Semantic service discovery in dynamic environments

Tommy Gagnes

Norwegian Defence Research Establishment (FFI)

27 September 2007

FFI-rapport 2007/02190

1085

ISBN 978-82-464-1260-3

Keywords

Service Discovery

Web Services

Semantic Web Services

Informasjonsforvaltning

Informasjonsteknologi

Approved by

Rolf Rasmussen	Project manager
Vidar S. Andersen	Director

Sammendrag

Denne rapporten samler tre publiserte artikler innenfor fagområdet “service discovery” i dynamiske omgivelser. Samlet gir artiklene en god introduksjon til fagfeltet, samtidig som de peker på hva som mangler for å forbedre eksisterende teknologier.

English summary

This report contains three peer-reviewed papers in the area of service discovery in dynamic environments. Together, they form a good introduction to the field of service discovery, pointing out what is needed to improve current technologies.

Contents

1 Introduction

This report is a collection of three peer-reviewed papers in the area of semantic service discovery. The papers have been published at different conferences in the field and build on each other. We present them in their original form, in inverse chronological order. We chose to do this because of relevancy, and because the articles are overlapping. In other words one should probably start with the most recent article, and read the remaining articles only if interested. Below, we briefly summarize the papers and the context they were written in.

Assessing Dynamic Service Discovery in the Network Centric Battlefield, published at the IEEE Military Communications Conference (MILCOM) 2007. The paper looks at dynamic service discovery in the context of Network Centric Warfare, arguing that service description and advertisement distribution should be separate issues. Therefore, a layered approach is needed.

A Conceptual Service Discovery Architecture for Semantic Web Services in Dynamic Environments, published at the Semantics-enabled Networks and Services (SeNS) Workshop at the IEEE International Conference on Data Engineering 2006. The paper presents a conceptual architecture for semantic web service discovery in dynamic environments.

Discovering Semantic Web Services in Dynamic Environments, published as a short paper at the IEEE European Conference on Web Services (ECOWS) 2005. The paper presents some important requirement for dynamic service discovery in dynamic environments.

Appendix A Assessing Dynamic Service Discovery in the Network Centric Battlefield

ASSESSING DYNAMIC SERVICE DISCOVERY IN THE NETWORK CENTRIC BATTLEFIELD

Tommy Gagnes
Norwegian Defence Research Establishment (FFI)
Kjeller, Norway

ABSTRACT

In the network centric battlefield it is crucial that participating services can be discovered and used on an as-needed basis. Opportunistic service discovery in dynamic environments imposes some important requirements on the underlying service discovery system. In this paper, we discuss requirements for dynamic service discovery in the network centric battlefield, showing that current technologies for Web Service discovery are insufficient in this kind of environments. We then discuss design aspects of a service discovery architecture, advocating an autonomous federated registry topology and use of aliveness information. Finally, we briefly sketch some of the ingredients of a suitable service discovery infrastructure for a network centric battlefield.

INTRODUCTION

Network centric warfare (NCW) [1] is based on increased sharing of information between decision-makers. The number of nodes in the network producing information will increase as systems and applications on the lower organizational levels need to become more opened up. As a result, the amount of information produced increases, providing more opportunities for dynamic information exchange based on loose coupling and late binding. There are several scenarios where opportunistic and rapid discovery and usage of application layer services can enhance and increase the amount of information accessible to warfighters.

We envision the concept of a service-oriented architecture to become pervasive in the information infrastructure [2]. To exploit this fully in dynamic environments like the network centric battlefield, service discovery mechanisms have to be used to connect producer and consumer nodes dynamically. Typically, dynamic selection of relevant services from some kind of registry node is needed.

Dynamic environments, which can be defined as surroundings with continuous change, may lead to frequent change in both service metadata (service descriptions) and the topology of the nodes that are part of the system. Frequent topology change means that both service nodes and registry nodes can come and go. In other words, they are transient.

A proper service discovery architecture for such an environment would reduce the amount of manual configuration, enable automatic discovery and selection of relevant services, and offer a complete and up-to-date picture of the services available at the given point in time. Moreover, it should be robust in terms of partial failure as well as bandwidth efficient, since nodes in dynamic environments may have wireless connections with low network capacity.

In this paper, we assess dynamic service discovery in the network centric battlefield along the dimensions of service description expressivity, topology, registry network bootstrapping and maintenance, and maintenance of service information in registries. To lay the ground for this, we start by identifying key requirements for a discovery infrastructure deployed in a network centric battlefield. At the end, we present some ideas on how to design a generic service discovery architecture.

REQUIREMENTS FOR DYNAMIC SERVICE DISCOVERY IN THE NETWORK CENTRIC BATTLEFIELD

While dynamic service discovery may not currently be feasible on the lowest level, we see several common requirements for achieving dynamic service discovery at both the lower and intermediate levels of the information infrastructure. Below, we briefly summarize some high-level requirements for a discovery infrastructure that can be deployed in dynamic environments. This builds on work initially presented in [3]:

- Whether the underlying network is a LAN, WAN or a mobile ad hoc network, a unified way to bootstrap and maintain the service discovery infrastructure is needed to avoid frequent manual configuration. There should be automatic discovery of registry nodes in a coherent and transparent way.
- The system should allow flexible resource utilization, since capacity (memory, CPU, storage) and connectivity distribution often are asymmetric.

1-4244-1513-06/07/$25.00 © 2007 IEEE

Limited clients should be allowed to delegate service selection to registry nodes (they may return only the best service advertisement) and thereby prevent receiving too many responses to queries. Especially in wireless environments, it is important to use bandwidth efficiently.

- To ensure discovery of the services available, robustness and survivability against registry failure or disappearance is important. This means that the system cannot depend on centralized components like a single registry.
- Moreover, service discovery should work in environments disconnected from the Internet (e.g. DNS, WWW). Additional artifacts needed by clients to evaluate or use services (e.g. XML schema, ontologies) must be obtained from elsewhere. Such functionality could be provided by the discovery service.
- The discovery infrastructure must provide a fresh view of available services. Responses to queries should mirror the current state in the service network and should not advertise services that are no longer present on the network.
- The infrastructure should support different kinds of service description mechanisms, ranging from simple (name, id, URI specifying a pre-agreed service type), to rich (e.g. semantic descriptions). Thus, both normal Web Services [4] as well as Semantic Web Services should be able to use this infrastructure. Also, services not relying on Web Services standards as their transport should be able to use the service discovery infrastructure. This could for example be services that broadcast information via UDP according to some custom standard (e.g. Tactical Data Links).
- Security must also be handled, but is outside the scope of this paper.

