mSLP - Mesh-enhanced Service Location Protocol *

Weibin Zhao *and* Henning Schulzrinne
Department of Computer Science
Columbia University, New York, NY 10027
{zwb, hgs}@cs.columbia.edu

Abstract

The Service Location Protocol (SLP) is a proposed standard from IETF. It provides a flexible and scalable service discovery framework in IP network, and it can work with or without a directory service. This paper presents mSLP - Mesh-enhanced Service Location Protocol. mSLP proposes to use a fully meshed peering Directory Agent (DA) architecture. Peer DAs exchange service registration information, and keep the same consistent data for the shared scopes. mSLP provides a reliable directory service for an SLP system. It also greatly simplifies SLP service registration leading to a thin-client Service Agent (SA) implementation. mSLP is backward compatible with SLPv2, and incremental deployment is supported.

Keywords: Service Discovery, Service Location Protocol, User Agent, Service Agent, Directory Agent, Fully Meshed Peer Relationship, Reliability, Scalability

1 Introduction

As computing continues to move towards a network-centric model, there is an increasing need to find and make use of the available services, devices, and applications in a variety of networking environments in order to properly complete specified tasks. For example, the mobile technology provides much convenience for a traveling user, it relies upon the support from automated service discovery. As a result, several service discovery protocols are emerging from network computing field to address this issue.

The Service Location Protocol (SLP) [4] from IETF is designed to perform service discovery in IP network. Jini [7] from Sun Microsystems is tailored specifically to Java platform. The Universal Plug and Play (UPnP) [9] from Microsoft proposes to use SSDP (Simple Service Discovery Protocol) [2] for service discovery, and it addresses the issue of automatic configuration such as the assignment of IP addresses and DNS names. The Salutation Consortium [8] has announced an open architecture for service discovery. The Bluetooth SIG [5] presents its own SDP (Service Discovery Protocol) optimized for the highly dynamic nature of wireless communication using radio link. The Infrared Data Association (IrDA) [6] also offers its unique service discovery protocol.

Although different service discovery protocols have different features, they use two basic models. In the directory-centric model, there is a centralized directory service which maintains dynamic information about available network services, devices and applications. Each device, service and application must discover this directory service first in order to register with it or use it to look up the information about other devices, services and applications. In the peer-to-peer model, no centralized directory service is needed. Devices, services and applications negotiate one-on-one with each other, announcing their presence, advertising their own capabilities, and finding devices, services or applications that meet their needs. This model fits perfectly in an ad hoc peer-to-peer networking environment. Jini adopts the directory-centric model with a lookup service while IrDA uses the peer-to-peer model. SLP and UPnP support both models.

No matter what model is used, the underlying principles for service discovery are same. First, an entity (device, service or application) needs to announce its presence to others. This can be done by registering with a directory service, by regularly advertising through a special channel or address, or by listening to a special channel or address and issuing a reply when there is a request. Second, the service discovery is performed by looking up a directory service, by sending a discovery request to a special channel or address, or by listening to a special channel or address for others announcements.

Most service discovery protocols utilize multicast (or broadcast) which is needed at least in the initial discovery stage such as discovering a lookup service. In some cases, information is needed if none of above mechanisms exists.

Among the current available service discovery protocols, SLP is a proposed standard from IETF. It has received wide industry support. For example, Sun Microsystems has built SLP into its Solaris 2.8 operating system; Novell supports SLP in its NetWare product. SLP can work with or without a directory service. The directory service is introduced for performance and scalability considerations, and it is provided by Directory Agents (DAs). However, SLP does not

*Supported by DARPA MarketNet project.

define how DAs should coordinate with each other when multiple DAs exist.

This paper presents mSLP - Mesh-enhanced Service Location Protocol, which enhances SLP with an interaction scheme for DAs. It proposes that if DAs are needed in an SLP system, a fully meshed peering DA architecture should be used, i.e., more than one DA should be present for each scope, and they should maintain a fully meshed peer relationship. Peer DAs exchange service registration information, and keep the same consistent data for the shared scopes. mSLP provides a reliable directory service for an SLP system. It also greatly simplifies SLP service registration leading to a thin-client Service Agent (SA) implementation. Thus, the scalability of an SLP system can also be enhanced. Moreover, mSLP is entirely backward compatible with the SLP specification defined in RFC 2608, and incremental deployment is supported.

