A Flexible Architecture for Wide-Area Service Discovery

An-Cheng Huang   Peter Steenkiste
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

***Abstract*-Users of networked services traditionally have to know the network address of the desired service. As the network evolves from a simple infrastructure offering basic services (e.g. printers) into a sophisticated programmable infrastructure that supports a rich set of services such as virtual reality games, real-time video transcoding, and content providers, manual configuration of services becomes impractical. The Service Location Protocol (SLP) provides a dynamic configuration mechanism for the discovery of networked services. While simple SLP solutions have proven to be effective in local-area networks, solving the problem in the wide-area environment turns out to be very challenging. In this paper, we present an architecture for wide-area service discovery. Our architecture utilizes both the push and pull techniques for scalable distribution of the service information, and uses different types of Directory Agents to help the search process. Service providers and clients can find each other through a "service discovery market" which operates on our architecture.**

I. Introduction

Users of networked services traditionally have to know the network address of the desired service. As the network evolves from a simple infrastructure offering basic services (e.g. printers) into a sophisticated programmable infrastructure that supports a rich set of services such as virtual reality games, real-time video transcoding, and content providers, we expect the number of services to soar and manual configuration of services becomes impractical. The Service Location Protocol (SLP) [1] eliminates the need for a user to know the network address of the service host. The user only has to specify the type and attributes of the desired service and SLP automatically resolves the network address of the desired service for the user. However, the current Service Location Protocol is intended to function within a local-area domain, and, to fully utilize the functionality of SLP, multicast must be supported in the environment. This solution does not scale easily to wide-area networks.

An architecture for wide-area service discovery faces many challenges [2]. First it has to meet the usual set of requirements such as fast response time for client queries, security and access control, and robustness in the presence of network or server failures. Moreover, the protocol has to scale to large numbers of services and service types, it has to support look up based on unordered sets of attributes, it has to work across administrative domains that might use different technology, and it has to automatically adjust for the addition, removal, and changing of services. Another issue is that, in the SLP model, the search process does not rely on interactive input from a user. However, in a wide-area environment, it is more likely that an interactive search is desirable or even necessary in some cases. For example, for services that require billing or provide contents (e.g. movie servers), a user may need to interactively select an appropriate service.

This paper focuses on one aspect of wide-area service discovery, that is, the scalable collection of service information and handling of client queries. While this issue has been addressed in the local-area domain, it is much more challenging in the wide-area environment, and current proposed solutions only partially address the problem. Instead of trying to distribute the service information to everywhere in the Internet, we propose an architecture which utilizes both "push" and "pull" techniques for scalable distribution of service information. A "service discovery market" can then operate on this architecture so that providers and clients can find each other.

II. A Comparison: Web Searches

The problem of finding a web page is quite similar to the wide-area service discovery problem. We will look at the solutions used for web searching as a starting point for wide-area service discovery.

When a user wants to find a web page, he/she often starts with a large, general search engine (e.g. Altavista or Lycos), which contains information of all kinds of web pages on the Internet. If it cannot find any satisfactory page, or it returns too many pages, the user can turn to a specialized search engine, for example, engines specializing in cities (e.g. citysearch.com) or music pieces (e.g. audiofind.com). In order to find a suitable specialized search engine, the user can use a "search engine directory", that is, a search engine that specializes in finding search engines, for example, searchpower.com or allsearchengines.com.

The search-engine scenario is similar to the service discovery problem in several aspects. The search space includes a huge number of web pages or services, and the space is dynamic. Both the client and the desired object (web page or service) can be anywhere in Internet. In addition, we need a multi-criteria search in both cases. These similarities suggest that the the search-engine model, more specifically the different types of engines and their roles, is a good starting point for the wide-area service discovery problem.

III. A Flexible Architecture

The service discovery problem can be stated as follows. Given a service provider and a client that might be interested in using the service, where should we store the contact information for the provider so it is easy for the client to find. In the original local-area Service Location Protocol, Directory Agents (DAs) accept service registrations from the services, store the service information, and handle client queries. In a wide-area environment, using a generic directory agent that try to store information on all services clearly will not scale, given the size of the Internet. Instead, we propose to group information of related services in "Specialized Directory Agents" that cover particular subsets of services. This is similar to the use of a specialized search engine that covers only a particular subset of web pages; users who are looking for a specific page are likely to find it much more quickly if a specialized search engine is available.

