Section 2 provides background on various SDP, Section 3 describes bridges between different protocols, Section 4 gives directions for future research and Section 5 concludes with conclusion.

2. Related Work:

This section provides a summary of different protocols used for Service Discovery.

2.1 Service Location Protocol (SLP)

Service Location Protocol is an Internet Engineering Task Force (IETF) standard for decentralized, lightweight, and extensible service discovery. It uses service URLs, which defines the service type and address for a particular service. Based on the service URL, users (or applications) can browse available services in their domain and select and use the one they want.

There are three agents in SLP: the *user*, *service*, and *directory*. The UA is a software entity that sends service discovery requests on a user application's behalf. The SA broadcasts advertisements on behalf of a service (Service Registration "SrvReg" message). As a centralized service information repository, the DA caches advertisements from SAs and processes discovery queries from UAs. An SA advertises itself by registering with a DA. The registration message contains the URL for the advertised service and for the service's lifetime, and a set of descriptive attributes for the service. The SA periodically renews its registration with the DA, which caches the registration and sends an acknowledge message to the SA (Service Acknowledgement "SrvAck" message). A UA sends a service request message to the DA to request the service's location (Service Request "SrvRqst" message). The DA responds with a service reply message that includes the URLs of all services matched against the UA request (Service Reply "SrvRply" message). Now, the UA can access one of the services pointed to by the returned URL. A SA issues a request to de-register a service when it becomes unavailable using "SrvDeReg" message. In SLP, the DA is optional. A DA might not exist in a small network, in which case the UAs service request messages are directly sent to the SAs, but it scales well to larger network. The scalability is supported by various features such as the minimal use of multicast messages, *scope* concept, and multiple DAs[15].

SLP supports service browsing and string-based querying for service attributes, which let UAs select the most appropriate services from among available services in the network. The SLP standard is accessible from the IETF SvrLoc working group Web site (see www.ietf.org).

2.2 Salutation:

Salutation Consortium is developing Salutation standard for service discovery and use, among broad set of devices and in an environment of widespread connectivity and mobility. The architecture provides a standard method for applications, services and devices to describe and to advertise their capabilities to other applications, services and devices, so that they can search each other for specific services and utilize them [1]. The Salutation architecture is composed of two major components: *Salutation Manager* and *Transport Manager*.

Salutation Manager

The Salutation Manager is the core of the architecture; it acts as a service broker. The Salutation Manager provides a transport-independent interface to Server and Client applications.
The main tasks provided by the Salutation Manager (SLM) can be summarized as follows.

1. Service Registry: A service provider registers its capability/service with a SLM. The SLM contains a registry to keep information about services. A client registers or unregisters itself with local salutation manager.
2. Service Discovery: When a client asks its local SLM for a service, local SLM performs search for the service in coordination with others SLMs. For this Salutation Manager discovers other Salutation Managers and services registered there. Remote services are discovered by matching type and set of attributes specified by local SLM. This communication protocol between SLMs is called the Salutation Manager Protocol. This feature of coordination between SLMs is called capability exchange. The client can then use the returned service.

