mSLP - Mesh-enhanced Service Location Protocol *

Weibin Zhao†, Henning Schulzrinne† *and* Erik Guttman‡
†Columbia University {zwb,hgs}@cs.columbia.edu
‡Sun Microsystems erik.guttman@germany.sun.com

Abstract–**The Service Location Protocol (SLP), a proposed IETF standard, provides a flexible and scalable service discovery framework for IP networks. It can be deployed with or without a directory service. This paper presents mSLP, the Mesh-enhanced Service Location Protocol. mSLP enhances SLP with a fully-meshed peering Directory Agent (DA) architecture. Peer DAs exchange service registration information, and maintain the same consistent data for shared scopes. mSLP improves the reliability and consistency of SLP directory services. It also greatly simplifies Service Agent (SA) registrations in systems with multiple DAs. mSLP is backward compatible with SLPv2 and can be deployed incrementally.**

Keywords–**Service discovery, Service Location Protocol, directory replication, fully-meshed peer relationship, reliability, consistency, scalability.**

1 Introduction

As computing continues to move towards a network-centric model, there is an increasing need to automatically find available network services and devices without administrative support in order to properly complete specified tasks, especially in mobile and ad-hoc networking environments. As a result, service discovery systems and protocols are emerging to address this issue, such as the Service Location Protocol (SLP) [7], Jini [9], the Simple Service Discovery Protocol (SSDP) [5] in Universal Plug and Play (UPnP) [11], efforts by the Salutation Consortium [10], the Service Discovery Protocol (SDP) in Bluetooth [8], the Berkeley Service Discovery Service (SDS) [3] and the MIT Intentional Naming System (INS) [1]. These systems use directory-centric and peer-to-peer service discovery models, with some systems combining both. In the directory-centric model, a directory service maintains dynamic information about available network services and devices. Services, devices and applications need to discover the directory service first and then either register with it or use it to look up service information. In the peer-to-peer model, there is no centralized directory service. Services, devices and applications interact directly with each other, announcing their presence, advertising their own capabilities, and finding service information. No matter what model is used, the basic mechanism for service discovery is the same. First, a service or device needs to announce its presence by registering with a directory service, by issuing regular multicast announcements, or by listening for multicast requests and sending unicast replies. Second, services are discovered by looking up a directory service, by sending multicast discovery requests, or by listening to a designated multicast address for service announcements.

As a proposed IETF standard, SLP supports both directory-centric and peer-to-peer service discovery models. In SLP, directory services are provided by Directory Agents (DAs). However, SLP does not define how DAs should coordinate with each other when multiple DAs exist, so we developed mSLP, the Mesh-enhanced Service Location Protocol, which defines a scheme for the interaction of SLP DAs. mSLP proposes that if DAs are needed in an SLP deployment, 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 shared scopes. mSLP improves the reliability and consistency of SLP directory services. It also greatly simplifies Service Agent (SA) registrations in systems with multiple DAs. mSLP is backward compatible with SLPv2 and can be deployed incrementally.

The rest of this paper is organized as follows. We first review SLP in Section 2, then we describe mSLP design considerations, peer relationship management and message forwarding control in Section 3, 4 and 5. We show an example of how mSLP works in Section 6, and discuss our implementation in Section 7. We list related work in Section 8, and conclude in Section 9.

2 SLP Overview

SLP provides a flexible and scalable framework for service discovery in IP networks, allowing a user to conveniently find available service types, the locations (URLs) where a specific service is provided, and service descriptions.

We first review some SLP terminology. *Service locations* are described by URLs [2] such as *http://www.srvloc.org*, or identified by the "service:" URL scheme [6] such as *service:lpr://mandolin.cs.columbia.edu*. Each service has a *service type*, e.g., the service type of *http://www.srvloc.org* and *service:lpr://mandolin.cs.columbia.edu* is *http* (web service) and *service:lpr* (printing service), respectively. *Service descriptions* are expressed as attribute/value pairs such as "resolution = 1200 dpi" for a printing service. SLP uses *service scopes* to arrange services into groups. A scope could indicate a geographic location such as "London", an administrative group such as "Law School", or other category such as "Emergency". Each service registration is valid for its specified *ser-*

*Supported by DARPA MarketNet project.