The remainder of this paper tries to take these requirements into account.

ASPECTS OF A SERVICE DISCOVERY ARCHITECTURE

In [5], we discussed semantic service discovery in dynamic environments along the axes of service description expressivity and service discovery infrastructure robustness. Let us take a look at what we think the network centric battlefield will demand from a service discovery architecture. We evaluate the most important standardization and research efforts as we present the different aspects. There is much related work in this area, ranging from current Web Service discovery mechanisms to Semantic Web Services to peer-to-peer computing. We discuss these topics briefly with respect to our goal, which is to discover services in dynamic environments like the network centric battlefield.

Service Description, Querying, and Selection

There are several aspects of a service which can be described, where the most common ones are name, type, operations, parameters (input and output), and attributes (e.g. quality of service attributes). Some efforts also allow goals and processes to be modeled and described.

Several efforts to describe services exist, and common to all of these is that some level of agreement must exist before invocation. The simpler ways to describe a service is using a string for its name, or an URI for its type, typically the case with Web Services, where one would let a URI correspond to a given Web Service Description Language (WSDL) [4] schema registered with a Universal Description, Discovery and Integration (UDDI) [6] registry. In WS-Dynamic Discovery [7], services are also described using Unified Resource Identifiers (URIs). Common to most of the current Web Service discovery standards is that service advertisement expressiveness for these technologies is not very rich and has no explicit semantics.

A more advanced technology is semantic description, which provides a higher level of interoperability crucial to enable dynamic linking across organizational boundaries and the ability to reason about whether a service can fulfill a task. Semantic Web Services thereby allow clients to engage newly encountered services, given a shared semantic model, or ontology. By using semantics we can enhance service descriptions, reduce ambiguity and enable dynamic service usage. There are several efforts in the area of Semantic Web Services research, notably OWL-S [8], WSMO [9], and SAWSDL [10]. These efforts are mostly concerned with describing services, and leave the challenge of service advertisement distribution to existing Web Service standards, specifically UDDI. It is important to point out that semantic service advertisements can become quite large, compared to for example URI strings.

Often, registry technologies have their own Registry Information Model, or RIM. Examples of this are the ebXML registry/repository [11] information model and the UDDI information model. This RIM must be used if one wants to exploit such a technology, and may pose restrictions on the aspects of a service possible to describe. Further, an agreed-upon taxonomy of service types can be registered with some of the registry technologies. This allows lookup based on a type hierarchy. In semantics-enabled registries, inference mechanisms can be used to

find matches based on a subtype hierarchy (e.g. a Radar is a kind of Sensor).

In our opinion, it is arguable whether this is really necessary. There are many examples of efforts trying to "map" their service description efforts to fields in the RIM. The drawback of this is that the registry cannot assist in fine-grained service matching, since it does not know the meaning of the custom fields.

Querying for a service is most often accomplished by filling out a partial template for the service wanted, and submitting this to the registry, which finds service advertisements matching this template. Several filters can be added, e.g. limiting the number of results when querying a registry.

Registry Network Topology

There are numerous alternative topologies that a registry network can be based on, each with their benefits and drawbacks. In [5], we discussed topology thoroughly, so we only give a brief summery here. There are two pure topologies, but in fact, a combination of these seems to us to be the best topology for obtaining the properties we would like to have in a registry network. Figure 1 below shows these topologies.

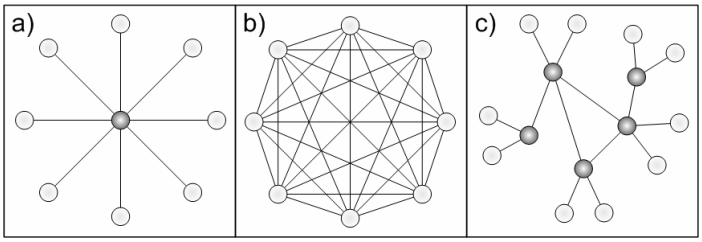

Figure 1. Service discovery topologies.

The *centralized* topology has one node (the registry node) with more responsibility than the others, which allows simple configuration, easy query response control, and a centralized and coherent view of the service network state. Its drawbacks are that it is brittle both to random failures and targeted attacks and that the load on the single node may become high.

In the *decentralized* topology, all nodes are equally important. This means that all nodes form a decentralized registry together and advertise services and evaluate queries themselves. The first advantage of this topology is that the registry always reflects the current state of the service network. Hence, no stale data will exist in the collective registry. The second advantage is that such a topology is extremely resilient to both targeted attacks and random failure. There are, however, several drawbacks to this topology as well, the first one being that the traffic between nodes may create massive overhead. As such, a system based on this topology may not scale to a wide-area network with many participants, and it will certainly consume a lot of bandwidth for administrative tasks. Further, there is little query response control, because all nodes will respond to a query independently of each other. Also, a query posed must be propagated to all nodes, again leading to heavy bandwidth consumption.

Fortunately, there is a third class of topologies, namely the *hybrid* topology. As the name indicates, this topology is a compromise between the centralized and the decentralized topologies. With this approach, some nodes are given more responsibility, creating a dynamic hierarchy with a decentralized topology between them. In peer-to-peer terminology, such nodes are called super-peers, but we will call them registry nodes. There are several variations on this scheme, depending on the level of responsibility given to the registry nodes, query evaluation capability and storage capability.

In peer-to-peer technologies based on *distributed hash tables*, e.g. [12], registry nodes are given as little responsibility as possible. Such systems are based on storage of hashes in the intermediate nodes, and therefore query evaluation other than string matching cannot be performed at the intermediate nodes. Logically, this makes these systems more decentralized, since registry nodes are only forwarding queries and answers. With peer-to-peer technology, the support for dynamic changes in connectivity of peers is an important advantage as reliance on DNS and WWW is not realistic for the environments we target. However, decentralized peer-to-peer systems often have different assumptions than a service discovery system should have. For instance, they may assume that resources (e.g. files) are replicated, and therefore exhaustive querying is not needed. This is not the case with service discovery, where each service should be considered unique. Some peer-to-peer efforts, like JXTA, are based on generic protocols that can carry various advertisements. This facilitates reuse of the peer-to-peer infrastructure.