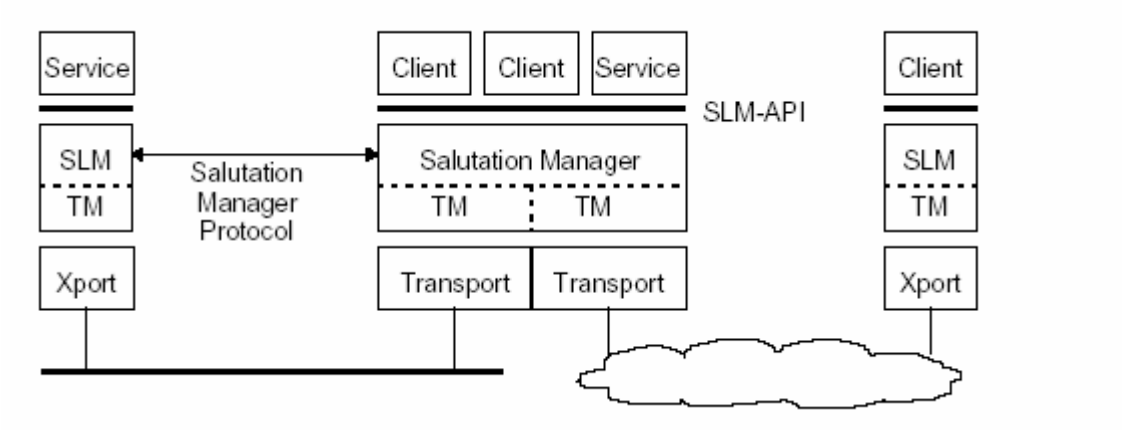

Figure1: Model of Salutation Manager

3. Service Session Management : This session management addresses the service invocation aspect of Salutation. A service session is established when a client wants to use a service discovered through Service Discovery. The service session is operated in one of 3 different modes: native mode, emulated mode, and salutation mode. The SLM may or may not be involved in message exchanges in the service session, depending on the modes. In the *native mode*, messages are exchanged through a native protocol and SLM is never involved in message exchange. In the *emulated mode*, the Salutation Manager Protocol is used to carry messages between client and service but Salutation Manager doesn't inspect the contents. In the salutation *mode*, Salutation Managers not only carry messages, but also define the message formats to be used in the session.
4. Service Availability: A client application can ask the local SLM to periodically check the availability of services. This checking is done between the local manager and the corresponding manager.

A *Functional Unit* is defined as a basic building block in Salutation architecture. In other words, it is the minimal meaningful function to constitute a client or service. A collection of Functional Units defines a service record.

Transport Manager

In the architectural diagram for Salutation a Transport Managers resides below Salutation Manager. Its function is to provide reliable communication channels, independent of the underlying network transports. The communication protocol independence of Salutation architecture is achieved by the interface (SLM- TI) between Salutation Manager and Transport Manager. Transport Manager is an entity, dependent on the network transport it supports. A

Salutation Manager may have more than one Transport Manager, in case it is attached to multiple, physically different networks [17].

2.3 JINI

JINI was introduced by Sun Microsystems in 1998. It is based on Java technology. The aim of Jini architecture is to federate groups of devices and software components into a single, dynamic distributed system. The heart of jini system is a set of three protocols called discovery, join, and lookup. A pair of these protocols discovery/join occurs when a device is plugged in [16]. Discovery occurs when a service is looking for a lookup service with which it can register. Join occurs when a service has located a lookup service and wishes to join it. In order to join, service provider registers a service object containing Java programming interfaces and its service attributes with the lookup service, Lookup occurs when a client or user needs to locate and invoke a service described by its interface type and possibly, other attributes.

Jini connection technology consists of an infrastructure and a programming model which address the fundamental issues of how devices connect with each other to form an impromptu community. Jini technology uses Java Remote Method Invocation protocols to move code around the network.

Other features provided by Jini are leasing and remote events and transactions. Using leasing, access to a particular service is leased to user according to negotiated time between the service user and provider. The lease must be renewed before expiration else the service is revoked.
The remote events and *transactions* feature helps programmers write distributed programs in reliable and scalable fashion. Remote event enables an object to be notified when desired change occurs in the system. These events can be triggered by newly-published services or some state changes of services. Also, Jini supports two-phase commit (2PC) protocol. By nature, Jini is used to build distributed systems where reliability and robustness are likely to get impaired by partial failures and recovery.

2.4 Universal Plug And Play (UPnP)

UPnP stands for Universal Plug and Play. Universal Plug & Play (UPnP) is a distributed, open networking architecture that is designed to enable simple, ad hoc communication among distributed devices and services. It is initiated by Microsoft. In UPnP, a device can dynamically join a network, obtain an IP address, convey its capabilities upon request, and learn about the presence and capabilities of other devices. A device can leave a network smoothly and automatically without leaving any unwanted state behind.