The rest of this paper is organized as follows: first, we provide some background information about SLP in Section 2, then we describe the design of mSLP in Section 3, covering design considerations, peer relationship management and message forwarding control. We show an mSLP system example in Section 4, and discuss the implementation in Section 5. We list related work in Section 6, and conclude in Section 7.

2 SLP Overview

Defined in Request for Comments (RFC) 2608, SLP provides a flexible and scalable framework for service discovery in IP network. Using SLP, a networked service (device, or application) can easily find all available service types, the locations (URLs) where a specific service is provided, and service descriptions. To understand how SLP works, we first review several important concepts SLP used to describe and manage networked services.

Service Location Service locations in SLP are given by URLs, which can be any URL conforming to URI standard [1] such as "http://www.srvloc.org", or can be of the service: scheme defined in [3] such as "service:lpr://mandolin.cs.columbia.edu".

Service Type Each service in SLP has a service type. For example, the service type of "http://www.srvloc.org" is "http" (web service); the service type of "service:lpr://mandolin.cs.columbia.edu" is "service:lpr" (printing service). All services in SLP are categorized by their service types.

Service Description Besides service location and service type, each service in SLP has many other properties which can be described by a set of attribute/value pairs. These properties include service capability and configuration parameters. For example, we may specify "resolution = 1200 dpi" for a printer service.

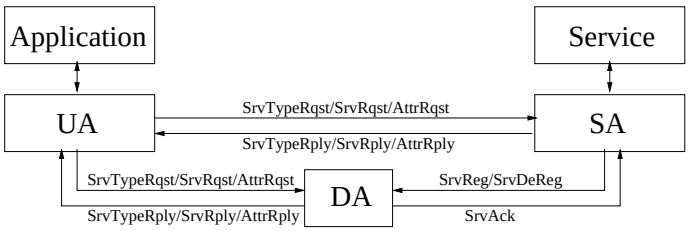

Figure 1: SLP System Architecture

Service Scope Each service in SLP has one or more service scopes which specify its service range. The service scope is a logical concept in SLP used to arrange services into groups. A scope could indicate a geographic location (such as "London"), a administrative group (such as "Law School"), or other category (such as "Emergency"). Service scope provides further scalability for SLP in large systems.

Service Lifetime Each service registration has a validity period given by its service lifetime (such as 12 hours). A registration needs to be refreshed before its lifetime has expired, otherwise the registration will be removed from the directory service.

There are three different entities in an SLP system: User Agents (UAs), Service Agents (SAs), and Directory Agents (DAs). Figure 1 illustrates their relationship.

User Agents (UAs) A UA initiates service discovery on behalf of one or more applications. It sends queries to all SAs via multicast or to a DA via unicast if there is such DA supporting the query scope. A UA uses three different types of SLP messages to discover the desired services. First, it issues a service type request (`SrvTypeRqst`) message to get a list of all available service types in a service type reply (`SrvTypeRply`) message. Second, it issues an attribute request (`AttrRqst`) message to get a list of all attributes for a given service type in an attribute reply (`AttrRply`) message. Third, it issues a service request (`SrvRqst`) message with an attribute predicate specifying the characteristics of the desired service to get a list of URLs giving the locations of matched services in a service reply (`SrvRply`) message. Starting from the desired service type, and a set of attributes describing the service, SLP resolves the service addresses (URLs) for the user.