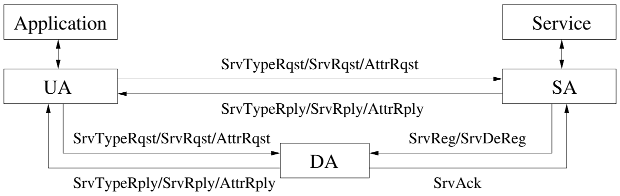

Figure 1: SLP System Architecture

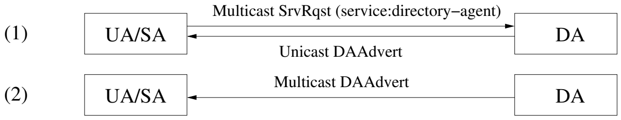

Figure 2: DA Discovery: (1) Active (2) Passive

vice lifetime (such as 12 hours). Unless it is refreshed before its lifetime expires, it is removed from the directory service.

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

User Agents (UAs) initiate service discovery on behalf of applications. A UA sends queries to all SAs via multicast or, if available, to a DA via unicast. A UA uses three different types of SLP messages to discover the desired services: a service type request (`SrvTypeRqst`) message to get a list of all available service types in a service type reply (`SrvTypeRply`) message, an attribute request (`AttrRqst`) message to get a list of all attributes for a given service type or a specific service instance in an attribute reply (`AttrRply`) message, and 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. `SrvTypeRqst`, `SrvTypeRply`, `AttrRqst` and `AttrRply` messages allow a user to interactively browse for available service types and their attributes, which can be used to construct service queries in `SrvRqst` messages. Given the desired service type, and a set of attributes describing the service, SLP derives the service addresses (URLs) for users.

Service Agents (SAs) work on behalf of services. An SA responds directly to UA queries. If DAs exist, it registers with them using service registration (`SrvReg`) messages. SLP supports incremental service registration whereby an SA can add or change attributes of a previously registered service. Thus, a `SrvReg` message can be a fresh service registration or an update to a previous registration. An SA can also remove service listings from DAs before they expire by sending service deregistration (`SrvDeReg`) messages.

Directory Agents (DAs) serve as centralized information repositories in an SLP system. They accept SA registrations and answer UA queries. DAs can be discovered either actively or passively (Fig. 2). For passive DA discovery, UAs and SAs simply listen for the unsolicited DA advertisement (`DAAdvert`) messages sent periodically by DAs to an administratively scoped multicast address [12]. UAs and SAs can actively discover DAs by issuing a multicast DA discovery service request message (`SrvRqst` with service type "service:directory-agent"). DAs answer each DA discovery request with a unicast `DAAdvert` message.

Figure 3: Small SLP System without DAs

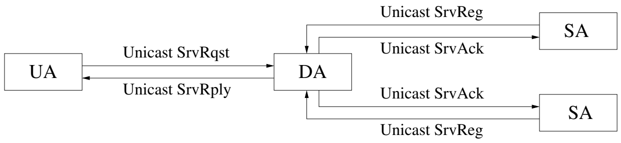

Figure 4: Mid-size SLP System with DAs

SLP achieves scalability by using DAs and service scopes. It supports service discovery in systems of different sizes and operates differently. In small SLP deployments without DAs (Fig. 3), UAs directly send `SrvRqst` messages to all SAs via multicast, and SAs respond with unicast `SrvRply` messages. Since this multicast based discovery cannot scale to more than a hundred or so services of the same type, DAs are introduced in mid-size SLP deployments (Fig. 4). SAs register services with DAs, and UAs look up service information at DAs, all using unicast. In large SLP deployments, (Fig. 5), DAs are arranged into different scopes to provide further scalability. For example, services in the Law School and Business School of Columbia University can be assigned to different scopes.

To avoid that a DA becomes a single point of failure for a scope, multiple DAs are needed for each scope. However, SLP does not define how these DAs should interact with each other. In SLP, the consistency of directory services relies upon all SAs registering their services with all DAs in their scopes. This cannot be guaranteed in large deployments, meaning DAs of the same scope may arrive at inconsistent state.

3 The Design of mSLP

mSLP improves the reliability and consistency of SLP directory services by using a fully-meshed peering DA architecture. Peer DAs exchange their data for 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 shared scopes. We first define some terminology, then we describe mSLP design considerations.

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

A *peering connection* is 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. The closing of the connection terminates the peering relationship.