One could view a *clustered registry* as a hybrid topology as well. With this scheme, one registry is replicated on several nodes. This means that exactly the same content is present at different nodes. An example of a system using this principle is UDDI, where either replication between registry nodes or a hierarchical model may be used.

A combination of the two aforementioned hybrid schemes is what we have termed *federated autonomous registries*. Here, registry nodes operate autonomously, with independent content, giving answers to queries based on more advanced service description efforts. Such autonomous registries can cooperate between them in a registry network, responding to each others' queries. This allows query evaluations to be performed in registries, providing

control over the number of query responses received. Moreover, this scheme may strike a balance between the decentralized and the centralized approach in terms of robustness, load-balancing, and bandwidth-efficiency. An important aspect in this picture is the degree of dynamism with which a federation of registries can be formed and maintained. This should draw on experiences from peer-to-peer technology, allowing new registry nodes to join and leave the registry network, maintaining connectivity between the remaining nodes at any time. By assigning clients to registries in an even distribution, load balancing could be obtained as well. To our knowledge, little research has been done in the area of registry cooperation strategies, so further research on which schemes work best under different conditions is needed. An example of a technology that we sort under this subcategory is the ebXML registry [11]. This solution supports a nonhierarchical multi-registry topology, facilitating federated queries.

Finally, some interesting research has emerged the last years in the complex networks area. Here, properties of various network topologies have been studied, but little has been done in the area of dynamic wiring of the networks. In [13], our view on robustness in decentralized systems is confirmed, topology-wise. In [14], several metrics are used to indicate survivability of different topologies. According to the authors, properties such as low characteristic path length (average number of hops), good clustering (proportion of links between nodes in a neighborhood divided by the maximum number of links possible), and robustness to random and targeted failure are all important for survivability. According to the same paper, the characteristic path length should be low for survivability reasons, with only a few nodes that have long-range connections. This matches quite well with the hybrid topology.

Registry Network Bootstrapping and Maintenance

An important question is how to bootstrap and maintain the registry network topology. To facilitate this, we will consider LANs and WANs separately.

To find out about present registry nodes, discovery of available registries must be carried out. We call this registry discovery. On LANs, registries may be discovered either by manually configuring the registry endpoint or by clients actively using local-scoped multicast to find available registry nodes. Also, registry nodes may periodically issue local beacon messages, enabling clients to do passive registry discovery. For LAN service discovery, WS-Dynamic Discovery [7] is based on local-scoped multicast. A discovery proxy is also specified to reduce the burden on the network.

For WANs, the use of multicast places a too heavy burden on the network. Therefore, manual configuration, or seeding, is necessary at some point in time, connecting different registries from different LANs into a distributed registry network on the WAN level. Some peer-to-peer systems like [12] allow the use of a set of well-known bootstrapping servers, which can point to an appropriate super-peer. Both UDDI and the ebXML registry are suited for WAN-level usage. Notice that Web Service discovery technologies target different domains, and no relationship between them exist. This basically means that one technology must be chosen for dynamic LAN discovery and another one to reach out on the WAN. This can introduce an unnecessary additional round-trip, e.g. if a discovery proxy is used to find a UDDI registry.

Once connected to a registry node that in turn is connected to other registry nodes on the WAN, it is possible to use what we call registry signaling to provide registry client nodes with alternative registry nodes' addresses. These addresses may be used to reconfigure the registry network dynamically in the event of failure, thus increasing the survivability of the registry network. Only peer-to-peer technologies support this feature as of today. However, existing peer-to-peer systems seem to have been designed mostly for scalability and replicated resources (e.g. files) and current research seems more occupied with structured and index-based peer-to-peer systems than unstructured systems.

To summarize, creating an architecture that supports automatic bootstrapping and registry network maintenance will reduce the amount of tedious, manual reconfiguration of registry endpoints that otherwise has to be done by users.

Maintaining Service Information in Registries

To ensure that client nodes issuing queries receive as correct answers to their queries as possible, the service advertisements in the registry nodes should mirror the state in the service network as closely as possible. When the topology and availability of services changes rapidly service advertisements representing obsolete services need to be removed from the system. This is also the case if services fail, meaning that we cannot rely on services de-registering from the registry themselves.

A common way to solve this problem is to build aliveness information into the architecture. Typically, the provider of a service obtains a lease when publishing its service description to the registry. From then on, the provider must periodically confirm that it is alive. Should a service crash, it would not be able to renew its lease, and the service description would be purged from the registry. Leasing mechanisms have been used in Jini [15] and JXTA [12].

Lack of such mechanisms is a major problem with today's technologies for Web Service discovery when being deployed in dynamic environments. Neither UDDI nor ebXML use leasing, and are dependent on services actively de-registering themselves. This is of course not possible in the event of a service provider crash, and is a serious shortcoming of these technologies. WS-Discovery, because of its decentralized nature, does not need an explicit leasing mechanism when used in decentralized mode. However, when used with a discovery proxy the same shortcoming applies to WS-Discovery.

In a dynamic registry network, registry coordination and signaling is important. We use the term registry signaling to describe the exchange of management information that is carried out between registries. This could for instance be to check aliveness of other registries, to share information about other registry nodes in the system, and to send out summary information about the advertisements present in a registry. Moreover, when bootstrapping a registry network, dynamic assignment of registry node responsibility is a challenging problem.

In [16], the authors investigate different self-healing mechanisms (e.g. leasing) in service discovery systems. [17] and [18] evaluates service discovery with respect to node failures and communication failures. However, the authors do not investigate systems that have a dynamic hierarchy as in hybrid peer-to-peer systems.

TOWARDS A COHERENT ARCHITECTURE FOR SERVICE ADVERTISEMENT DISTRIBUTION

Based on the previous sections, we now summarize a few key principles that we think should be followed in a service discovery system that takes into consideration the dynamic nature of a network centric battlefield.