Service Agents (SAs) An SA works on behalf of one or more services. It responds directly to UA queries. If DAs exist, it registers the services with DAs using service registration (`SrvReg`) messages. SLP supports incremental service registration in which an SA can

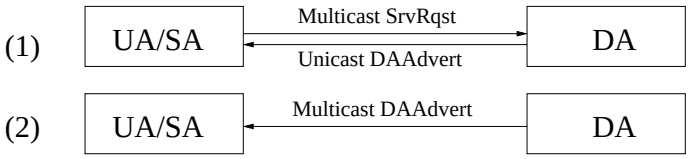

Figure 2: DA Discovery: (1) Active (2) Passive

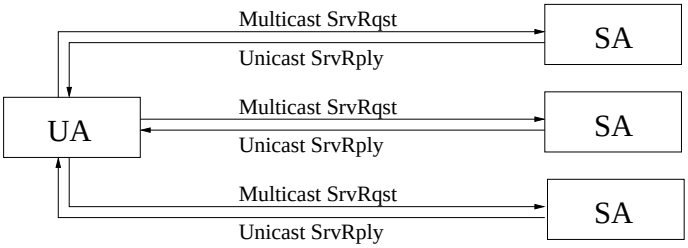

Figure 3: Small SLP System without DAs

update attribute values or add new attributes to a previously registered service. The new attributes replace the old values, but do not affect the attributes which were included previously and are not present in the update. Thus, a `SrvReg` message can be a fresh service registration or an update to a previous registration. An SA can also deregister the services with DAs using service deregistration (`SrvDeReg`) messages before the registered services become expired.

Directory Agents (DAs) A DA serves as a centralized information repository in an SLP system. It accepts SA registrations and answers UA queries. DA support is optional in an SLP system; it is introduced for performance and scalability considerations. DAs can be discovered (Figure 2) in two ways: active or passive. In SLP, each DA periodically sends unsolicited DA advertisement (`DAAdvert`) messages to the administratively scoped SLP multicast [10] address (239.255.255.253). All UAs, SAs and DAs listen to this address to discover the existing DAs. This is known as passive DA discovery. UAs and SAs can also perform active DA discovery by issuing a multicast DA discovery service request message (`SrvRqst` with a service type of "service:directory-agent") to above address. DAs answer each DA discovery request with a unicast DA advertisement message.

Whether DAs exist or not, an SLP system can provide the same service discovery functionality. However, it works a little bit differently.

- In a small SLP system without DAs (Figure 3), UAs directly send service request (`SrvRqst`) messages to all SAs via multicast, and SAs respond with unicast service reply (`SrvRply`) messages.

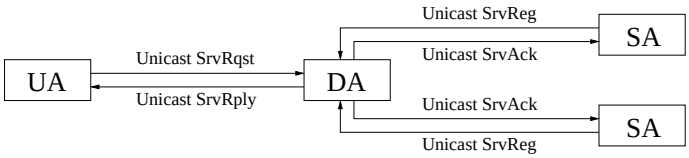

Figure 4: Medium SLP System with DAs

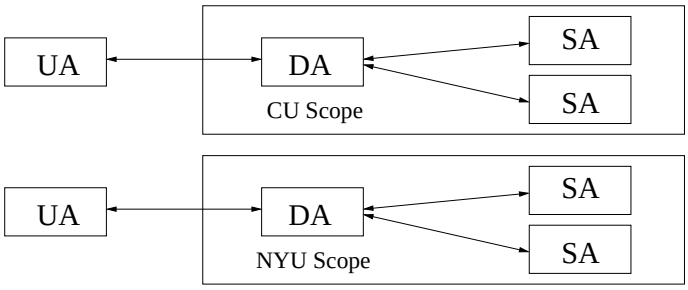

Figure 5: Large SLP System with Multiple Scoped DAs

- Since it is not very efficient for a UA to query all SAs via multicast every time it does service discovery, in a medium-sized SLP system (Figure 4), DAs are introduced to provide a centralized directory service. SAs register services with DAs, and UAs look up service information from DAs. All service registrations from SAs and service queries from UAs are sent to DAs via unicast.

- In a large SLP system (Figure 5), DAs are arranged into different scopes. For example, services from Columbia University and NYU are assigned to different scopes served by different DAs. This ensures further scalability of SLP.

We can see that SLP achieves scalability by using DAs and service scopes. However, the centralized directory service provided by DAs presents a single failure point in an SLP system. To ensure reliability, multiple DAs are needed for each service scope. When multiple DAs are present, how should they interact with each other? This is an open issue in SLPv2 (SLP version 2).