The main components of an UPnP network are control points and services. Control points can be considered as lookup service or directory agent. UPnP uses service discovery protocol (SSDP) for service discovery [2]. This protocol detects presence of other devices and service and announces its own presence to other. The different protocol messages used by devices in order to announce its presence, register its service and perform other tasks are

1. Advertisement - used by devices to advertise its service to control points.
2. Discovery – used by device/control points to announce its presence
3. Description – XML file containing information about the device name, service type, and pointer to other file containing service description.
4. Control - control point can send actions to a device's service using control message. Control messages are expressed in XML using the Simple Object Access Protocol

(SOAP). In response to the control message, the service returns any action-specific values.

5. Eventing -Event Notification is possible in which service publishes its updates by sending event messages to control points and control points updates itself accordingly.
6. Presentation - UPnP provides a higher level description of services in the form of user interface through which user can directly control the service.

2.5 Bluetooth

Bluetooth is a low-power, short-range, wireless radio system being developed by the Bluetooth Special Interest Group, an industry consortium. Bluetooth operates in the 2.4-GHz industrial scientific and medical (ISM) band to maximize international acceptance and employs a frequency- hopping system to minimize interference.

The Bluetooth protocol stack contains the Service Discovery Protocol (SDP) [5]. It is used to configure the stack in order to support end-user applications. SDP can further be used to locate services that are available on devices in the vicinity of the user. Having located available services, a user may then select to utilize any of them.

At the bottom, the radio and baseband layers provide the short-range, frequency-hopping radio platform. The link manager protocol (LMP) handles data link setup and provides authentication and encryption services. The logical link control and adaptation protocol (L2CAP) supports multiplexed connectionless and connection-oriented communication over the LMP layer. L2CAP is proprietary, but other network protocols, such as IP, can be built on top of it. L2CAP is also used by higher level protocols.

Groups of up to eight Bluetooth devices can form ad hoc networks called *piconets* to communicate, share services, and synchronize data. In each piconet, a master device coordinates the other Bluetooth devices. Individual devices can participate in more than one piconet at a time and can be in one of several states:

1. Standby—the device is conserving power and waiting to connect to another Bluetooth device.
2. Inquire—the device is searching for nearby Bluetooth devices.
3. Page—the device is connecting to another Bluetooth device.
4. Connected—the device is connected to another Bluetooth device.
5. Hold and park—the device is participating in a piconet with varying degrees of power savings.

The Bluetooth SDP provides a simple API for enumerating the devices in range and browsing available services. It also supports stop rules that limit the duration of searches or the number of devices returned. Client applications use the API to search for available services either by service classes, which uniquely identify types of devices or by matching attributes. Attributes that describe the services offered by a Bluetooth device are stored as a service record and are maintained by the device's SDP server.

Bluetooth's SDP does not provide a mechanism for using discovered services—specific actions required to use a service must be provided by a higher level protocol. However, it does define a standard attribute ProtocolDescriptorList, which enumerates appropriate protocols for communicating with a service.
Bluetooth devices provide data security through unique 48-bit identifiers, 128-bit authentication keys, and 8- to 128-bit encryption keys. Bluetooth devices can form pairs so that it can

authenticate each other and protect sensitive data from snooping. Regardless of encryption strength, Bluetooth's fast frequency- hopping scheme makes snooping difficult.

3 Bridges

The above section gives us some idea about the variations in service discovery standards. In order that service discovery become pervasive, either a single service discovery technology must dominate or the most commonly used technologies must be made interoperable. In addition to this each service discovery technology has its own advantages and disadvantages. Therefore interoperability among different concepts seems a feasible solution, since it is very unlikely that device manufacturers will embrace multiple service discovery technologies on low-cost devices. In this section we will discuss some bridging protocols.

