Ontology-based Service Discovery in P2P Networks

Daniel Elenius, Magnus Ingmarsson
Department of Computer and Information Science
Linköping University
581 83 Linköping, Sweden
{daele, magin}@ida.liu.se

Abstract

The ubiquitous computing *vision is to make knowledge and services easily available in our everyday environments. A wide range of devices, applications and services can be interconnected to provide intelligent and automatic systems that make our lives more enjoyable and our workplaces more efficient. Interaction typically is to be between peers rather than clients and servers. In this context, the JXTA peer-to-peer infrastructure, designed for interoperability, platform independence and ubiquity, is a suitable foundation to build future computer systems on.*

Peers need ways to effortlessly discover, consume and provide services, and to take advantage of new services as they become available in a dynamically changing network. However, JXTA does not currently handle this service-discovery problem. In this paper, we examine several service-discovery architectures, to see whether they can be adapted to JXTA. We conclude that none of them adequately support the flexibility and expressiveness that ubiquitous computing requires. We therefore argue that Web Ontology Language (OWL) and OWL Services (OWL-S) ontologies should be used to express detailed semantic information about services, devices and other service-discovery concepts. This kind of approach allows peers to reason about service offerings and achieve intelligent service discovery by using an inference engine. We present an experimental implementation of this ontological approach to service-discovery, called Oden (Ontology-based Discovery-Enabled Network).

1 Introduction

In the ubiquitous computing [1] (ubicomp) vision of the future, workplaces, homes and public environments will contain a wide range of networked devices intended to make workplaces more efficient, to increase quality of life, and empower users by providing information and services in an effortless way. Devices, and the applications and services running on them, will be highly interconnected, but usually cannot depend on pre-configured servers or static network addresses. Instead, these environments will typically be highly dynamic. Mobile networked entities appear and disappear, are reconfigured on-the-fly, and must adapt to an ever-changing network.

The peer-to-peer (P2P) paradigm of computing provides a viable approach to these issues. P2P does away with many assumptions in typical client–server systems, such as continuity of service access, and static configuration of servers, routers and naming services. Consequently, a P2P approach is well suited to ubiquitous-computing and ad-hoc network systems.

Enabling devices to work in concert over P2P networks, and allowing users to interact with networked services seamlessly, poses new technical challenges. Users should not need to configure services manually or have detailed knowledge of technical details, and interaction between peers should not depend on which operating systems or programming languages peers use. Solving this service-discovery problem will become even more important as more numerous and complex devices and services become embedded in our everyday environments.

There are several P2P systems available, for example Gnutella and Napster. However, most of these systems are intended for one specific application, such as file sharing. Therefore, our current research on ubiquitous computing uses the P2P infrastructure JXTA. This is a general-purpose infrastructure designed for interoperability, platform independence and ubiquity.

While providing for some of the needs of future computing systems, JXTA does not currently provide an adequate solution to the service-discovery problem. JXTA has a basic advertisement/search mechanism, described in the next section, but this is not sufficient for the general case. JXTA needs a flexible service-discovery system, not just for ubicomp applications, but also for any application with dynamic interconnected services.

After a short introduction to JXTA (Section 2), we examine existing service-discovery systems (Section 3), to see whether any of these can fill this need for JXTA. Section 4 discusses problems with the existing approaches, and Section 5 suggests a solution—integrating service discovery based on *ontologies* with JXTA—and our experimental implementation Oden (Ontology-based Discovery-Enabled Network). Section 6 relates our work to some other research projects, and Section 7 provides a discussion of our results. Finally, we present our summary and conclusions, in Section 8.

2 JXTA

The goals of the peer-to-peer infrastructure JXTA [2] (short for "juxtapose") are *interoperability*, *platform independence*, and *ubiquity*. Through the JXTA protocols, peers using different transport protocols and hardware platforms, and programmed in different languages, can interact with each other. Currently, JXTA has support for TCP/IP and HTTP networks. The reference implementation of JXTA is written in Java. There are also versions for J2ME [3] (Java 2 Micro Edition), a very light-weight C language implementation suitable for embedded devices [4], and several other implementations in development.

In JXTA, peers are organised into *peergroups*. A peergroup can be used to represent the *context* of peers' interactions—types of service, current state, location, etc.—and representing context is of the utmost importance in ubiquitous computing [5].