3 Design of mSLP

mSLP enhances SLP with an interaction scheme for DAs to provide a reliable directory service. It proposes that if DAs are needed in an SLP system, a fully meshed peering DA architecture should be used, i.e., more than one DA

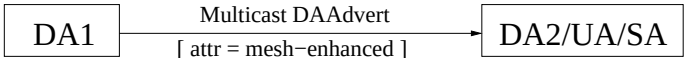

Figure 6: Mesh-enhanced DA Advertisement

should be present for each scope, and they maintain a fully meshed peer relationship. Peer DAs exchange their data for the shared scopes when they set up a peer relationship, and continue to exchange new service registration information during the entire peering period. As a result, peer DAs maintain the same consistent data for the shared scopes. Our design uses the following terminology:

Peer DAs If two DAs have one or more scopes in common within one administrative domain, they are peers. Peer DAs coordinate with each other and store the same consistent data for the shared scopes.

Peering TCP Connection - a persistent TCP connection kept by a pair of peer DAs for the entire peering period. It provides a reliable communication channel for the peer DAs to exchange messages. Therefore, a DA implementation is not burdened by managing message retransmissions. Its closing can be an indication of the termination of the peer relationship.

Mesh-enhanced DA - a DA who maintains a fully meshed peering DA architecture with other DAs in an mSLP system. It carries the "mesh-enhanced" attribute in its `DAAdvert` messages (Figure 6) to distinguish itself from non-mesh-enhanced DAs.

Mesh-aware SA - an SA who understands the "mesh-enhanced" attribute in a `DAAdvert` message and uses the mesh-enhanced DA capability accordingly.

In the next three subsections, we first describe our design considerations for mSLP, then we discuss two important issues: how to manage the peer relationship and how to control message forwarding among peer DAs.

3.1 Design Considerations

Reliability: The fully meshed peering DA architecture provides a reliable directory service for an SLP system. It achieves maximum robustness by ensuring that no single failure point exists in the system. All service registrations are kept by at least two DAs. If one DA is down, at least one other peer DA can continue to provide the service information for the scope. More importantly, the peering DA architecture enables a DA to recover from crash with much less effort since a rebooted DA can get the existing data from its peering DA set.

The fully meshed peering DA architecture is built on top of a set of fully meshed peering TCP connections. We choose to use this set of fully meshed peering TCP connections mainly because it greatly facilitates message forwarding control. Any service registration information received by a DA only needs one-hop forwarding to reach all other DAs in the peering DA set. Since adding too many DAs to a peering DA set does not help a lot with the reliability, and it increases the maintenance overhead, we anticipate a small number of DAs for each service scope, and 2-4 is the recommended value. There is no need to have separate DA for each scope. A DA can serve multiple scopes, and a peering TCP connection is used for all shared scopes between each pair of peer DAs.

Scalability: The SLPv2 specification requires that SAs register with all existing DAs and re-register when new DAs are discovered, or old DAs are found to have rebooted. This places a substantial burden on an SA implementation. With mSLP, this requirement on SAs can be greatly simplified, i.e., a mesh-aware SA only needs to register with one mesh-enhanced DA in the registration scope, and the registration information will be propagated automatically within the meshed DA set. With mesh-enhanced DAs and simplified SAs, the overall system scalability can be enhanced.

Compatibility: mSLP is fully backward compatible with SLPv2. It does not introduce any new protocol elements. mSLP only defines several new attributes for the `DAAdvert` message and a new SLP extension ("mesh-forwarding extension").

As we have described in Section 2, each DA periodically multicasts unsolicited `DAAdvert` messages, which enable a DA to be discovered by UAs, SAs and other DAs. A DA can put new attributes in its `DAAdvert` messages to signify the information for DA coordination. Thus, by properly setting the attributes in `DAAdvert` messages and properly handling these attributes at the other party, peer DAs can interact with each other to provide the desired functionality.

The DA coordination is added as an enhancement to a DA, its original functions are not affected. Moreover, the changes are almost transparent to UAs and SAs. UAs can be kept unchanged. SAs can be greatly simplified if they use the DA mesh-forwarding capability for their service registrations. To do this, a mesh-aware SA needs to explicitly specify its mesh-forwarding request by using the mesh-forwarding extension (see Section 3.3 for detail).