We assume here that no node in the service network is stable and define client nodes, service nodes, and registry nodes as roles that nodes participating in the application-level service network could have. This corresponds to the three roles of provider, consumer, and registry known from the service oriented architecture triangle [19]. It is possible for nodes to engage in several roles simultaneously and obviously several registry nodes would need to exist for robustness reasons. We use the term registry network to refer to a dynamic hierarchy of registry nodes.

Service invocations are performed directly, depending on the service description mechanism used. The service discovery architecture is responsible for establishing contact between clients and services. It should therefore be possible to reuse existing Web Services in dynamic environments by developing a generic service discovery architecture that can be used with different service description models.

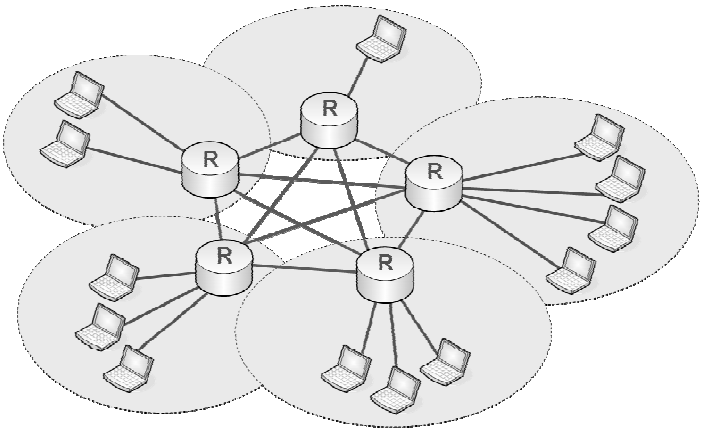

Figure 2. A super-peer registry network with five registry nodes.

It is our opinion that a system based on autonomous, dynamically federated registries is the best solution for wide-area service discovery. These registry nodes must act as super-peers, being able to dynamically connect and disconnect to the system and to forward information about other registries to its clients in case of failure. A registry super-peer is responsible for answering queries based on its knowledge and for forwarding queries and answers to and from other registries. In addition, the registry must cooperate with other registries to maintain the connectivity of the registry network.

We think a registry network topology like in Figure 2 where registries are self-contained, not just an index, provides the best balance between robustness and efficiency. There are lots of different design choices, e.g. to push or pull advertisements between registries, active or passive registry discovery, how many registry nodes on each LAN and so on. Actually, these could even be made configurable on an individual deployment basis. Other configurable parameters could be the interval between registry beacons, the number of registry nodes to traverse for a query, and the advertisement lease period. Some systems today also allow registration for notifications about service advertisements of interest.

Bootstrapping of the registry network can be done either manually or automatically. On the LAN level, the latter could be done either actively or passively, both by using multicast. Once a client has obtained a connection to a local registry, it may ask this registry for other (remote) registries. It should also be possible to do decentralized service discovery on the LAN level as a fall-back solution in the case of registry failure. Initial bootstrapping between registries on different LANs could be done either by providing some seeding information or through some well-known servers.

Letting service advertisements have limited lifetime ensures removal of obsolete advertisements. Service nodes are responsible for renewing their leases with the registry they published the advertisement to.

It is very likely that several service description and query mechanisms will coexist in such a system. A layered approach, separating the service advertisement and query infrastructure from the service description method, will facilitate reuse. This way, primitive devices using only a lightweight URI-matching service discovery (and not necessarily relying on Web Service technology for their delivery) can use the same service discovery infrastructure as the more heavyweight ones based on semantic service descriptions. Some kind of "next header" field like in the Internet Protocol could be present in all registry protocol messages, allowing nodes to choose the right handling of the service description payload. It also lets nodes quickly filter and silently discard messages they cannot understand anyway. This may also provide a "hook" for using compression or binary XML versions to reduce the burden on the network, a not insignificant issue when it comes to XML-based semantic service descriptions, since these typically are quite large.

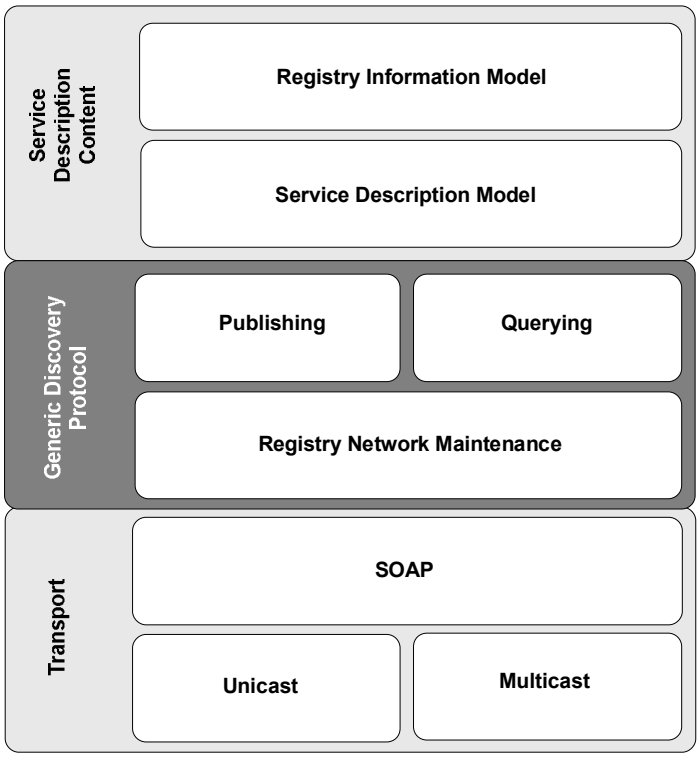

Figure 3. A generic service discovery stack.

Additional benefits of separating the distribution architecture from the service description scheme are that it allows incremental evolution, support for new service description schemes, and promotes reuse. Software libraries for distribution would only need new plug-ins or handlers for new models, keeping the same stack underneath. An example of the opposite is UDDI, where one must try to "map" the service description to the fields that UDDI specifies if another scheme for describing services is to be used.