How should we group service information? Intuitively, everyone may be interested in different services, and there are so many different types of services that this scheme does not seem to work. However, we believe that, just like the web search engine analogy, a "service discovery market" will begin to operate if we provide a network architecture to support it. For example, at the beginning only several large General Directory Agents exist, and people are excited about it because this is new. After a while, users find that the DAs do not always return desired service locations, and often they return too many results to be useful. Therefore, Specialized DAs will start to appear. If a large number of clients are interested in a special type of services (say, video transcoding services), sooner or later someone will create a Specialized DA for that type of services. Finally, there will be more and more General and Specialized DAs, and eventually users just cannot memorize all of them anymore. At that time, a "DA for DAs" (a Directory Agent which provides service information of lots of Directory Agents) will appear, just like those "search engines for search engines". In the remainder of this section, we describe our architecture.

A. *The Components*

The building block of our architecture is the Directory Agent (DA) of the original SLP. Based on what kind of service information a DA stores, there are different types of DAs, which are very similar to the different types of search engines in the search engine model:

- *Local DA*: The role of the Local DA is the same as defined in the original SLP [1]: it accepts local service registrations and requests within the domain. It may also cache service information for wide area services, and it is the first point of contact for local clients looking for service information.
- *General DA*: A General DA stores all kinds of service information, and it provides the information to potentially any host in the Internet. The role is similar to the general search engines in the search engine model.
- *Specialized DA*: A Specialized DA provides only a particular subset of all service information, for example, for services in a specific geographic region, for services of certain service type(s), etc.
- *DA Directory*: A DA Directory is a "DA for DAs". It stores the service information of many DAs (since we can see DAs as services).

We now describe the operation of our architecture.

B. *Server Side: Push and Pull*

There are two models for gathering service information, each with its own limitations. The original (local-area) SLP uses the push model, that is, services actively send registration messages to the DAs. It has the advantage of rapid availability of service information, but it does not scale well beyond local administrative domains, especially considering the limited support for multicast in the wide-area network. In the pull model (for example, web search engines using web crawlers), the DAs actively looking for service information in the Internet. This approach consumes little traffic and is more suitable for the unstructured nature of the Internet, but it is not dynamic enough and the information gathered may be very incomplete.

Given the diversity of services, we believe there is no one-size-fit-all model for gathering service information. Our approach uses a combination of the push and pull models to do this. In local-area networks, all services use the push model, and Local DAs will accept the service registrations of services within their domain. General and Specialized DAs cover a large number of services, so it is impossible to allow all services sending registrations. In this case, General and Specialized DAs can use web crawlers to gather service information in the Internet. Service providers can embed service information in their web pages by using a special markup language. For example, the Extensible Markup Language [3] can be utilized to define such a markup language. Then the web crawlers can parse these pages and the service information is stored in the database of those General/Specialized DAs. Of course, these wide-area DAs can allow a subset of the services they cover to send service registrations. This subset can be determined according to many factors, for example, the capacity of the DA, the network capacity, market demand/supply, and so on.

We believe such a combination of the push and pull models will provide a practical and efficient method. By allowing only a limited portion of all services to send service registrations, and limiting the frequency of service registrations, we believe it can reduce the traffic to an acceptable amount and still allow "mildly" dynamic services. It is also possible to *selectively pull* service information, for example, a General or Specialized DA can direct web crawlers to get updated information for those services whose lifetimes have expired. Note that while the use of multicast can be useful to reduce the overhead of registration traffic from service providers to DAs, this architecture does not depend on the wide-spread availability of multicast.

C. *Client Side: Use Different Directory Agents*

Here is the client-side operation (see Figure 1). For each domain, there is a Local DA and a default General DA. The Local DA will handle local service registrations and requests, just as defined in the original SLP. If the Local DA cannot resolve a request from a client, the Local DA can forward the service request to the domain's default General DA. If the default General DA still cannot provide satisfactory results, the client can try to find a suitable Specialized DA. For example, the client can use a DA Directory to lookup a Specialized DA, or the client can get the address of a Specialized DA on a web page banner. After the client determines which Specialized DA it will use, it can send the service request to the Specialized DA and get the service reply. In this architecture, the request/reply messages for all DAs (Local, General, and Specialized) are the same as the original SLP messages.

One interesting issue here is the usage of the "DA Directory". We can imagine two alternative ways of using a DA Directory. The first is to make a DA Directory behave like a web search engine. Clients (human users) can interact with such a (search engine-like) DA Directory to find desired General/Specialized DAs, for example, clients can input several keywords for the search. Another possibility is to use *agent* techniques to automate the process of searching for General/Specialized DAs, that is, clients are represented by software agents which negotiate with a DA Directory (which is also an agent) automatically to find the desired DAs. For example, LARKS [4] defines an agent capability description language which can be used to describe the capabilities of a provider and the preferences of a client, and also specifies the matchmaking process of determining whether an advertisement matches a request. Another study discusses the roles of middle-agents in different architectures and reports some initial performance evaluations [5]. Although the number of service providers may be very huge, the number of General and Specialized DAs should be much smaller. Therefore, using agent techniques to match client requests to appropriate General/Specialized DAs (instead of matching client requests to the service providers directly) may actually be manageable.