Deployment: mSLP supports incremental deployment of mesh-enhanced DAs. A mesh-enhanced DA can set up peer relationships with both mesh-enhanced DAs and non-mesh-enhanced DAs. In the latter case, the message forwarding only goes in uni-direction from the mesh-enhanced DA to a non-mesh-enhanced DA. When mesh-enhanced DAs coexist with non-mesh-enhanced DAs in a system, a mesh-aware SA should take care of newly found non-mesh-enhanced DAs and rebooted non-mesh-enhanced DAs since these non-mesh-enhanced DAs can not get existing data from other DAs. Therefore, uniformly mesh-enhanced DAs can provide a much easier job for mesh-aware SAs.

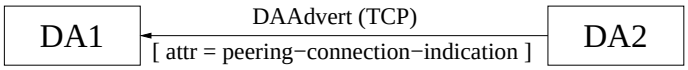

Figure 7: Peering Connection Indication

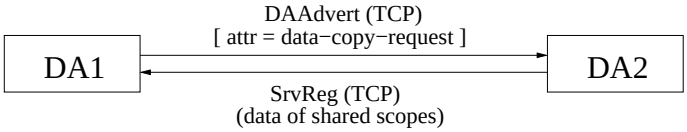

Figure 8: Dumping Data

3.2 Peer Relationship Management

A peer relationship has three stages: setting up, keeping, and tearing down. In mSLP, a mesh-enhanced DA maintains a peer list. It adds an entry to the list whenever it discovers a new peer, and removes an entry when it finds the corresponding peer is down. Each entry in this peer list should include peer URL, shared scopes, boot timestamp, and peering TCP connection ID. The boot timestamp is used to identify a rebooted peer. The peering TCP connection is used for message forwarding.

When a mesh-enhanced DA learns about a new peer, it creates a peering TCP connection to the peer (at port 427) if such connection does not yet exist. Then it sends a `DAAdvert` message with the "peering-connection-indication" attribute through this channel to notify the peer that this is a peering TCP connection (Figure 7). Thus, the peer can also use this channel to send messages in opposite direction.

After the peering TCP connection is established or identified, peer DAs begin to forward new service registration information to each other via this channel. At the same time, a mesh-enhanced DA should decide whether it needs to download the data from its new peers. For example, when a newly booted DA joins a peering DA set of three DAs, it needs to get a copy of the existing registration data from any one of these three DAs (the choice can be made randomly), but not from all of them which incurs a lot of redundant transmissions. To do this, it sends a `DAAdvert` message with the "data-copy-request" attribute to the chosen peer (Figure 8). On the other end, when a mesh-enhanced DA receives the "data-copy-request" in a `DAAdvert` message, it dumps all the data of the shared scopes to the requesting DA. Each data record is sent as a `SrvReg` message, with a re-calculated new lifetime (= old lifetime − elapsed time). After exchanging the data in both directions, peer DAs have the same consistent data for the shared scopes.

According to SLPv2, a DA could close a TCP connection if it has been idle for too long. To keep a peering TCP connection alive, a mesh-enhanced DA sends a `DAAdvert` message with the "keepalive" attribute (Figure 9) to the peer if no other messages have been sent via the channel for a long period (predefined).

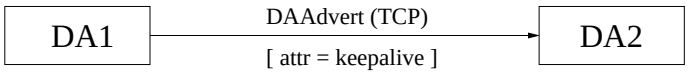

Figure 9: DA Keepalive

A mesh-enhanced DA should tear down a peer relationship when it finds that the peer has closed the peering TCP connection; when it receives a multicast `DAAdvert` message from the peer with a DA stateless boot timestamp set to 0 meaning the peer is going to shutdown; or when it has not received any message from the peer via the peering channel for a long period (predefined). In the last case, there may be a network partition and peer DA states get inconsistent. To tear down a peer relationship, a DA stops forwarding any service registration information to this peer, closes peering TCP connection with this peer, and removes this peer from its peer list.