We would like to reuse the same generic operations and messages, regardless of the payload (based on the service description model). We classify such operations and messages in three categories: registry network maintenance, publishing, and querying. Typical registry network maintenance operations would be to ask for nearby registries, ping a registry, or to obtain a list of other registry nodes from a registry. Publishing operations would be to publish a service description, to renew a lease, to update or remove a service description, and to forward advertisements (in the case of a replication-like registry cooperation strategy). Querying operations would be to query a registry node and receive a result.

While layered, it is important that the system is coherent, meaning that the possibly different parts of the specification stack interoperate and relate to each other. An example of the opposite is the non-existing coherence between WS-Dynamic Discovery and e.g. UDDI.

Figure 3 depicts an example service discovery stack. SOAP would provide the basis for transport and would need both unicast and multicast bindings for registry discovery and decentralized LAN service discovery to work. Multicast could be based on SOAP-over-UDP [20]. On top of SOAP, we have sketched a generic discovery protocol, which would correspond to the generic operations mentioned above. Some kind of protocol profiling could be desirable, since registries typically would have to support more such operations than service and client nodes. Examples of this could be capacity and statistics reports, exchanging registry node lists, uploading service taxonomies, information model extensions and so on.

On the top level, we have illustrated the service description model, of which there possibly could coexist several, and the registry information model. This model would serve as the foundation for exchanging information between registry nodes as mentioned above. A service discovery infrastructure that is independent from the World Wide Web should probably also host additional artifacts needed by nodes in the system. This could be e.g. ontologies and ontology mappings, XML Schema, XML transformations, XQuery transformations and so on.

A unique identification convention, e.g. based on Universally Unique Identifiers (UUIDs) like in UDDI 3.0 would be needed in order to reference published advertisements when updating information, renewing leases, and removing advertisements. Such UUIDs could also be used to cor-

relate query responses received from different registry nodes with a registry node's own results. Furthermore, we think that giving queries their unique query ID is a good approach to avoid query looping between registry nodes.

A final observation is that a hybrid topology probably maps best to a military organization, making it possible to have autonomous registries on each branch of the organization. This means that a network disconnect between branches will not prevent services running on the same organizational level from discovering each other. The more decentralized topology is also in line with theories in [21].

From the above discussion, we summarize that the larger part of this functionality is in fact not tied to a specific service description effort. Also, different deployment strategies should not mandate changing the way one would describe and query for a service.

CONCLUSION

In this article we have claimed that current Web Service discovery technologies are not sufficient for opportunistic service discovery and usage in dynamic environments like the network centric battlefield. Especially aliveness information, coherence between LAN and WAN service discovery, and dynamic registry cooperation is lacking. We have presented several open issues in the area of dynamic cooperating registries, and we have taken an initial step towards defining a set of requirements for a coherent service discovery architecture for dynamic environments. Key requirements are: robustness, registry autonomy, aliveness information, and pluggable service description models. We also discussed topology, and explained why we think a hybrid topology with autonomous, federated registries cooperating dynamically is the direction to move in. Our hope is that this discussion will trigger further research and possibly some standardization or harmonization between existing standards.

ACKNOWLEDGEMENTS

Thanks to my colleagues Bjørn Jervell Hansen and Ketil Lund at FFI and Nanda Kol at the NATO C3 Agency for valuable comments during the preparation of this paper.

REFERENCES

[1] D. S. Alberts, J. Garstka, and F. P. Stein, *Network Centric Warfare: Developing and Leveraging Information Superiority* CCRP Publications Distribution Center, 1999.

[2] K. Lund, A. Eggen, D. Hadzic, T. Hafsøe, and F. Johnsen, "Using Web Services to Realize Service-Oriented Architechture in Military Communication Networks," *IEEE Communications*, 2007.

[3] T. Gagnes and T. Plagemann, "Discovering Semantic Web Services in Dynamic Environments," *European Conference on Web Services (ECOWS)*, 2005.

[4] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, and N. Mukhi, "Unraveling the Web Services Web, An Introduction to SOAP, WSDL, and UDDI," *IEEE Internet Computing*, vol. 6, no. 2, pp. 86-93, 2002.

[5] T. Gagnes, T. Plagemann, and E. Munthe-Kaas, "A Conceptual Service Discovery Architecture for Semantic Web Services in Dynamic Environments," in *ICDEW '06: Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06)* Washington, DC, USA: IEEE Computer Society, 2006, p. 74.

[6] OASIS, "UDDI Version 3.0.2," 2004.

[7] Microsoft Corporation Inc., "Web Services Dynamic Discovery (WS-Discovery)," 2007.

[8] OWL-S Coalition, "OWL-S: Semantic Markup for Web Services, W3C Member Submission," 2004.

[9] D. Fensel, "The Web Service Modeling Framework WSMF," *Electronic Commerce: Research and Applications*, vol. 1, pp. 113-137, 2002.

[10] K. Verma and A. Sheth, "Semantically Annotating a Web Service," *IEEE Internet Computing*, 2007.

[11] OASIS, "ebXML Registry Services and Protocols," 2005.

[12] L. Gong, "JXTA: A Network Programming Environment," *IEEE Internet Computing*, no. 3 2001.

[13] R. Albert and H. &. B. A.-L. Jeong, "Error and attack tolerance of complex networks," *Nature*, vol. 406, p. 378, 2000.

[14] H. P. Thadakamaila, U. N. Raghavan, and S. Kumara, "Survivability of multiagent-based supply networks: a topological perspective," *Intelligent Systems, IEEE*, vol. 19, no. 5, pp. 24-31, 2004.

[15] J. Waldo, "The Jini Architecture for Network Centric Computing," *Communications of the ACM*, vol. 42, no. 7 1999.

[16] C. Dabrowski, "Understanding Self-healing in Service Discovery Systems," *Proceedings of the First ACM SigSoft Workshop on Self-healing Systems (WOSS '02)*, 2002.

[17] C. Dabrowski and K. Mills, "Performance of Service-Discovery Architectures in Response to Node Failures," *the Proceedings of the International Conference on Software Engineering Research and Practice*, pp. 23-26, 2003.

[18] C. Dabrowski, "Understanding consistency maintenance in service discovery architectures during communication failure," *Proceedings of the third international workshop on Software and performance*, pp. 168-178, 2002.