3.1 Salutation – Bluetooth

Two approaches to map Bluetooth SDP primitives to Salutation APIs are discussed in [6]. Before explaining the approaches the primitives which are used in Salutation and Bluetooth to locate services are discussed. This will aid in understanding the mapping of Salutation API to Bluetooth SDP functions.
The application programming interfaces provided by the Salutation Manager to Salutation API are Service Registration and Service Discovery:

Service Registration API:

1. slmRegisterCapability() - The slmRegisterCapability() function is called by Services to register their specific instances of Functional Units with the local Salutation Manager.
2. slmUnregisterCapability() – Services calls this function to unregister itself from the local Salutation Manager.

Service Discovery API:

1. slmSearchCapability() – clients calls this function to ask local SM to search for SMs having a registered Functional Unit with a specific capability.
2. slmQueryCapability() – clients calls this function to discover registered Functional Units and their capabilities at a specific SM.

The primitives provided by Bluetooth protocol stack to its user in order to performs its task are as follows

1. serviceBrowse – search for service based on attributes shared by all service classes (BrowseGroupList attribute) from the list of devices.
2. serviceSearch – search whether the devices supports the service mentioned.
3. enumerateRemDev – searches SDP Server in the vicinity of the client
4. getRemDevName – returns the list of the SDP Server identified by the SDP service searches.

Some of the transactions used by Bluetooth SDP are

1. ServiceSearch transaction: SDP client generates a SDP_ServiceSearchRequest to locate service records that match the service search pattern.
2. ServiceAttribute transaction: The SDP client generates an SDP_ServiceAttributeRequest to retrieve specified attribute values from a specific service record.

The 1st approach assumes that the Salutation APIs are implemented on top of the Bluetooth SDP. In this approach SDP attributes are passed in the Salutation APIs, i.e. Salutation APIs are implemented as the entry to Bluetooth SDP. SDP extracts the information it requires from the APIs and processes according to the mapping to SDP primitives.
The configuration used for mapping is shown in Figure 2.