3.3 Message Forwarding Control

During the peering period, peer DAs forward new service registration information from SAs to each other. We define a new SLP extension - "mesh-forwarding extension" for the message forwarding purpose. This extension is always six bytes long: five-byte extension header and one-byte extension data denoted as "Action" field. The "Action" field is used to specify forwarding actions: TO_BE_FORWARDED marks the message needs to be forwarded; HAS_BEEN_FORWARDED means the message has already been forwarded, and there is no need to be forwarded again. The message forwarding rules are as follows:

1. Explicit Forwarding: A message is forwarded only when it explicitely requests to be forwarded. A mesh-aware SA needs to attach the mesh-forwarding extension to a `SrvReg` or `SrvDeReg` message and set the "Action" field to TO_BE_FORWARDED if it wants the message to be forwarded by a mesh-enhanced DA. This is for the compatibility with SLPv2, where SAs need to register with all existing DAs. To avoid unnecessary forwarding, the explicit forwarding rule is used.

2. One-hop Forwarding: A `SrvReg` or `SrvDeReg` message is forwarded only once by a mesh-enhanced DA to all its peers in the registration scope. Before forwarding a message, a mesh-enhanced DA sets the "Action" field to HAS_BEEN_FORWARDED. In this way, a forwarded message will never be forwarded again. Since the peering DA set is in a fully connected mesh, one-hop forwarding is enough to ensure that the service registration information from an SA can reach all peer DAs. Figure 10 shows how a service registration message is forwarded.

3. Handling `SrvAck`: A DA always replies with a ser-

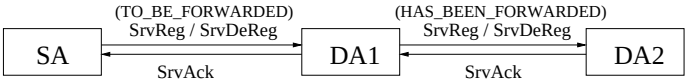

Figure 10: Forwarding Registration

vice acknowledgment (`SrvAck`) message when it receives a `SrvReg` or `SrvDeReg` message. So a mesh-enhanced DA should process `SrvAck` messages from other DAs.

4 Example

Let's go through an example to see how mSLP works. Considering that we want to deploy an mSLP system in Columbia University main campus, and we would like to assign the services in Law School (L-School), Business School (B-School) and Engineering School (E-School) to three different scopes. Instead of using a separate DA for each school, mSLP uses three DAs in a fully meshed peering architecture. Each DA serves two scopes and each school is served by two DAs (Figure 11). For example, DA1 serves both B-School and E-School, and L-School is served by DA2 and DA3.

We can run this example on four computers: one for the UA/SA, and three for the DAs. Each machine can host only one DA since a DA needs to bind to the SLP reserved port 427. An mSLP system can be in one of the three stages:

1. Normal stage: This is a stable stage featuring automatic registration information distribution. Assume that the mesh-aware SA does service registrations with the mesh-forwarding extension, say it registers services in B-School with DA1, services in E-School with DA2, and services in L-School with DA3, then this mSLP system will distribute the service registration information automatically among peer DAs for the shared scopes. Now let the UA query service information in L-School with DA2, it will get the same data as what the SA has registered with DA3.

2. Under partial DA crash: This is an unstable stage needs to be taken care of. Assume one of the three DAs crashes, say DA1 (we can simulate its crash by removing its network connection and then turning it off), this mSLP system can still function well unless more DAs crash. Although the data of B-School and E-School are not available from DA1, they can be retrieved from DA3 and DA2, respectively.

3. Recovery from crash: When a crashed DA comes up, say DA1 (we reconnect DA1 machine to the network and reboot it), it sets up peer relationship with other DAs again. Since now DA1 carries a new boot timestamp, DA2 and DA3 knows that it is rebooted and coordinates with it. DA1 retrieves B-School data from DA3 and E-School data from DA2, then this peering DA set is back to normal. Thanks to the fully meshed peering DA architecture, the recovery of DA1 from crash is done automatically through DA coordination, no SA involvement is needed.