A *mesh-enhanced DA* is a DA which maintains a peering connection to each of its peers and forwards messages to its

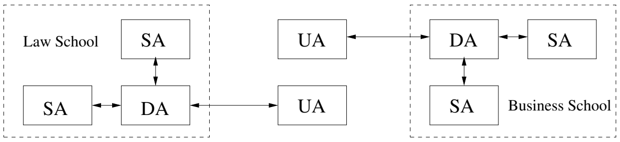

Figure 5: Large SLP System with Multiple Scoped DAs

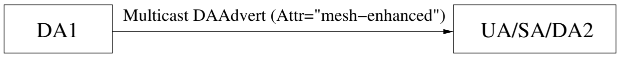

Figure 6: Mesh-enhanced DA Advertisement

peers according to the rules given in Section 5. Such a DA carries the “mesh-enhanced” attribute keyword in its `DAAdvert` messages (Fig. 6).

A *Mesh-aware SA* is an SA which understands the “mesh-enhanced” attribute in `DAAdvert` messages and uses the mesh-forwarding capability of mesh-enhanced DAs for its service registrations.

3.1 Reliability

The fully-meshed peering DA architecture avoids a single point of failure by replicating data among at least two peer DAs, automatically synchronizing data among these DAs. It also enables a DA to recover from a crash with much less effort since a rebooted DA can get the existing registration data from its peering DA set. This is done through DA coordination, without involving SAs.

The fully-meshed peering DA architecture is built on top of a set of fully-meshed peering connections. Any service registration information received by a DA can be replicated directly to all other DAs in the peering DA set. We anticipate that two to four DAs are sufficient to achieve very high reliability; larger peer sets significantly increase maintenance overhead. There is no need to have a separate DA for each scope. A DA can serve multiple scopes, and a single peering connection is used across all shared scopes between each pair of peer DAs.

3.2 Scalability

SLPv2 requires that SAs register with all existing DAs in their scopes 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, a mesh-aware SA only needs to register with one mesh-enhanced DA in the registration scope; the registration information will be propagated automatically within the meshed DA set. The overall system scalability can be improved by using mesh-enhanced DAs and simplified SAs.

3.3 Compatibility

mSLP is backward compatible with with SLPv2. It only defines a new attribute and a new SLP extension. The new attribute called “mesh-enhanced” is used in `DAAdvert` messages to identify mesh-enhanced DAs. The new SLP extension called “mesh-enhancement extension” is used by `DAAdvert`, `SrvReg` and `SrvDeReg` messages to specify the operations of mesh-enhanced DAs. This extension shown in Fig. 7 has a

Mesh−enhancement Extension ID		Next Extension Offset (NEO)
NEO, contd.	Action−ID	DA List

Figure 7: Mesh-enhancement Extension

five-byte extension header, a one-byte field denoted as “Action-ID”, and an optional DA list. Currently, six actions are defined. The *mesh forwarding request* requests that the receiving DA forward the message to all DAs (both mesh-enhanced and non-mesh-enhanced) in the registration scope. The *null operation* is used by the sending DA to turn off the *mesh forwarding request* in a message, informing that the receiving DA should ignore the mesh-enhancement extension in the message. The *data copy request* asks that the receiving DA send all the service registration data in shared scopes to the requesting DA. The *peer connection indication* informs the receiving DA that the connection from which the message is received is a peering connection. The *peer DAs indication* carries a list of URLs indicating the DAs that the receiving DA should peer with. Finally, the *peer connection keepalive* indicates that this peering connection is alive and should not be closed.

By properly using the mesh-enhancement extension and properly handling it, peer DAs can interact with each other to provide the desired functionality. This DA interaction is added as an enhancement to a DA without affecting its original functions. Moreover, the changes are mostly transparent to UAs and SAs. UAs can be kept unchanged. SAs can simplify their service registrations by using mesh-forwarding.

mSLP supports incremental deployment of mesh-enhanced DAs, e.g., they can be deployed one scope at a time. However, since a mesh-aware SA still needs to take care of newly found and rebooted non-mesh-enhanced DAs as these DAs cannot get existing data from other DAs, uniform deployment of mesh-enhanced DAs is much more desirable than partial deployment.

4 Peer Relationship Management

In mSLP, a mesh-enhanced DA maintains a peer list. Each entry in this peer list includes the peer URL, a list of shared scopes, a boot timestamp for the peer to distinguish it from its rebooted instance, a reference to the peering connection, and a mesh flag indicating whether the peer is mesh-enhanced or not. A mesh-enhanced DA adds an entry to its peer list when it discovers a new peer, removes an entry from the peer list when it finds that the corresponding peer is down, and updates an entry when it detects that the corresponding peer has rebooted.