[19] M. N. Huhns, "Service-oriented computing: Key concepts and principles," *IEEE Internet Computing*, vol. 9, no. 1, pp. 75-81, 2005.

[20] M. Gudgin, H. Combs, J. Justice, G. Kakivaya, D. Lindsey, D. Orchard, A. Regnier, J. Schlimmer, S. Simpson, and H. Tamura, "SOAP-over-UDP," September, 2004.

[21] D. S. Alberts and R. E. Hayes, *Power to the edge: command, control in the information age* CCRP Publication Series, 2003.

Appendix B A Conceptual Service Discovery Architecture for Semantic Web Services in Dynamic Environments

A Conceptual Service Discovery Architecture for Semantic Web Services in Dynamic Environments

Tommy Gagnes, Thomas Plagemann, and Ellen Munthe-Kaas
University of Oslo, Department of Informatics
{tommyg, plageman, ellenmk}@ifi.uio.no

Abstract

Web Services technology is being used for increasingly different environments than it was designed for. To facilitate discovery of Web Services in dynamic environments, both service description and distribution of descriptions must be improved. Several research efforts target semantic description of services. However, Semantic Web Service discovery in peer-to-peer-like, dynamic environments where services and registries are transient cannot be based on current mechanisms for distribution of Web Service descriptions. Based on a set of generic requirements, we introduce a conceptual architecture that aims to solve many of the problems related to Semantic Web Service discovery in dynamic environments. The architecture is based on a distributed multi-registry topology and aliveness information. We present several research problems we have identified during our initial work on this architecture.

1. Introduction

Web Services technology is being used for tasks increasingly different from the ones initially targeted. As Web Services technology moves into more peer-to-peer-like, dynamic environments where nodes come and go, discovery of services becomes a bigger challenge than in the original environments initially targeted by Web Service description and discovery technologies. If Web Services are to be used in dynamic environments, there is a need for dynamic discovery of services, which in turn demands two things: richer descriptions of services and better distribution mechanisms for these descriptions.

A significant contribution in this respect is the research on Semantic Web Services, where services are described semantically. Semantic Web Services allow clients to engage newly encountered services, given a shared semantic model, or ontology. By using semantics, we can significantly enhance service descriptions, reducing ambiguity and enabling dynamic service usage.

For distribution of service descriptions, however, most of the Semantic Web Services efforts build on today's mechanisms for Web Service discovery, such as UDDI [1]. We argue that these mechanisms are not particularly well suited for distributing semantic service advertisements in dynamic environments. Our position is that Semantic Web Service descriptions introduce such possibilities in dynamic environments that they deserve a better service advertisement distribution architecture.

By dynamic environments we mean surroundings with continuous change. This may lead to frequent changes in both service descriptions and topology. Frequent topology change means that both services and registries can come and go. In other words, they are transient. An example of a dynamic environment could be a crisis management scenario where members from several agencies, potentially at different locations, have to cooperate in order to run an operation as efficiently as possible. These members carry with them various devices that spontaneously form a network where application layer services are offered. Assembly of emergency response teams must often happen very rapidly, and their different applications are not always designed to work together. To facilitate information sharing in an ad hoc way, upper-level ontologies and service taxonomies could be standardized, facilitating semantic service descriptions, and thereby precise selection of relevant services for information exchange. A proper service discovery architecture for such an environment would reduce the amount of manual configuration, enable automatic discovery and selection of relevant services, and offer a complete and up-to-date picture of the services available at the given point in time. Moreover, it should be robust in terms of partial failure and bandwidth efficient, as nodes in dynamic environments may have wireless connections with low network capacity.

Our research interest lies in developing a service advertisement distribution architecture that supports the possibilities introduced by semantic service descriptions in dynamic environments, such as the environment described in the example above. Our goal is to make it possible to reuse already implemented Web Services in dynamic environments by developing a generic service discovery architecture. Therefore, service invocations are performed directly, depending on the service description mechanism used. The service discovery architecture is responsible for establishing contact between clients and services.

This paper presents preliminary work on a conceptual multi-registry service discovery architecture that supports discovery of Semantic Web Service descriptions in dynamic environments. In the next section, we summarize some general requirements for such architectures. These requirements were initially presented in [2]. Building on this work, the rest of the paper proceeds as follows: Section 3 briefly discusses different topologies with respect to service discovery in dynamic environments. In Section 4, we introduce our conceptual architecture and the research problems we have identified during our work on this architecture. Section 5 briefly discusses related work, before a conclusion is given in Section 6.

2. Semantic Web Service Discovery in Dynamic Environments – General Requirements

Designing a service advertisement distribution infrastructure that supports the possibilities introduced by Semantic Web Service descriptions in dynamic environments raises several research questions. In [2] we presented several general requirements, which we summarize here:

To avoid frequent manual configuration, there should be automatic discovery of registries on LANs and WANs wherever possible. To enable a client to find out about all available services, service discovery should be possible in a coherent and transparent way on LANs and WANs. The responses to queries should mirror the current state in the service network and should not return obsolete service descriptions that represent services that are no longer present on the network.

Since services can be quite complex, service selection based on semantic descriptions is necessary to find the best-suited services for given tasks. This means that it can become more costly to evaluate queries, since reasoning about service descriptions may be necessary. Richer expressivity also allows more detailed service descriptions, which in turn may demand more frequent service advertisement updates (for example the coverage area of a given service) than simple service descriptions.

Also, new functionality such as mediation between different vocabularies may introduce additional queries or hints by the discovery service. This could be the case when an interesting service is found, but an additional translation or mediation service may be needed to use it.

Moreover, service discovery should work in environments disconnected from the Internet. In some cases, additional ontologies may be needed by clients for them to be able to evaluate and use services. Such functionality could be provided by the discovery service.

To enable discovery of available services in dynamic environments, robustness against registry and service failure or disappearance is important. Additionally, in wireless environments, it is important to minimize resource (e.g. bandwidth) usage and to prevent receiving too many responses to queries. Semantic service advertisements can become quite large, compared to the use of for example URI strings.