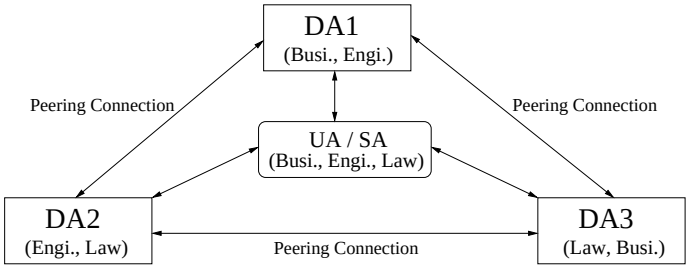

Figure 11: an mSLP System Example

5 Implementation

We have done a prototype implementation of mSLP. It is available at http://www.cs.columbia.edu/~zwb/project/slp. A little more effort is needed for an mSLP DA implementation which requires a peer relationship management and service registration forwarding control. However, an mSLP SA implementation is greatly simplified since it no longer needs to implement the complicated algorithm to register with all existing DAs and to re-register when new DAs are discovered, or old DAs are found to have rebooted. So the overall implementation cost of mSLP is about the same as SLPv2 if it is not reduced.

6 Related Work

Our early work on mSLP was presented as an Internet Draft in "Interaction of SLP Directory Agents for Reliability and Scalability" [14]. The fully meshed peer relationship is used in IBGP [13]. Redundancy is a basic method to ensure reliability, the DNS [11, 12] primary and secondary server architecture is a good example of this.

7 Conclusion

In this paper we presented mSLP - Mesh-enhanced Service Location Protocol. It enhances SLP with a fully meshed peering DA architecture. mSLP can provide a reliable directory service for an SLP system. It can also greatly simplify service registration which can lead to a thin-client SA implementation. mSLP is fully compatible with SLPv2, and it supports incremental deployment.

A further extension to the interaction of mSLP DAs is to forward UA queries besides SA registrations. It works as follows: When a mesh-enhanced DA receives a UA query which is not in its scope, it forwards the query to another DA which supports the scope. This can simplify UA implementation since UAs do not need to keep track of DA

scopes. A UA can send its queries to any mesh-enhanced DA. However, this adds much complexity to the mesh-enhanced DA implementation. First, a mesh-enhanced DA needs to keep track of all DAs of all scopes, not only the DAs that share some scopes with it. Second, a mesh-enhanced DA needs to forward the query to another DA, and it also needs to forward the reply from another DA back to the UA. mSLP does not include this extension mainly due to its complexity. However, for a thin-client UA implementation, it might deserve further considerations.

Acknowledgments: Erik Guttman provided valuable comments and suggestions for our work on mSLP.

References

[1] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifiers (URI): generic syntax. Request for Comments 2396, Internet Engineering Task Force, August 1998.

[2] Y. Goland, T. Cai, P. Leach, Y. Gu, and S. Albright. Simple service discovery Protocol/1.0 operationg without an arbiter. Internet Draft, Internet Engineering Task Force, November 1999. Work in progress.

[3] E. Guttman, C. Perkins, and J. Kempf. Service templates and service: Schemes. Request for Comments 2609, Internet Engineering Task Force, June 1999.

[4] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service location protocol, version 2. Request for Comments 2608, Internet Engineering Task Force, June 1999.

[5] Bluetooth home page. http://www.bluetooth.com/.

[6] IrDA home page. http://www.irda.org/.

[7] Jini home page. http://www.sun.com/jini/.

[8] Salutation home page. http://www.salutation.org/.

[9] UPnP home page. http://www.upnp.com/.

[10] D. Meyer. Administratively scoped IP multicast. Request for Comments 2365, Internet Engineering Task Force, July 1998.

[11] P. V. Mockapetris. Domain names - concepts and facilities. Request for Comments 1034, Internet Engineering Task Force, November 1987.

[12] P. V. Mockapetris. Domain names - implementation and specification. Request for Comments 1035, Internet Engineering Task Force, November 1987.

[13] Y. Rekhter and T. Li. A border gateway protocol 4 (BGP-4). Request for Comments 1771, Internet Engineering Task Force, March 1995.

[14] W. Zhao and H. Schulzrinne. Interaction of SLP directory agents for reliability and scalability. Internet Draft, Internet Engineering Task Force, April 2000. Work in progress.