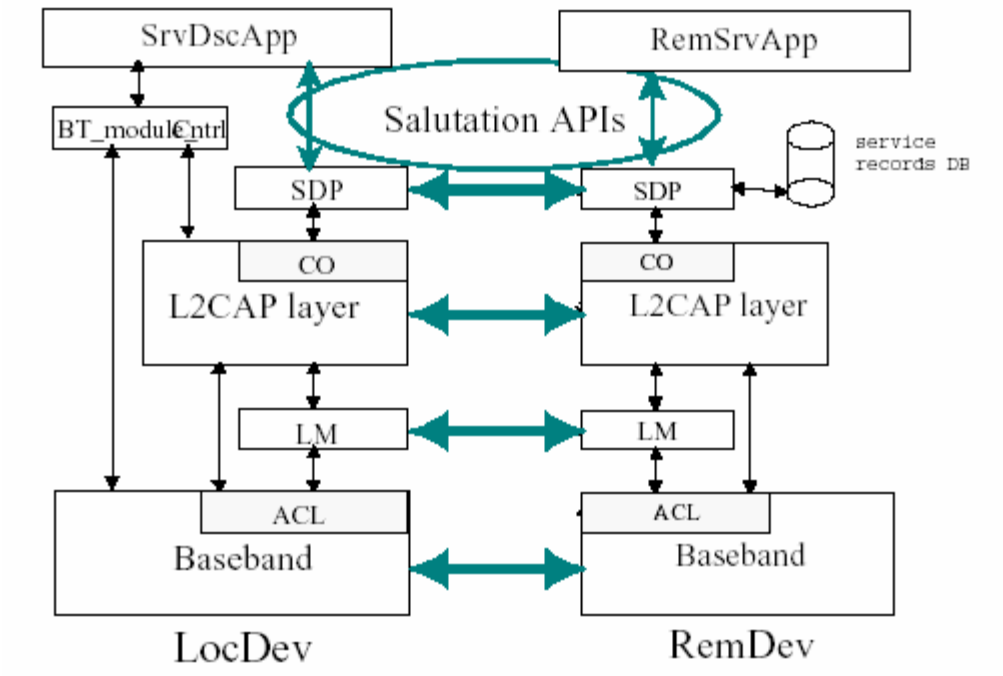

Figure 2: Salutation API Mapping to Bluetooth SDP

The table below shows the general mapping of SDP primitives to Salutation APIs. This mapping uses the functional definition of the Salutation APIs, but not the parameter values. The APIs becomes a vehicle for passing the SDP parameters to the Bluetooth SDP Manager. That is, the SLM-ID and Service Description Records parameters of the Salutation APIs are replaced with the appropriate SDP parameter values. The parameter returned from the search operations is a list of the names of the devices identified by the search.

SDP Primitives	Salutation API
serviceBrowse,getRemDevName	slmQueryCapability
serviceBrowse, getRemDevName	slmQueryCapability
enumerateRemDev, getRemDevName	slmSearchCapability
Undefined	slmRegisterCapability(),slmUnregisterCapability

The second approach assumed that the Salutation manager (SLM) can be mapped directly to the SDP protocol using a Bluetooth specific transport manager (TM). SDP is replaced by the Salutation manager mapping its functionality to the SDP protocol. Only the transport manager had to be rewritten.

The configuration used for this mapping is shown in Figure 3. The figure shows the use of the Salutation Manager (SLM) in both SDP Client (LocDev) and SDP Server (RemDev) to provide the service the management functionality of SDP. SLM exposes the existing Salutation APIs to the SrvDscApp and the RemSrvApp. SLM generates the appropriate SDP protocol through its TM, handing it off to L2CAP layer. SLM also responds to SDP protocol received from L2CAP.
The general mapping of Salutation Manager functions to the SDP protocol is shown in table below. LocDev and RemDev use the Salutation APIs to access their respective Salutation Managers. Bluetooth Service Discovery Protocol flows between the LocDev and RemDev Salutation Managers