Any peer that wishes to make a service available on a JXTA network needs to create an *advertisement* of the service. An advertisement is a small piece of XML data that announces the existence, and some properties of, a peer, a peergroup, or a pipe. The peer then needs to publish the advertisement. Publishing an advertisement allows other peers in the same peergroup to find it, using a standardized search mechanism, until the expiration time of the advertisement has passed. At that time, the service provider should publish a new advertisement, if it still wishes to provide the service. When a peer finds an advertisement, it usually puts it in its local cache. Other peers can then retrieve it from there as well as retrieving the advertisement from the actual service provider. This mechanism provides additional redundancy and scalability of JXTA networks.

Advertisements in JXTA have a string Name field. When a peer wants to locate other peers, it can use these names to guide its search. If we adopt the convention of using colon-separated strings, such as "Device:Printer:Printer1" or "Service:E-shop:UsedComputersInc" for the advertisement names, they can be used to indicate what type of service the peer provides, in a hierarchical fashion. The discovery mechanism in JXTA also allows '*' as a wild card, so for example a peer could search for "Device:Printer:*" in order to find all peers with names starting with "Device:Printer:".

However, the name of a device or service often does not provide sufficient information. Until there is a globally agreed-on hierarchy of devices, the search string above would perhaps result in a few matching printing services, but other printers could be named for example "Hardware:OutputDevice:Printer," or something entirely different, and these services would not be discovered. A user looking for a used PC may be happy to find "UsedComputersInc," not knowing that it only sells Macs. These examples hint that more than just simple name matching is needed for service-discovery.

3 Current Service-Discovery Technologies

To implement service discovery for JXTA, we first examine existing service-discovery protocols, to see if any of these can provide a solution, or give valuable insights into the service-discovery problem. Several surveys of service discovery have been published [6, 7]. Our examination will only present some of the most important characteristics of the protocols. We will discuss shortcomings of these protocols, especially related to service-discovery for JXTA and ubiquitous-computing environments, in Section 4.

3.1 Bluetooth SDP

Bluetooth[1] is a short range, low power wireless technology for cable replacement. In other words, it is intended to rid users of cable clutter and the need for frequent disconnection and reconnection. Bluetooth was invented by Ericsson but is now controlled by the Bluetooth special interest group (SIG), today comprised of a large number of companies including 3com, NOKIA, IBM, Intel, Microsoft and Motorola.

Bluetooth defines its own protocol stack, including a service-discovery protocol, SDP. This is based on unique identification numbers (UUIDs), with several predefined services such as Headset, Printing, Fax, etc. After a device has been discovered, a Protocol Descriptor List in the SDP service description is consulted to find out which protocols can be used to initiate contact with the device. An interesting project [8] has shown that the Bluetooth SDP can also be extended with semantic descriptions written in RDF or DAML+OIL to support more expressive information about devices.

While not truly peer-to-peer in the sense of equality among the peers (one is master the other is slave), Bluetooth does provide the ability to switch between these roles. The Bluetooth specification defines so-called 'piconets' which are groups of up to 255 devices. Only eight of these devices

[1] `http://www.bluetooth.com/`

can be active at any given time. The rest have to be in 'park mode'. Several piconets can be connected into 'scatternets' using a device acting as a 'bridge-slave'. A device can be master in one piconet and slave in another.

3.2 Universal Plug and Play (UPnP)

Poised to make home networking easier, the UPnP[2] standard, geared towards both software services and physical devices, is developed by a forum led by Microsoft.

UPnP builds on existing protocols and standards: DHCP and AutoIP for addressing; IP, UDP, TCP and HTTP for communication; and SOAP for remote invocation. UPnP also defines a couple of additional XML-based protocols to support service-discovery: SSDP (Simple Service Discovery Protocol) and GENA (Generic Event Notification Protocol). We will here focus on SSDP as it has the most relevance for our purposes.

When entering a network, devices broadcast a short advertisement containing its type, unique identifier, and an URL to more information. These advertisements are stored by *control points*. Searching for a device is done by broadcasting a request for the desired type of device. This request is intercepted by all control points, and matching advertisements are sent back to the requester. Next, the service requester retrieves *device descriptions* and *service descriptions* of the devices found, using URLs embedded in the advertisements. These descriptions are based on *templates* defined by the UPnP forum. Only a few templates have been defined however, among them Internet Gateway Device, Printer Device, and Lighting Controls. SSDP device descriptions can also have an URL to a HTML *presentation page*, which can be used to control the device or service and view information about it.