A peer relationship has three stages: setting up, maintaining, and tearing down. mSLP considers the situation where mesh-enhanced DAs, non-mesh-enhanced DAs, mesh-aware SAs and non-mesh-aware SAs coexist. It works even if multicast is not supported or a DA’s multicast `DAAdvert` messages cannot reach all of its peer DAs.

4.1 Setting Up a Peer Relationship

When a mesh-enhanced DA learns about a new peer (either mesh-enhanced or non-mesh-enhanced), it creates a peering

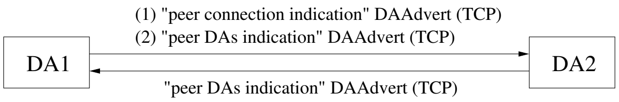

Figure 8: Peering Connection and Peer DAs Indication

connection to that peer if such a connection does not exist. The mesh-enhanced DA uses this peering connection to forward messages to the peer. The peer, if mesh-enhanced, also uses the connection to forward messages in the opposite direction. Therefore, a peering connection can be set up between two mesh-enhanced DAs or between a mesh-enhanced DA and a non-mesh-enhanced DA. In the latter case, messages are only forwarded from the mesh-enhanced DA to the non-mesh-enhanced DA.

For a non-mesh-enhanced peer, the mesh-enhanced DA just sets up a peering connection with it and forwards messages to it. But for a mesh-enhanced peer, the mesh-enhanced DA needs additional procedures to interact with it by using *peer connection indication*, *peer DAs indication* and *data copy request* DAAdvert messages[1].

After a peering connection is established, the DA who initiated the connection sends the following two messages through the connection (Fig. 8): a *peer connection indication* DAAdvert message marking the connection as the peering connection and a *peer DAs indication* DAAdvert message carrying a list of DAs that the receiving DA should peer with. This DA list is constructed based on the sending DA's peer information and the receiving DA's service scopes. More precisely, this list includes those DAs in the sending DA's peer list that share some scopes with the receiving DA.

Upon receiving a *peer connection indication* DAAdvert message, a mesh-enhanced DA should use the TCP connection from which the message is received as the peering connection to the sending DA instead of establishing another one[2], and reply with a *peer DAs indication* DAAdvert message (Fig. 8). By exchanging the peer information through *peer DAs indication* DAAdvert messages, the mesh-enhanced DAs can learn about other DAs in shared scopes even if multicast is not supported or a DA's multicast DAAdvert messages cannot reach all of its peer DAs. Initial peer relationships can be configured by hand or through DHCP [4].

In Fig. 6 and Fig. 8, assume DA2 is a peer of mesh-enhanced DA1, the protocol sequence can be summarized as follows: (a) DA1 discovers DA2 and creates a peering connection to it; (b) DA1 sends a *peer connection indication* and a *peer DAs indication* DAAdvert message to DA2; (c) DA2 replies a *peer DAs indication* DAAdvert message to DA1. If DA2 is mesh-enhanced, all (a) (b) (c) should be performed, otherwise, only (a) happens.

Upon receiving a *peer DAs indication* DAAdvert message, a mesh-enhanced DA checks the DA list in the message. If any URL in the list is not in its peer list, the mesh-enhanced DA unicasts an active DA discovery service request to the DA corresponding to the URL, to obtain a DAAdvert message from it. In Fig. 8, if DA1 received the URL for DA3 from DA2 in the *peer DAs indication* DAAdvert message, and DA3 is not in DA1's peer list, DA1 must then acquire a DAAdvert message from DA3 (Fig. 9). Upon receiving the DAAdvert message from DA3, DA1 can go through the whole process to set up a peer relationship with DA3.

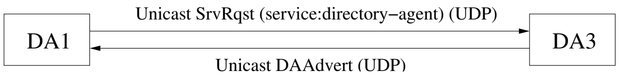

Figure 9: Obtaining DAAdvert from Peer

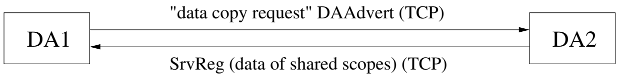

Figure 10: Copy Data from Peer

After sending the *peer DAs indication* DAAdvert message, the mesh-enhanced DA should decide whether it needs to get the data from the new peer for shared scopes. If it does, it sends a *data copy request* DAAdvert message to the peer (Fig. 10). Note that a mesh-enhanced DA does not need to download data from all of 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 one of these three DAs, but not from all of them. Each implementation can decide the criteria to select a DA from the peering DA set to download the data, for example, choosing one randomly, using the first one it found, using the nearest one or the least loaded one. On the other hand, when a mesh-enhanced DA receives a *data copy request* DAAdvert message, it sends all the data of shared scopes to the requesting DA. Each data record is sent as a SrvReg message, with a re-calculated new lifetime computed as old lifetime minus elapsed time. After exchanging data in both directions, peer DAs share the same consistent data for their common scopes.