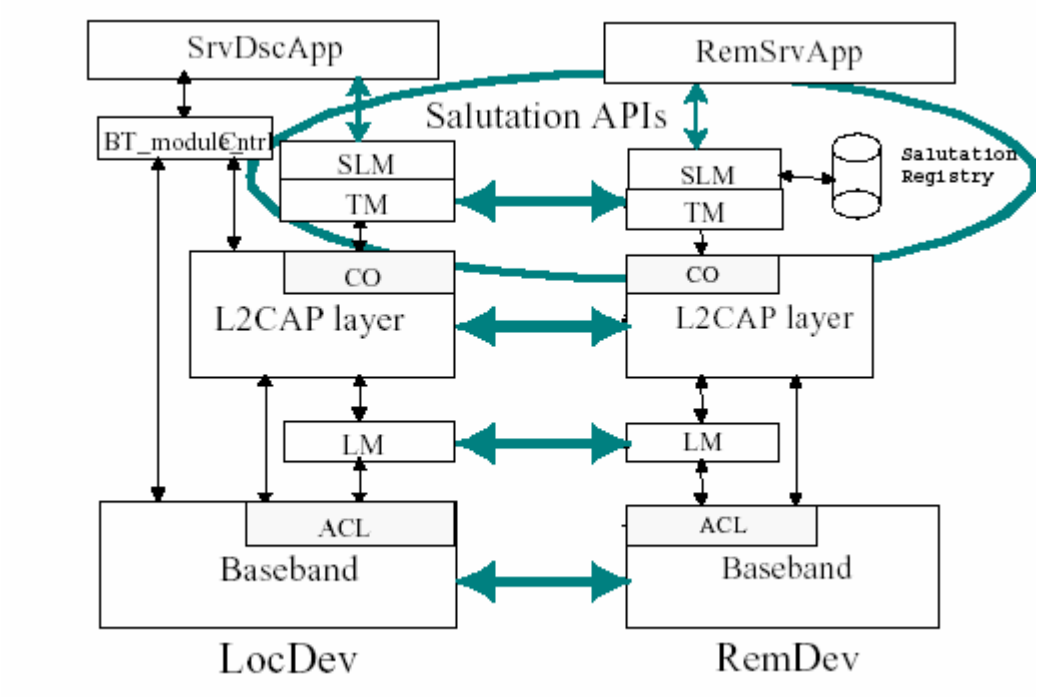

Figure 3: Salutation Manager Mapping to Bluetooth SDP Protocol

Salutation API	SDP Primitives
slmSearchCapability()	SDP_ServiceSearch
slmQueryCapability()	SDP_ServiceSearch, SDP_ServiceAttribute

Salutation Capability Search maps SDP Service Search Patterns to/from Salutation Functional Unit Description Records
Salutation Capability Query maps SDP Service Attributes to/from Salutation Functional Unit Description Records

3. 2 *SLP – Jini*

In Jini system, a device needs JVM environment to provide service. However some servers lack the resources to host JVM. Therefore we need a bridge that enables this server without JVM to provide service information and service object for registration to Jini.

SLP-Jini Bridge [7] is one such kind of bridge. In this approach servers equipped with SLP SAs advertise the availability of a Jini driver factory. The SLP-Jini Bridge acting as an SLP UA finds all services advertised with SLP that offer a Jini driver factory. Each such service advertisement is turned into a Jini service registration and registered with all appropriate Jini Look up servers. This service registration requires that the driver factory create a service object and insert appropriate attributes in the LUS.

The service and attribute objects registered with Jini are obtained using *driver loading*. This enables a system with no JVM (such as the servers on the left of Figure 3) to supply objects to a system with a JVM. Figure 3 illustrates how driver loading works.

1. The client system requests the Jini driver factory jar file from the server
2. The client downloads the jar file. This file contains a manifest that includes the name of the driver factory class.

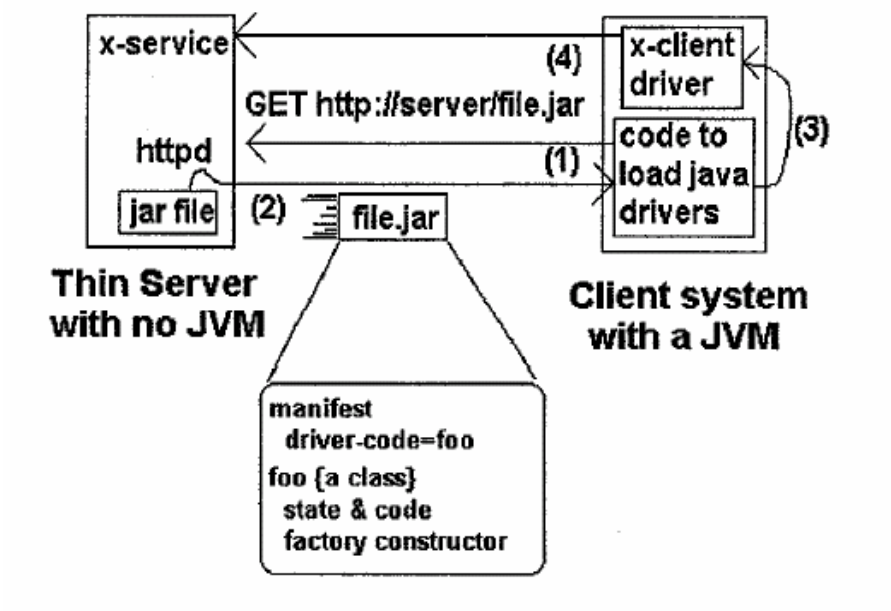

Figure 4: SLP-JINI Bridge

3. Using this manifest, the jar file and initialization arguments, the client instantiates the driver object.
4. This client is then used by client software to drive the service offered by the server. All classes that are necessary to offer service should be present in the jar file.

Whenever the attributes of these services change, the updates are forwarded to the LUS. Finally, when discovered services are no longer available, the bridge deregisters them with the LUS.