3.3 Salutation

The Salutation architecture[3] is controlled by the Salutation Consortium, ranking IBM and a large number of printer and digital camera manufacturers among its members.

The architecture is based on service brokers called *Salutation Managers* (SLMs). These play a crucial role in Salutation, mediating all interaction between devices, including actual data transfer. SLMs use *Transport Managers* (TMs) to achieve network independence and connect to remote SLMs. A device that wishes to provide a service registers it with an SLM, or provides its own SLM and makes it available on the network. Services are described by *Functional Units* (FUs). FUs such as Print, Fax Data, Address Book, etc. have been defined. Each FU defines attributes relevant to its type of service, and specific services fill in values for these attributes. To discover a service, a request is sent to an SLM, which tries to match the request with the FUs that have been registered there. An SLM can also propagate a request to other SLMs that may know of more devices. SLMs can also exchange information among themselves, creating a topology of the network.

[2]`http://www.upnp.org/`

[3]`http://www.salutation.org/`

3.4 Service Location Protocol (SLP)

In designing SLP[4], the Internet Engineering Task Force (IETF) aimed at IP-based networks, and this architecture relies heavily on TCP and UDP to determine existence, location and settings of the services offered. There are three types of agent in SLP: User agents (UAs), Service agents (SAs) and Directory agents (DAs). UAs discover locations and settings needed by the potential user of the service; SAs advertise the availability of services; and DAs act as brokers, caching information about services. The system can operate in two modes—with or without DAs. When operating without DAs, the UA will send a multicast request for services, and will receive unicast replies. When there are DAs present, SAs will attempt to register with a DA, and UAs will send all discovery requests to these brokers. Service descriptions in SLP are very basic—'Service URLs' that categorize service types.

SLP strictly deals with discovery. What happens after that is not within the range of the SLP specification.

3.5 Jini

To ensure platform independence, Sun developed Jini[5] to run on Java. This of course leads to a certain amount of platform independence but unfortunately it also creates the requirement that devices that want to use Jini have to have a running Java implementation on them. When a service wishes to make its presence known on the network it will register itself by uploading a proxy object to a *lookup service*. When searching for a service, the searcher sends out a multicast UDP request. After receiving the results for the search from the lookup services, the searcher can download the proxy objects and run them locally in order to establish contact with the desired service. Jini is one of the few service-discovery systems that rely on code mobility and serialization of (Java) objects.

Jini also sports other features such as a transaction server, but we will not describe these here.

4 Issues in Current Service Discovery

The five technologies we have discussed above are representative of current approaches to service discovery. We

[4]RFC 2165

[5]`http://www.jini.org/`

have already mentioned some limitations of these technologies. This section will give a deeper discussion of the issues.

First, we note that some of the existing protocols depend on specific network transport layers. UPnP integrates a suite of such protocols, from IP and TCP for basic communications, up to SOAP and GENA for service invocations and event notifications. SLP is also IP-only, but uses its own formats rather than XML for its various service-discovery messages. Jini uses UDP. Bluetooth SDP runs only on Bluetooth networks, of course. Other protocols have abstractions for the transport layer that allow different infrastructures to be used. Salutation, with its Transport Managers, is a case in point.

A service-discovery architecture for JXTA should be independent of network infrastructure. Using something like UPnP or SLP would compromise this platform independence, which we view as important in a ubicomp setting. To ensure this, service-discovery in JXTA should be placed at a higher level of abstraction than JXTA's own primitives, such as peers and peergroups. This is not possible if the service-discovery protocol is tightly integrated into the underlying network infrastructure. Of the technologies we have examined, only Salutation meets this first goal without major modifications. A JXTA Transport Manager can be written to enable the Salutation architecture to run on JXTA networks.

A second issue is the use of brokers, and the suitability of the service-discovery technologies for peer-to-peer networks. Bluetooth SDP does not require brokers, but connections have a master-slave setup, whereas peers in P2P networks are usually considered as fundamentally equal. UPnP makes use of control points, but these are usually directly connected to the device itself, rather than acting as proxies or brokers for many devices. Salutation's SLMs can be both local and remote, so both direct and mediated discovery is supported. SLP and Jini require the use of central repositories of service descriptions or interfaces.