To our knowledge, no coherent infrastructure for Web Service discovery exists that supports all of these requirements. In the next section, we discuss different properties of various topologies with respect to service discovery.

3. Service Discovery Topology and Dynamic Environments

At the top level, we can categorize different service discovery topologies into three basic topologies, namely (completely) decentralized, distributed, and centralized, shown in Figure 1. The centralized topology has one node with more responsibility than the others, whereas in the decentralized topology, all nodes are equally important. The distributed topology is a compromise between the centralized and the decentralized, where a group of nodes has more responsibility than the others.

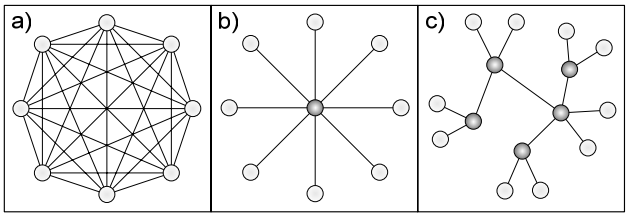

Figure 1. Decentralized (a), centralized (b), and distributed (c) topologies.

3.1. Decentralized Topology

There are several reasons why a decentralized approach is attractive for service discovery. For instance, most decentralized systems are based on service provider nodes hosting their own service advertisements. This means that updates to these advertisements will be immediate, and that there is no need to republish and propagate any new service metadata. Further, the available service advertisements correspond directly to the state in the service network. This means that if a service provider node goes down, its service will no longer be advertised. Decentralized systems are also robust against failure and attacks, since they have no weak points that can bring the whole system down if they disappear.

There are, however, several problems related to the decentralized approach as well: One problem is that a

query must be propagated to all nodes, typically through broadcast. Then, all provider nodes must evaluate the query independently of each other before they return their responses to the querying node. Therefore, the decentralized approach can lead to high bandwidth consumption in the system. It also increases the load on all provider nodes because they have to do processing every time a query is received. This becomes an even greater problem when richer, semantic descriptions are used. Because device capabilities are often different, not all nodes may be able to evaluate queries on semantic service descriptions. This is different from the peer-to-peer world, where simpler (string- or hash-based) advertisements are used. A decentralized topology therefore cannot be used in all cases.

Services, and the real-world resources they represent, should be considered unique. Again, this is different from peer-to-peer systems, where several identical files could be replicated across the system. Therefore, in a service discovery system, all available advertisements should be queried in a deterministic way, not in a random way that does not guarantee discovery of available advertisements. Random querying is often used in peer-to-peer systems.

Another effect of using a decentralized approach is that there is no central point where the total number of responses to a query can be controlled. This lack of *query response control* can at worst, if a query is too broad, lead to "response implosion" at the querying node, meaning that the client is bombarded with responses from potentially all provider nodes in the service network. This certainly increases network load, especially as service descriptions become larger. It also increases the processing load on the querying client, because now it must evaluate all the received responses itself, to make a final selection between the candidate service descriptions it has received. Of course, the number of responses from each node can be limited, but still, query response control is very coarse-grained. The total cost in terms of processing and network capacity therefore is high with this topology.

3.2. Centralized Topology

A centralized topology is probably the most efficient in terms of configuration, because clients only have to maintain the location of one registry. Also, since there is only one centralized location with a complete view of the service network state, query response control is not a problem.

To maintain an up-to-date view of the service network in a centralized registry, removal of old advertisements should be done, since we have to deal with dynamic conditions where services may disappear abruptly. Should the number of service advertisements grow very large, storage may be a constraint as well, since one node must host all advertisements. Further, by delegating service selection to the central registry, query evaluation may only have to be carried out once. The opportunity to allow service selection support in registries is important to relieve constrained clients.

However, a completely centralized solution has problems related to robustness, since we now have a single point of failure. Also, processing load could be high on the central node, since all query evaluation must be performed on a single node. Moreover, with the centralized approach bandwidth must be shared between all querying clients, potentially leading to a bottleneck situation. This must, however, be compared with a greater total load on the network generated by the decentralized topology, which can be critical for networks with broadcast media.

3.3. Distributed Topology

As mentioned, a distributed topology is a compromise between the decentralized and the centralized approach. It is based on a group of intermediate nodes that form a centralized network of nodes that have more responsibility than the rest. Potentially, this can strike a balance between the two "pure" topologies discussed above. Finding the right combination of different properties may enable query response control, robustness, load balancing, and bandwidth-efficiency, and therefore may offer support for dynamic environments. Various systems with a distributed topology place different degrees of responsibility at the intermediate nodes, depending on properties such as query evaluation capability and storage capability. In the peer-to-peer world, intermediate nodes are often termed super-peers, and systems relying on a distributed topology are called hybrid peer-to-peer systems.

We have identified three subcategories of distributed topologies: clusters, distributed hash tables, and multi-registries. Clusters are basically one registry replicated on several nodes. This means that the same content is present at different nodes. An example of this is UDDI. Super-peer distributed hash tables are used in several peer-to-peer systems, like in [3]. Such systems are based on storage of hashes in the intermediate nodes, and therefore, semantic query evaluation cannot be performed at the intermediate nodes in such systems. Logically, this makes these systems more decentralized. The multi-registry topology is based on autonomous registries with independent content. They can cooperate, but to our knowledge, little research has been done in the area of registry cooperation strategies. Examples of technologies that we sort under this subcategory are the CORBA Trading Object Service [4] and the ebXML registry [5]. In the next section, we present our conceptual architecture for semantic service discovery in dynamic environments.

4. A Conceptual Architecture for Semantic Service Discovery in Dynamic Environments

Since neither a completely centralized solution nor a completely decentralized solution is feasible for our purposes, we propose a distributed multi-registry service discovery architecture for distribution of semantic service descriptions. The conceptual architecture is a compromise of the different properties presented in the previous section, potentially making it possible to deploy a coherent, bandwidth-efficient, and robust service discovery infrastructure for both LANs and WANs. The architecture is based on the assumption that there will be an inherent difference in device capability and network capacity. Making a multi-registry system appear externally as one centralized registry and keeping registry content relevant are the core problems of our research. In a dynamic environment, this means that we have to address several new requirements, such as those in Section 2, compared to today's solutions. To give a more in-depth view of our conceptual architecture, we now present the various aspects that constitute it. We start by describing the different roles in the architecture.