3.3 Salutation – SLP

Another type of bridge, Salutation SLP Bridge [9] discusses how salutation architecture can be enhanced to support directory based service discovery by utilizing SLP.

In this approach the Salutation Manager (SLM) search for a SLP directory agent through multicast, broadcast, or manual configuration. If it finds one, the SLM will use SLP protocol instead of Salutation protocol to register and un-register Functional Units it supports with the SLP directory. The SLM will also use SLP Protocol to search for services requested by Salutation client applications.

If directory service is not available or not necessary, the SLM may not find a SLP directory agent. In this case, SLM will use Salutation protocol broadcasts to find other SLMs and advertise the capabilities of the Functional Units it supports. Salutation Protocol is also used to locate the services requested by Salutation client applications.

The different message type which is used in SLP for service register, request and de-register are explained in Section 2.2. Salutation uses SLP messages type for utilizing SLP. As part of the FU registration process, a Service Agent sends a service registration “SrvReg” message to a Directory Agent. As a part of service registration, DA caches the relevant information and sends back an acknowledgement.

User Agents make requests to DAs when a service is required. There are different ways a UA may discover a DA. In addition to statically configuring the UA with the address of the DA, a UA may request the address of a DA using the Dynamic Host Configuration Protocol (DHCP). Or a UA may use the SrvRqst to find a DA. The SrvRqst is multicast to the IANA assigned multicast address for Service Location Protocol. The "service" requested in this message is called "directory-agent". Because this is a multicast request, it may receive more than one unicast reply. The resulting list of directory agents can then be used for other service requests. Once a UA has the address of a DA, subsequent service requests can be made directly to that entity.

3.4 Other Bridges

In addition to bridges discussed above there are few more bridges that aim to make different devices interoperable. One such protocol is Bluetooth Enhanced SDP for UPnP [12].

In a Bluetooth networked computing environment, it is desirable for devices to be able to discover services beyond their current piconet. In addition, extended service discovery could enable devices with Bluetooth wireless technology to control remote resources on other devices within and outside their piconets.

Characteristics of UPnP as discussed in Section 2.4 make it suitable as a Bluetooth Enhanced Service Discovery Protocol (ESDP) that is intended to provide an enhanced mechanism for service discovery and control within Bluetooth environments.
There are two different approaches for implementing UPnP within a Bluetooth system

1. L2CAP -based solution- the UPnP services is layered over the L2CAP layer of the Bluetooth protocol stack for use by devices that lack IP support.
2. IP-based solution – the UPnP services is layered over the Bluetooth Personal Area Networking (PAN) Profile and the Bluetooth Local Area Network (LAN) Access Profile; these profiles define IP support in the Bluetooth protocol stack.

The advantage of the IP based solution is that it can be used as a reference for layering Jini on the Bluetooth stack because the PAN as well as LAN profile provide IP based connections giving one more bridge.

4. Discussion & Future Work

The emergence of Internet has changed the way computing was performed. Advent of mobile and wireless computing has an enormous effect on this. The number of mobile user is constantly increasing, with different kind of devices such as PDA, laptop etc. Mobile user expects same kind of service which they used to enjoy in wired network. Service discovery technology aims to provide such kind of services to mobile user, so that services can be discovered and configured with minimal effort.

There are number of technologies in this area. Leading among there are Service Location Protocol, Salutation, Bluetooth, UPnP and Jini. Most of these technology promise similar functionality, namely to reduce configuration problem, improve device cooperation and automate discovery of required service.

Since none of these technologies are superset of others and for service discovery to be prevalent it is necessary that either one of them dominate or they are interoperable. Interoperability among them will require bridging mechanism. Different bridging protocols such as Salutation-SLP, Salutation-Bluetooth, SLP-Jini, Bluetooth-UPnP aim to provide interoperability among devices.