Using brokers, or proxies, to mediate service-discovery requests, while done by for example JXTA Search [9], is not suitable for ad-hoc and ubiquitous-computing networks. A requirement to use proxies or mediator services may mean that many peers lose the service-discovery capability if the central mediator goes down, or if the network connectivity to the mediator is faulty. It could also mean that users have to configure which mediator service to use manually, or at least have knowledge of which mediator services there are. Furthermore, in an environment where devices are highly mobile, and the state of devices is rapidly changing, having a mediator could mean excess work, as the mediator service must be updated to reflect all such changes. It is fundamental to P2P systems that all peers should be able to directly connect to each other, and this is one of the reasons why we feel that a P2P approach is suitable for our ubiquitous computing research. Of the examined technologies, Bluetooth, UPnP and Salutation meet this goal of peer-to-peer discovery.

Table 1: Summary of current service-discovery technologies.

	Network independence	P2P discovery	Expressiveness
Bluetooth	No	Yes	No
UPnP	No	Yes	No
Salutation	Yes	Yes	No
SLP	No	No	No
Jini	No	No	No

A third issue to discuss is the expressiveness and flexibility of device descriptions and discovery requests. Bluetooth SDP uses globally reserved unique identifiers for predefined service types. UPnP uses its own XML format, which allows more information about devices to be described, but these descriptions must be based on one of the (so far very few) existing templates agreed upon by the UPnP forum. Salutation has a similar approach with its Functional Units. SLP only allows a simple categorization of services based on its Service URLs. Jini describes services only from a programming perspective; its service descriptions are the Java interfaces to services.

While using appropriate network technologies can solve the first two issues we have mentioned, those of network independence, and direct peer-to-peer discovery, this third issue of expressiveness and flexibility of device descriptions and discovery is more difficult, and requires some novel thinking. None of the service-discovery architectures we have examined scale well in this regard; without expressive device descriptions, service-seeking peers cannot reason about devices in an intelligent way.

Our survey, summarized in Table 1, has not given us a ready-to-use service-discovery architecture for JXTA. It has, however, resulted in three requirements for such an architecture:

1. Network independence. By placing the service-discovery mechanism at a higher level of abstraction than peers, peergroups, etc., we can take advantage of the network independence already provided by JXTA.

2. Peer-to-peer discovery. We have argued that service-discovery should be unmediated in order to be suitable for ubiquitous computing and ad-hoc networks.

3. Expressiveness. Service descriptions must be expressive and flexible, scaling to future device types and providing support for reasoning about services.

The next section describes an approach to solving the service-discovery problem and meeting the above requirements, by integrating *ontologies* with JXTA.

5 Ontology to the Rescue

We propose to enhance JXTA with semantic models of services using OWL [10] (Ontology Web Language). OWL has its roots in the semantic web and Description Logic [11] fields, and can be used to create ontologies to represent any sort of knowledge. An ontology in computer science is usually defined as *an explicit specification of a conceptualization of a domain*. For our purposes, we can use ontologies to describe a shared conceptualization of the domain of services, devices and other concepts that could influence the service-discovery process, such as different kinds of context [5]. Using ontologies will enable service-seeking peers to reason about available services and devices, and make intelligent and informed decisions regarding which services to use, and how.

OWL is the emerging standard representation language for ontologies, and as such has good tool support. Also, the OWL Services[6] (OWL-S) ontology is written in OWL, providing further motivation for using this way of representing ontologies.

The OWL-S ontology for semantic web services provides a starting-point for our work, by providing a set of concepts for modeling some aspects of services. For example, it lets us model inputs and outputs, preconditions and effects, and the relations that different processes have to each other. However, we feel that a more comprehensive ontology is needed for service-discovery in ubiquitous computing in general, since OWL-S does not include concepts for *device capabilities* and *context*. It must also be demonstrated how such an ontology can be integrated with the JXTA peer-to-peer network.

Our experimental research platform Oden uses ontologies expressed using OWL, and expanding on OWL-S concepts, for semantic modeling of services and devices. Oden integrates these ontological descriptions with JXTA in a way that preserves the fundamental P2P characteristics that we believe are important, *viz.* independence of network infrastructure, and direct P2P discovery. While JXTA handles the basic networking duties, service ontologies enable reasoning and evaluation of services. We thus have two layers in Oden: the low-level communications infrastructure, and the high-level ontologies and reasoning. These two layers are mostly independent of each other. We could rewrite the ontologies without changing how they are integrated with JXTA, and we could replace JXTA and still use the same ontologies. This is an important feature, as both P2P systems and ontologies are rapidly evolving fields.