4.2 Maintaining a Peer Relationship

To maintain a peer relationship, a mesh-enhanced DA should send a *peer connection keepalive* DAAdvert message through the peering connection (Fig. 11) if no other messages have been sent for a predefined period. There are two reasons for doing this. First, idle connections could be closed by SLPv2 DAs, closing a peering connection terminates the peer relationship, and setting up a peer relationship has overhead, so it is more efficient to keep a peering connection alive between two peer DAs than to establish it on demand. Moreover, keepalive messages help peer DAs to stay synchronized with each other. If no message has been received from a peering connection for too long, there may be a network partition and the two peer DAs may have inconsistent data for shared scopes. In that case, they should tear down the existing peer relationship and set up a new one which enables them to exchange data and get synchronized.

[1]These messages have a mesh-enhancement extension with the specified Action-ID; similar notations are used in the rest of this paper.

[2]There is a small possibility that a pair of peering connections might be created between the two peer DAs if they try to set up a peering connection to each other almost at the same time. That is inefficient, but it does not affect the correctness of mSLP.

Figure 11: DA Keepalive

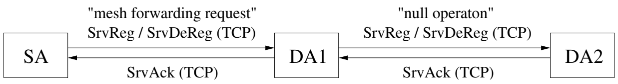

Figure 12: Forwarding Service Registrations

4.3 Tearing Down a Peer Relationship

A mesh-enhanced DA should tear down a peer relationship when it finds that the peer has closed the peering 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 shut down, or when it has not received any message from the peer for a predefined period. To tear down a peer relationship, a DA removes the peer from its peer list and closes the peering connection.

5 Message Forwarding Control

Two types of messages are forwarded by mesh-enhanced DAs: `SrvReg` and `SrvDeReg` messages from mesh-aware SAs and `DAAdvert` messages from non-mesh-enhanced peers. The message forwarding rules are as follows:

5.1 Forwarding `SrvReg/SrvDeReg` Messages

A mesh-enhanced DA forwards a `SrvReg/SrvDeReg` message when the message has a mesh-enhancement extension and the Action-ID is *mesh forwarding request*. In other words, a mesh-aware SA needs to use the mesh-enhancement extension to explicitly specify its mesh-forwarding request for messages that are intended to be forwarded by a mesh-enhanced DA. This *explicit forwarding* rule avoids unnecessary forwarding and it is fully compatible with SLPv2, where SAs need to register with all existing DAs.

A `SrvReg/SrvDeReg` message is forwarded only once by a mesh-enhanced DA to all of its peers, both mesh-enhanced and non-mesh-enhanced, in the registration scope. Since the peering DA set is in a fully connected mesh, this *one-hop forwarding* rule ensures that a message can reach all peer DAs.

Fig. 12 shows how a `SrvReg/SrvDeReg` message is forwarded. Before forwarding a message, a mesh-enhanced DA sets the Action-ID in the mesh-enhancement extension to the *null operation*. That way, a forwarded message will never be forwarded again. Since all forwarded messages are received from peering connections, this property can also be used to decide whether a message is forwarded or not. We prefer to use the *null operation* Action-ID in the mesh-enhancement extension to label forwarded messages since letting a message itself carry a label for properly handling is more robust to avoid unnecessary forwarding, and it is more efficient to check the message itself than to look up a peering connection table to make the decision.

As a DA always replies with a `SrvAck` message when it receives a `SrvReg/SrvDeReg` message, a mesh-enhanced DA should *handle `SrvAck`* messages from other DAs. In Fig. 12, DA1 returns a `SrvAck` message to the SA upon receiving and processing a `SrvReg/SrvDeReg` message. As DA1 also forwards the message to DA2, it should properly handle the `SrvAck` message from DA2.

5.2 Forwarding `DAAdvert` Messages

First, a `DAAdvert` message is forwarded only when it comes from a new or rebooted non-mesh-enhanced peer. Second, a mesh-enhanced DA forwards a `DAAdvert` message to all of its mesh-enhanced peers that share some scopes with the advertised DA. Third, a forwarded `DAAdvert` message should not be forwarded again. It can be identified easily since the sending DA and the advertised DA are different for a forwarded `DAAdvert` message.