IV. Related Work

The Service Location Protocol (SLP) [1] provides a feasible solution for discovery of local-area service locations. To fully utilize the functionality of SLP, supports for DHCP [6] and/or multicast are needed, and it does not scale to wide-area networks.

Several research groups have proposed different approaches for the problem of wide-area service discovery. The service-broker proposal [7] extends the SLP with two components, Brokering Agent and Advertising Agent, to allow service registration and discovery across wide-area networks. It assumes wide-area multicast support is established and tries to distribute service information to everywhere in the Internet. The DNS-based solution exploits the current DNS system [8, 9]. Service information is stored in the DNS hierarchy, and clients can find service information in a particular domain via DNS lookup. Therefore, it assumes that the domain in which the desired service resides is known a priori. The secure Service Discovery Service (SDS) [10] also mentions the scalability issue of wide-area service discovery, and it proposes two techniques: using hierarchies of SDS servers and aggregation of service information. However, the attributes for establishing the hierarchies are hard to define in an environment as diversified as the Internet, and the (lossy) aggregation of service information may limit the range of possible service queries. There is also an approach exploiting web crawlers to find service information on the *web* [11]. The basic idea is that service information is provided on web pages, and an enterprise network will have a special entity, Wide Area Directory Agent (WADA), which uses web crawlers to gather the service information. However, a single WADA cannot cover a significant portion of all services in the Internet, and the service information may not be updated quickly enough in a pull-based model.

V. Summary

In this paper, we present an architecture for wide-area service discovery. We do not try to distribute the service information to everywhere in the Internet, and we do not rely on any a priori knowledge to establish a hierarchy. Instead, we see the relationship between clients and Directory Agents as consumers and producers in a "service discovery market". Therefore, our architecture provides a platform for them to match their demands and supplies in the service discovery market. Our approach uses a combination of the push and pull models to gather service information in the Internet more efficiently. Using different types of Directory Agents greatly helps the distribution of service information and makes the search process faster.

References

[1] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service location protocol, version 2. RFC 2608, IETF, June 1999.

[2] J. Rosenberg, E. Guttman, R. Moats, and H. Schulzrinne. WASRV architectural principles. Internet Draft, work in progress, IETF, February 1998.
http://www.bell-labs.com/mailing-lists/wasrv/.

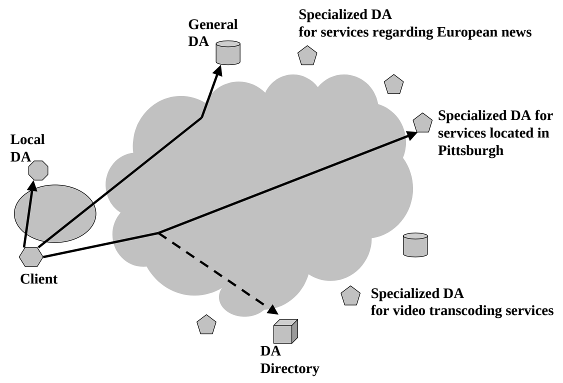
General DA
Specialized DA for services regarding European news
Specialized DA for services located in Pittsburgh
Local DA
Client
Specialized DA for video transcoding services
DA Directory

Figure 1: Client-side operation

[3] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible markup language (XML) 1.0. Recommendation 10-Feb-98, W3C, February 1998.

[4] K. Sycara, J. Lu, M. Klusch, and S. Widoff. Matchmaking among heterogeneous agents on the internet. In *Proceedings AAAI Spring Symposium on Intelligent Agents in Cyberspace*, 1999.

[5] K. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In *Proceeding of IJCAI-97*, 1997.

[6] R. Droms. Dynamic host configuration protocol. RFC 2131, IETF, March 1997.

[7] J. Rosenberg, H. Schulzrinne, and B. Suter. Wide area network service location. Internet Draft, work in progress, IETF, November 1997. http://www.bell-labs.com/mailing-lists/wasrv/.

[8] R. Moats and M. Hamilton. Advertising services. Internet Draft, work in progress, IETF, October 1997. http://www.bell-labs.com/mailing-lists/wasrv/.

[9] R. Moats, M. Hamilton, and P. Leach. Finding stuff. Internet Draft, work in progress, IETF, October 1997. http://www.bell-labs.com/mailing-lists/wasrv/.

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

[11] C. Perkins. Wide area service location protocol. Work in progress, March 1998. http://www.bell-labs.com/mailing-lists/wasrv/.