[6]`http://www.daml.org/services/owl-s/1.0/`

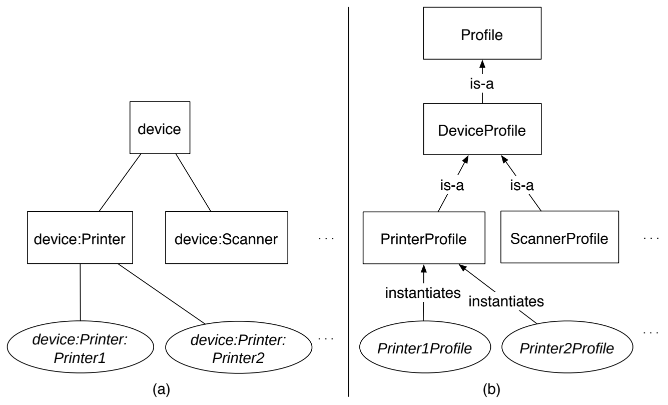

Figure 1: The one-to-one mapping of (a) the JXTA Name hierarchy and (b) the OWL-S Profile hierarchy.

The following subsections will discuss discovery-enabling ontologies, how Oden integrates such ontologies with JXTA, and how the service-discovery procedure in Oden takes place in more detail.

5.1 Ontologies and Reasoning

Service-seeking peers must be able to evaluate service descriptions, compare their pros and cons and deduce facts that might be needed to make a decision on which service to use, and how. To evaluate OWL ontologies, peers need an inference engine. Several inference engines are available. The peers we have implemented currently use JTP [12] (Java Theorem Prover), as it offered a good combination of usability, performance and documentation; but we could substitute this for other inference engines if necessary. In fact, all evaluation in Oden takes place at the service consumer, so different peers could use different engines and altogether different methods of evaluating services.

The service descriptions in Oden are currently based on a hierarchy of OWL classes, shown in Figure 1, based on the OWL-S *Profile* class. OWL-S encourages subclassing the Profile class to present properties specific to particular types of services to agents that wish to evaluate the service. We introduce a class called DeviceProfile, with properties common to all types of devices. For example, each DeviceProfile can have several DeviceConfigurations, each of which can, in turn, hold several ServiceParameters. This lets us express different *combinations* of parameters that the device supports. For example, a printer may have a high-resolution black-and-white configuration, and a low-resolution colour configuration.

Further subclasses of DeviceProfile define properties specific to different types of devices. As an example, we have created a PrinterProfile class, with properties for paper size, resolution and so on. New subclasses can be added for any type of device, and relations between device types can be expressed using subclassing, and other OWL constructs, such as class disjointness, unions, etc.

We plan to further expand our ontology with more concepts that can be useful for service discovery, for example different types of context; location, movement, social context, etc. These discovery-enabling technologies are a promising direction for future research. The ontologies we have implemented so far are not comprehensive, but the Oden architecture shows how such ontologies will fit into an integrated architecture, and thus lays a foundation that this research can build on.

5.2 Communications Infrastructure

In Oden, each peer represents a device or service. Rather than a classic client–server approach, Oden uses peers that can both consume and provide services, and does not make assumptions on pre-configured servers. JXTA was designed specifically for ubiquitous and ad-hoc networks, and can handle these issues using abstractions such as peers, peer-groups and advertisements.

Oden builds on the basic discovery mechanism in JXTA, i.e., advertisements and searching for service names, and augments it with a layer of semantic information. We define two important additions to service advertisements that peers providing services using Oden must implement: 1) A WSDL[7] interface description of the service, and 2) Pointers to OWL files that describe the service (see Figure 2). Furthermore, service names must follow a common hierarchy. This hierarchy maps one-to-one to the OWL-S Profile hierarchy described in Section 5.1, as shown in Figure 1.

WSDL is a language for defining programming interfaces of objects and methods, in a way that is independent of the programming language the objects and methods are written in. WSDL uses XML descriptions to achieve this language independence, and is being standardized by the W3C. OWL-S uses WSDL interface descriptions to provide a concrete 'grounding' to its abstract descriptions of methods, so our use of WSDL follows naturally from our choice to use OWL-S.

Peers that wish to evaluate a service need to retrieve the OWL files that describe the service. This remote communication is also kept language-independent by using SOAP[8] (Simple Object Access Protocol). SOAP is also an XML format, and an associated protocol for remote invocation also standardized by W3C.

To use SOAP in the JXTA infrastructure, Oden uses a JXTA-SOAP bridge that embeds the SOAP messages in JXTA messages. This bridge builds on existing work, and has been extended to better interact with the other parts of Oden. Integrating WSDL and SOAP with JXTA was straightforward, as JXTA uses open XML messages and advertisements, and is not commited to any particular standard for remote communication.

