A Framework for Semantic Service Discovery

Xiaowei Yang YXW@LCS.MIT.EDU
MIT Laboratory of Computer Science, 200 Technology Square, Cambridge, MA 02142,, USA

1. Introduction

Locating a network service or a device on demand is a challenging task for enabling mobile and pervasive computing. There exists a variety of network protocols and architectures that enable an application to discover a network service with little manual configuration. Most existing work supports an attribute-based discovery as well as a simple name lookup to locate a service. Usually there are only a set of primitive attribute types, such as string and integer, to characterize a service. Thus, the service discovery process is primarily done by type matching, string comparison, or integer comparison. When there are complex attribute types, such as in Jini (Edwards, 2000), the service discovery is done by exactly matching attribute values in a query with those in a registration. For representing real world objects such as network services, it is necessary to have more complex data structures to capture richer semantics. Moreover, a user may not be able to specify the exact values of interested attributes. Thus, approximate matchings are desirable. To the best of our knowledge, no existing work allows both a service description with arbitrary complex attribute types and a set of meaningful comparison operations based on the semantics of those attributes. For example, if a service has an attribute that specifies its 3-dimension GPS location, it is difficult for a client to pose a query with the exact value of the attribute. A client usually wants to find a service that is close to it. The notation of physical proximity is best captured by comparing the distance between two locations, instead of syntactically comparing two location expressions, or comparing the values independently at each dimension.

In this paper, we articulate the conceptual issues involved in service discovery protocols. Structured service representations are not sufficient to define data semantics. The semantics of data is conveyed by how data are interpreted. We also present a framework for discovering a network service based on the semantics of its representation. The framework can also be applied to discovering general distributed objects.

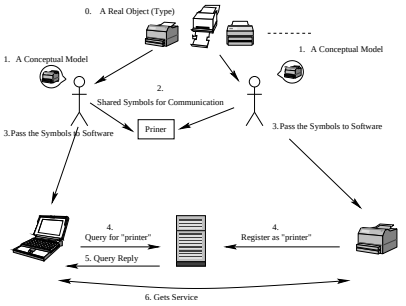

Figure 1. How Ontologies are Shared Between Clients and Services

2. The Design Framework

As in most existing service discovery work, we assume there are three components in the system, which are a service agent, a client agent and a directory agent. A service agent registers its description with a directory agent and a client agent requests a service by submitting a query to the directory agent. The unique points in our framework lie in the way services are described, queries are formatted, and queries and descriptions are matched.

2.1 Ontological Level Issues

In the mobile and pervasive computing environment, there are two problems needed to be solved. First, how does a client discover a service? Second, how does a client use a service? For a client to be able to use a service, it expects to get a reply pointing to the "right" service after it poses a query for that service. For example, the client who issues a query for "printer" should get a reply that points to a printer service. This requires that both the service and the client agree on the meaning of "printer". As computers can do nothing but symbol computing, this agreement is actually one between humans who create the query and those who create the service description. Figure 1 shows what happens in this process. First, a human forms a conceptual model about a real world object or an object type, which is a set of objects that share common features. Second, to enable communication, shared symbols are created among humans to represent the conceptual model. We call this set of shared symbols with agreed meanings a shared ontology. The ontology reflects the shared conceptual model of the

service, which includes what a service is capable of doing (e.g. the functional interface of the service), the terms in which the service is described (e.g., the data types for describing the service) and the meanings of the terms (e.g., what they stand for and what operations are allowed on them). This shared ontology is passed to software through the efforts of programmers or software users. Therefore, the software produced will show behaviors consistent with human's conceptual model. For the service discovery to work, we require that in our framework, the client of a service and the service share a common ontology on the service representation, which is ultimately shared between humans who create the service description and who create the query.

2.2 The Logical Structure of a Service Representation

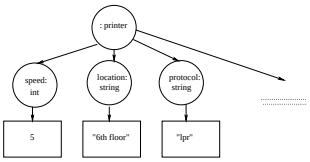

Figure 2. An Example of a Service Representation of Primitive Types

In most of the existing service discovery architectures, features of a service are represented by a set of attribute-value pairs. The logical structure of the representation can be viewed as a one-level tree, as shown in Figure 2. This model is limited because it does not have enough complex data structures to capture arbitrary service models. In our framework, a service is modeled by an arbitrary-depth tree structure. Figure 3 shows an example. For a complex type such as *GPSLocation*, a simple floating point comparison at each dimension may not make sense. A natural way of using this information is to match it with a service within some distance from the client. Thus, a complex type such as the *GPSLocation* may require a customized matching operation that captures the semantics of the type.

2.3 Semantic Aware Query-Representation Matching

A directory agent is expected to cache a variety of service representations. Thus it is difficult for the directory agent to know the semantics of all data types for heterogeneous service representations. Moreover, it is impossible for the directory agent to predicate what data types will be used in the future. However, as the functionality of a directory agent is to match a query against service representations, unlike clients and services, the directory agent does not have to know the full semantics of the representations. To enable semantic service discovery, a directory agent need to understand how to match two values given any data type, which may include: 1. whether two values are equivalent (the equivalence relation); 2. whether two values are approximate equivalent according to a user's criterion (the approximately equivalence relation); 3. whether one value is "less" than another value (the partial order relation). For example, a directory agent does not have to know the meanings of "60 minutes" and "an hour", but it should know they are semantic equivalent, despite of the different syntaxes.

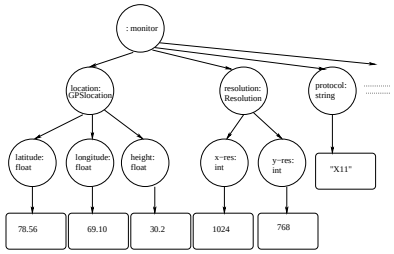

Figure 3. An Example of a Service Representation of Complex Types

To achieve the functionality of semantic service discovery, we introduce mobile code in our framework. A service representation also specifies the operations for matching or comparing new data types used in its representation. The operations are specified by code. On the service registration, the directory agent uploads the code associated with new data types. When processing a query that contains the new data types, the directory agent calls the code to compare the values in the query and the values in a service registration. Therefore, the directory agent is able to support semantic service discovery for arbitrary data types, including those unknown types when the directory agent software is built.

3. Implementation

In our implementation, we chose XML as the service description language. Despite its verbosity, XML is a well-defined language and easy to use. We choose Java bytecode as the mobile code, for its wide availability and flexible security model. The prototype is still under development.

4. Conclusion

In this paper, we described a framework to enable semantic service discovery in the mobile and pervasive computing environment. For more details, please see the extended version of this paper (Yang, 2001).

References

Edwards, W. K. (2000). *Core Jini*. Prentice Hall.

Yang, X. (2001). A Framework for Semantic Service Discovery. http://ana.lcs.mit.edu/ yxw/SSD/service.ps.