Forwarding the `DAAdvert` message from a new or rebooted non-mesh-enhanced peer ensures that the DA is known to all of its mesh-enhanced peers even if multicast is not supported or its multicast `DAAdvert` messages cannot reach all of its mesh-enhanced peers. With the peering procedure described in Section 4.1 and the forwarding rules given in this section, a DA, whether mesh-enhanced or not, known to one mesh-enhanced peer can be known to all of its mesh-enhanced peers. Thus, a mesh-enhanced DA can know all of its peers and forward service registration information to them properly.

6 Example

We present an example to show how mSLP works. In the example, the mSLP system is deployed at Columbia University, with three different scopes for the services in the Law School (L-School), Business School (B-School) and Engineering School (E-School). Instead of using a separate DA for each school, mSLP uses three DAs in a fully-meshed peering architecture, where each DA serves two scopes and each school is served by two DAs (Fig. 13). An mSLP system can be in one of the three stages: normal operation, DA failure, and recovering from a failure.

In normal operation, registration information is distributed automatically. In Fig. 13, the mesh-aware SA uses the mesh-enhancement extension with the Action-ID set to the *mesh forwarding request* for its service registrations, and it registers services in B-School, E-School and L-School with DA1, DA2 and DA3, respectively. The mesh-enhanced DAs will forward service registrations to peer DAs automatically. If the UA queries service information in L-School with DA2, it will get the same data as what the SA has registered with DA3.

As long as only one of the three DAs (say, DA1) fails, this mSLP system can still function. Although the data of B-School and E-School are not available from DA1, they can be retrieved from DA3 and DA2, respectively.

When a failed DA (say, DA1) comes up again, it sets up peer relationship with the other DAs again. Since now DA1 carries a new boot timestamp, DA2 and DA3 know that it has rebooted and coordinate with it. DA1 retrieves B-School and E-School data from DA3 and DA2 respectively, then this peering DA set is back to normal. Thanks to the fully-meshed peering DA architecture, the recovery of DA1 happens automatically through

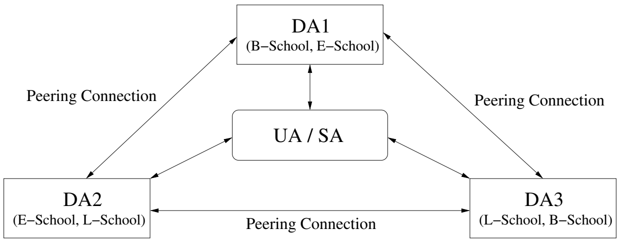

Figure 13: an mSLP System Example

DA coordination, without SA involvement.

7 Implementation

We have implemented a prototype of mSLP, which is available at *http://www.cs.columbia.edu/~zwb/project/slp*. Mesh-enhanced DA support somewhat complicates the implementation of SLPv2 DAs, as it requires peer relationship management and message forwarding control. However, 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 complexity of mSLP is about the same as SLPv2 without mesh-enhancement.

8 Related Work

We presented the specification of mSLP as an Internet-Draft [17]. The fully-meshed peer relationship is used in IBGP [16]. Server redundancy, such as in DNS [13, 14], is a basic method to provide reliability. There are a lot of research efforts on synchronizing data among replicated servers efficiently [15].

9 Conclusion

In this paper we presented mSLP, the Mesh-enhanced Service Location Protocol. mSLP enhances SLP with a fully-meshed peering DA architecture. It improves the reliability and consistency of SLP directory services. It also greatly simplify SA registrations in systems with multiple DAs. mSLP is backward compatible with SLPv2 and can be deployed incrementally.

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 consideration, assuming that clients will regularly participate in multiple scopes.

References

[1] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *Proc. ACM Symposium on Operating Systems Principles*, December 1999.

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

[3] S. Czerwinski, B. Zhao, T. Hodes, A. Joseph, and R Katz. An architecture for a secure service discovery service. In *International Conference on Mobile Computing and Networks*, August 1999.

[4] R. Droms. Dynamic host configuration protocol. Request for Comments 2131, Internet Engineering Task Force, March 1997.

[5] 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.

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

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

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

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

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

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

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

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

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

[15] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw-Hill, 2000.

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

[17] W. Zhao, H. Schulzrinne, and E. Guttman. mSLP - mesh-enhanced service location protocol. Internet Draft, Internet Engineering Task Force, June 2000. Work in progress.