[7] Web Services Description Language (WSDL) 1.1, `http://www.w3.org/TR/wsdl`

[8] SOAP Version 1.2 Part 1: Messaging Framework, `http://www.w3.org/TR/soap12-part1/`

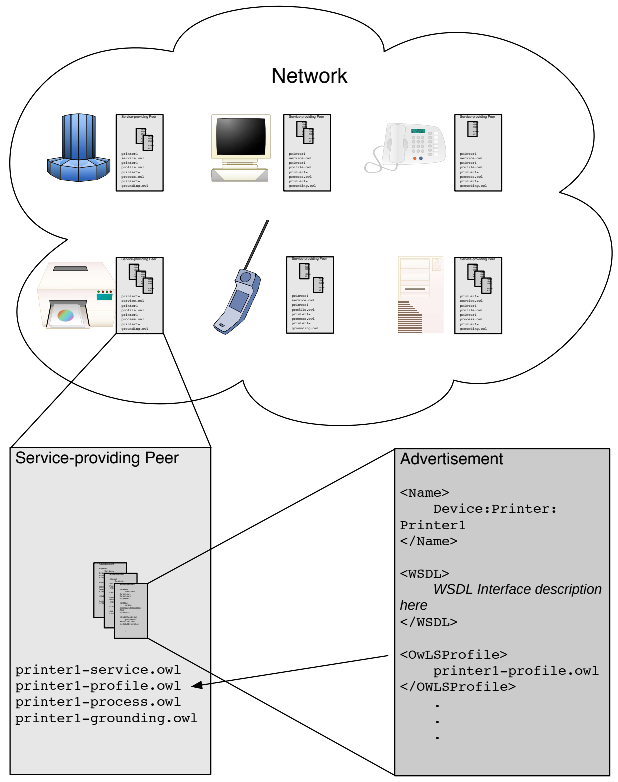

Figure 2: A service advertisement in Oden. The Name places the device or service in a hierarchy; the WSDL field provides a programming interface to the service; and the OWLS tags point to detailed semantic descriptions, retreivable from the service-providing peer.

5.3 Service Discovery Procedure

The procedure, then, for a peer to find, evaluate and use a service in Oden is as follows:

1. Retrieve advertisements from other peers that are of the correct type (e.g. "Device:Printer:*").

2. Retrieve the OWL files, indicated by the advertisements, that describe the services provided by these peers. Calling a getFile SOAP method that all service-providing peers in Oden must implement retrieves the files.

3. Load the OWL descriptions into an inference engine.

Evaluate the data and decide which service(s) to use, if any.

4. Extract the WSDL interface description of the service from the service's advertisement, and call the SOAP methods described there in order to invoke the service.

6 Related Work

Developing for a ubiquitous P2P environment requires new ways of approaching the development process. The authors of [13] present three dimensions that they claim will create a new paradigm of *service-driven* architectures: Information vs. activity, centralized vs. ad-hoc networks and implicit vs. explicit semantic descriptions. The authors argue that ad-hoc configured, activity-oriented services with explicit semantics will offer new opportunities. On the downside, going from centralized to ad-hoc configurations and from implicit to explicit semantic descriptions, usually generates a higher degree of complexity in the system. However, this kind of migration is probably necessary if we are to support future dynamic and ubiquitous environments. Using JXTA to solve the ad-hoc issue as we have done in Oden, is a good starting point.

Several attempts have been made to marry the peer-to-peer and semantic web worlds. Edutella [14] is a metadata query infrastructure based on RDF[9], and layered on top of JXTA. Oden is also based on JXTA, so it is interesting to contrast Oden with Edutella. The similarities do not run very deep, however. Edutella uses an approach similar to JXTA Search (already mentioned, [9]), that uses 'super-peers' (called *hubs* in Edutella) to register query-answering capabilities and propagate queries. Querying takes place using a pre-defined query language and data model, and reasoning is not done on the service-seeking agent. These approaches are, as we have argued, not appropriate for service-discovery in ubiquitous computing networks. Edutella is not intended for discovering and using *services*, but rather for finding metadata about information providers. It has no explicit SOAP/WSDL integration as Oden does. Despite these difficulties, there has been research [15] on using DAML-S[10] service descriptions with Edutella.