Though bridging seems to be a promising field, there are shortcomings in my opinion. First of all since there are many different technologies so each pair will require a different bridge. This will increase the number of bridges developed. Secondly if the primitives used by a particular technology are operating environment dependent, then portability can be issue.

Therefore what is needed is a common platform or standard interfaces that would be interoperable with all Service Discovery technology. Also as mentioned in [17] another area of research is to extend current service discovery frameworks to support context awareness. Also issues related to mobile user such as support in wide area network need to addressed.

5. Conclusion

Service Discovery has come a long way to become a major standard and development effort. This paper attempts to take a look at key service discovery protocols that address more or less the same concepts. It gives a brief summary of SDP such as Service Location Protocol, Salutation, Jini, Bluetooth and Universal Plug and Play in terms of how to discover and register services.

The table below gives a comparison between different SDP.

Features	Bluetooth	Jini	Salutation	UPnP	SLP
Service Discovery	Yes	Yes	Yes	Yes	Yes
Service Announcement		Yes	Yes	Yes	Yes
Service Registry		Yes	Yes		Yes
Interoperability	Yes	Yes	Yes	Yes	Yes
Security	Yes	Yes			Yes

It also explores different bridging protocols which aim to make different standard interoperable. The architectural issues regarding SLP-Salutation, Salutation-Bluetooth and SLP-Jini is discussed in detail while providing the approach used to bridge Bluetooth over UPnP.

Among so many standards there seems to be no clear choice today. All of them have wonderful architectures but technological superiority is not the only factor to be prevalent in the market. These service discovery protocols are contending with one another to be final winners. But on the other hand, they are also making efforts to integrate themselves with other protocols. Therefore more changes and competitions are awaited.

References

1. Salutation's Homepage: http://www.salutation.org
2. Microsoft's Universal Plug and Play Homepage: http://www.upnp.com
3. Sun Microsystems, Jini Connection Technology: http://www.sun.com/ jini
4. Sun Microsystems, .Jini Community Resources: Jini Technology Architectural Overview. January 1999. http://www.sun.com/jini /whitepapers/architecture.html
5. Specification of the Bluetooth System*;* available at http://www.bluetooth.com/developer/specification/specification.asp
6. Brent Miller, .Mapping Salutation Architecture APIs to Bluetooth Service Discovery Layer,. Bluetooth Consortium 1.C.118/1.0, July 1, 1999.

7. 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.
8. 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
9. Pete St. Pierre, Sun Microsystems and Tohru Mori, IBM Japan “Salutation and SLP” http://www.salutation.org/techtalk/slp.htm
10. Bluetooth Consortium, .Specification of the Bluetooth System Core Version 1.0 B: Part E, Service Discovery Protocol (SDP),. Nov 29, 1999. http://www.bluetooth.com/developer/specification/specification.asp
11. E. Guttman, C. Perkins, J. Veizades and M. Day,. Service Location Protocol, Version 2, IETF RFC 2608, June 1999. http://www.ietf.org/rfc/rfc2608.txt
12. Bluetooth ESDP for UPNP http://www.bluetooth.com/pdf/ESDP_UPnP_0_95a.pdf
13. *Perkins, C.E.* “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 vol.1
14. M. L. Diane , T. Noel and J. J. Pansies: “Extension of Service Location Protocol for IPv6 Communication Mobility”
15. C. Lee, A. Heal, “Protocols for Service Discovery in Dynamic and Mobile Networks,” The International Journal of Computer Research, Special issue on Wireless Systems and Mobile Computing, September 2000
16. “The Application of Jini Technology to Enhance the Delivery of Mobile Services”, white paper (Dec 2001). http://wwws.sun.com/software/jini/whitepapers/index.html
17. S. Helal, “Standards for service discovery and delivery IEEE Pervasive Computing,” IEEE Pervasive Computing, vol , 1 no. 3 , 2002 ,pp. 95 - 100
18. Erik Guttmann, “Service Location Protocol Modifications for IPv6” IETF, draft-ietf-mobileip-ipv6-13.txt, November 2000.