Still, Edutella is an impressive infrastructure for information integration in distributed systems, and its makers have shown how a multitude of different types of information resources can be aggregated and translated using its mediating 'wrapper' peers.

Other developments focus more on the low-level details of peer-to-peer networks, such as routing of messages between peers. In [16], an "ontology-based P2P Infrastructure for semantic seb services" is presented. While JXTA relies on broadcasting to distribute queries and advertisements, and rendezvous and relay peers to propagate these messages to other networks if necessary, the infrastructure presented in [16] partitions the P2P network into a so-called *hypercube*. There are specific algorithms for entering and leaving the cube, and for routing and broadcasting messages, that *all* peers must implement. Consequently, all peers share the responsibility for the integrity of this delicate topology, which may be a disadvantage. Furthermore, an 'outer' hypercube partitions the network according to global ontologies, for example ontologies for service types. This approach can be seen as a more sophisticated version of the *first* step of our approach to service-discovery. That is, instead of using a hierarchy of JXTA names and the standard JXTA search mechanism, they use global service ontologies that partition a hypercube topology. They claim superior scalability to very large networks for their approach, and it is possible that it is very useful for these situations. Indeed, it could be combined with the *second* stage of service-discovery that we have suggested, that is, OWL and OWL-S service descriptions that are retrieved from the service-providing peer and loaded into an inference engine on the client. However, their approach places an extra burden on all peers, and the benefits must be weighed against this disadvantage. Many ubiquitous-computing applications that have been envisioned have a high degree of *locality*. Peers will mostly communicate with other peers that are physically nearby, and for these situations a local broadcast may be a sufficient solution.

A similar approach is done in [17]. Their Semantic Overlay Network groups nodes according to semantic similarity.

7 Discussion

A significant feature of Oden is that the service consumer controls all evaluation of services. A broker or service-evaluation service could still be implemented to help resource-constrained peers to find and use the services they need, but the choice to use such brokers is up to the service consumer, not to the service provider. Our service-discovery architecture itself does not assume such a broker.

A second important point to note is that a service-seeking peer first filters out potentially interesting services using JXTA's name-matching search, and then loads and evaluates only these services. This two-step process reduces network traffic and the amount of work the service-seeking peer has to do, since a smaller number of advertisements have to travel the network, and there is less data for the peer to evaluate. The first step is a coarse filtering, and the second, using the ontologies, provides precision.

[9]Resource Description Framwork, a W3C Recommendation, `http://www.w3.org/TR/REC-rdf-syntax/`

[10]The precursor to OWL-S.

As we mentioned in Section 2, this process assumes a global namespace of devices and services. This is, of course, difficult to achieve, but there are different ways to alleviate this problem. At one extreme, the first step could be skipped completely, which happens if peers simply search for "*" instead of e.g. "Device:Printer:*". This may be feasible if the contexts of discovery (i.e. the peergroups) are sufficiently small. All evaluation of services would then be performed using their ontologies. This solution would be highly robust, but as we have mentioned, it may give the searching peer far too many service descriptions to evaluate.

At the other extreme, the namespace hierarchy would be extremely fine-grained, so a peer would search for, e.g. "Device:Printer:LaserPrinter:HPLaserJet:*". This approach would give the searching peer exactly what it was looking for, provided that it knows exactly what it wants, and that the hierarchy is stable. These are very strong provisos however, and this solution is highly brittle.

We believe that the optimal solution lies somewhere between these two extremes. The exact layout of the namespace hierarchy is still an open question, but it should only provide broad categories—sufficiently general to avoid the brittleness of a too specific approach, but specific enough to avoid flooding the searching peer with irrelevant advertisements.

A third issue we wish to mention is the concerns that have been stated regarding the performance of JXTA [18, 19]. It should be noted, however, that the JXTA platform is under intense development, and these problems are being addressed. Furthermore, our concern here is not maximum throughput of data between peers, but to build an architecture for service discovery. The evaluation of service advertisements, using an inference engine, currently takes much longer than retrieving the advertisements, so the speed of network traffic is not the most critical issue in this context.

So far, we have not made large-scale experiments with Oden as it is still under development.

8 Summary and Conclusions

Mobility, dynamically changing networks and the desire for automatic configuration without user intervention are factors that combine to motivate the need for powerful *service discovery*. While P2P systems are well suited to fill many of the needs of current and future computer systems, such as ad-hoc and ubiquitous computing networks, no system offers a sufficient solution to the service-discovery problem. In this paper, we have suggested a solution to this issue for the JXTA peer-to-peer network infrastructure.

We have examined existing service-discovery technologies to determine whether any of these can be of use, or at least provide some additional insight into the service-discovery problem. We concluded that none provide the network independence, suitability for P2P systems or expressiveness that we need.

We have presented a novel solution to the problem: Integrating discovery-enabling OWL and OWL-S ontologies with JXTA to provide semantic models of services. This enables service-seeking peers to perform their discovery more intelligently than otherwise possible, using an inference engine. With our research platform Oden, we have shown how to integrate these ontologies with JXTA without compromising important P2P characteristics, that is, Oden fulfills all three criteria in Table 1. A significant feature of Oden is that the service consumer controls the discovery process.

Acknowledgements

The preparation of the manuscript was supported by Vinnova (grant no. 2002-00907) and by The Swedish Research Council (grant no. 621-2003-2991).

References

[1] M. Weiser, "The Computer for the Twenty-First Century," *Scientific American*, pp. 94–100, September 1991.

[2] S. Oaks, B. Traversat, and L. Gong, *JXTA in a Nutshell*. Sebastopol, CA: O'Reilly, 2002.

[3] Kim Topley, *J2ME in a Nutshell*. Sebastopol, CA: O'Reilly, April 2002.

[4] B. Traversat, M. Abdelaziz, D. Doolin, M. Duigou, J. Hugly, and E. Pouyoul, "Project JXTA-C: Enabling a Web of Things," in *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, pp. 282–290, 2003.

[5] A. Schmidt, "Ubiquitous Computing—Computing in Context," November 2002. Submitted to Lancaster University for PhD degree, `http://www.comp.lancs.ac.uk/~albrecht/pubs/`.

[6] S. Helal, "Standards for Service Discovery and Delivery," *IEEE Pervasive Computing*, vol. 1, pp. 95–100, July 2002.

[7] G. G. Richard, *Service Discovery Protocols and Programming*. Developer Guides, McGraw-Hill Education, June 2002.

[8] S. Avancha, A. Joshi, and T. Finin, "Enhanced Service Discovery in Bluetooth," *Computer*, vol. 35, pp. 96–99, June 2002.

[9] S. Waterhouse, D. Doolin, G. Kan, and A. Faybishenko, “Distributed Search in P2P Networks,” *IEEE Internet Computing*, vol. 6, pp. 68–72, January–February 2002.

[10] D. L. McGuinness and F. van Harmelen (eds.), “OWL Web Ontology Language Overview.” W3C Recommendation 10 February 2004, `http://www.w3.org/TR/owl-features/`.

[11] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, eds., *The Description Logic Handbook - Theory, Implementation and Applications*. Cambridge: Cambridge University Press, January 2003.

[12] R. Fikes, G. Frank, and J. Jenkins, “JTP: A System Architecture and Component Library for Hybrid Reasoning,” in *Proceedings of the Seventh World Multiconference on Systemics, Cybernetics, and Informatics*, 2003.

[13] A. Mädche and S. Staab, “Services on the Move - Towards P2P-Enabled Semantic Web Services,” in *Proceedings of the 10th International Conference on Information Technology and Travel & Tourism, ENTER 2003*, (Heidelberg), Springer-Verlag, 2003.

[14] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch, “EDUTELLA: a P2P networking infrastructure based on RDF,” in *Proceedings of the eleventh international conference on World Wide Web*, pp. 604–615, ACM Press, 2002.

[15] U. Thaden, W. Siberski, and W. Nejdl, “A Semantic Web based Peer-to-Peer Service Registry Network,” tech. rep., Learninglab Lower Saxony, 2003.

[16] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl, “A Scalable and Ontology-based P2P Infrastructure for Semantic Web Services,” in *Proceedings of the Second International Conference on Peer-to-Peer Computing*, pp. 104–111, 2002.

[17] A. Crespo and H. Garcia-Molina, “Semantic Overlay Networks for P2P Systems,” tech. rep., Computer Science Department, Stanford University, October 2002.

[18] E. Halepovic and R. Deters, “The Costs of Using JXTA,” in *Proceedings of the Third International Conference on Peer-to-Peer Computing*, pp. 160–167, 2003.

[19] K. Burbeck, D. Garpe, and S. Nadjm-Tehrani, “Scale-up and performance studies of three agent platforms,” in *Proceedings of International Performance, Communication and Computing Conference, Middleware Performance workshop.*, pp. 857–863, April 2004.