4.1. Roles

Our system basically consists of three different roles, namely client nodes, service nodes, and registry nodes. This matches the three roles known from the service-oriented architecture triangle of provider, consumer, and registry [6]. It is possible for nodes to engage in several roles simultaneously. We will describe the roles and their responsibilities below.

A registry node, labeled with an R in Figure 2, is a registry capable of collaborating in a dynamic way with other registry nodes. A registry node can operate autonomously since it stores advertisements and is capable of evaluating queries. In addition, it is responsible for cleaning up advertisements representing obsolete services. Since services can be transient in dynamic environments, service advertisements will need to include aliveness information. This information allows service advertisements to be deleted after a certain period, should their providers not be able to renew their advertisement lease. Typically, one or more registry nodes will be present on each LAN, enabling local service discovery. Connecting local registries from different LANs makes service discovery on the WAN level possible. We call the network formed by connecting several registry nodes a registry network. For robustness and network capacity reasons, we need "thick" registries that contain all the information in the service advertisements, not just pointers to where the advertisements are. As device capabilities can be different, we argue that the most capable nodes should play the roles as registry nodes.

Service nodes, labeled with an S in Figure 2, are the providers of services. They are responsible for obtaining a connection to the registry network to be able to publish the service description of the services it hosts. As mentioned, periodic messages indicating that services are still alive will be important in dynamic environments. Service nodes are responsible for sending such messages. Also, advertisement content, such as coverage area information, could change frequently in dynamic environments. Republishing of updated service advertisements is therefore likely to occur more frequently than with simpler service description mechanisms. Again, the service node is responsible for such republishing. Finally, should the registry node disappear, the service node must try to find another connection point to the registry network and publish its advertisement there.

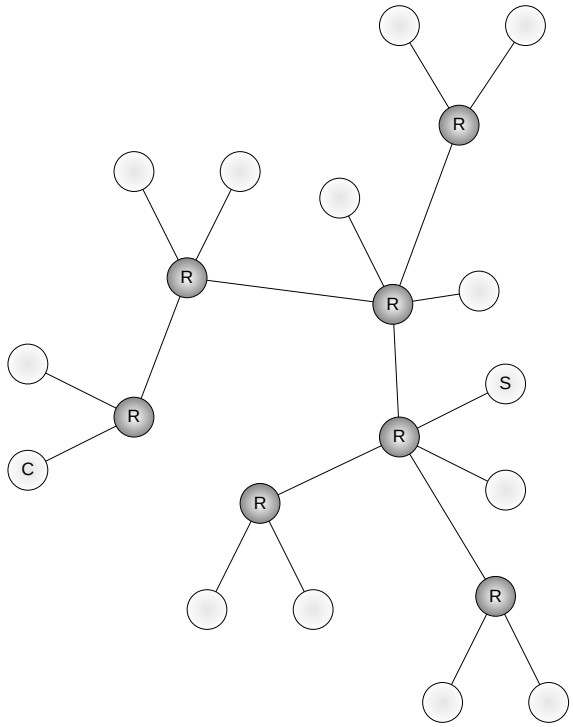

Figure 2. The different roles in the service discovery architecture, with registries forming a registry network.

A client node, labeled with a C in Figure 2, is one that wants to discover a service that can fulfill its needs. To do this, it first has to discover whether there are any registry nodes available. When a client has obtained a connection to the registry network, it can issue a query. Based on the response it gets, it may invoke the service directly, according to the service description mechanism used.

4.2. Service Description and Selection

For automated service selection in dynamic environments to be possible, semantic service descriptions are necessary. Our goal is to build a generic, layered architecture that can be used with different registry information models and languages. It should also be possible to use different query evaluation or matchmaking strategies, as well as registry cooperation strategies. This is different from for example UDDI, where the registry information model is closely tied to the message formats used to carry service advertisements. We aim to standardize interfaces vertically, to allow different registry implementations to register with our distribution system, thereby reusing the distribution system for different service description efforts, WSDL [7] being one example of this. It is even possible to describe services using different service description languages and to publish these.

4.3. Service Composition, Mediation and Translation

To reduce the load on limited devices, service selection, mediator selection, composition and reasoning support in registries may be needed. This will need protocol support from the service discovery architecture. It is a topic of further research to see whether it is feasible to design these protocols in a generic way, allowing reuse of the protocols with different service description languages.

4.4. Service Discovery Topology

By service discovery topology we mean the topological structure of the service discovery system. To cover all possible uses of a service discovery architecture, we differentiate between LAN service discovery and WAN service discovery. By LAN service discovery we mean discovering available services on a LAN level. As we elaborate below, LAN service discovery should ideally be centralized or distributed, but may also be performed in a decentralized way.

To enable remote service discovery between different LANs, a WAN level registry network system is needed. The registry topology should be distributed, but not completely decentralized, so that clients can connect to other registry nodes in the case of a registry node failure. As discussed in Section 3, this also allows the use of a query response control mechanism, which is not possible in completely decentralized systems.

4.5. Registry Discovery

An important question is how to bootstrap and maintain the registry network topology. To facilitate this, we advocate a coherent architecture for both LANs and WANs, using the best mechanisms available for these two different cases in an integrated way.

To find out about present registry nodes, discovery of available registries must be carried out. We call this *registry discovery*. Registries may be discovered either by manually configuring the registry endpoint or by clients actively using local-scoped multicast to find available registry nodes on LANs. Also, registry nodes could issue local beacon messages, enabling clients to do passive registry discovery.

For WANs, the use of multicast places a too heavy burden on the network. Therefore, manual configuration, or seeding, is necessary at some point in time, connecting different registries from different LANs into a distributed registry network on the WAN level, as shown in Figure 4. However, once connected to a registry node that in turn is connected to other registry nodes on the WAN, it is possible to use what we call registry signalling to provide the client node with alternative registry nodes' addresses. These addresses may be used in the event of failure, and may help reduce the amount of tedious, manual reconfiguration of registry endpoints. Actually, registry nodes can use this mechanism between them as well, allowing them to respond to dynamic changes in the registry network topology.

4.6. Registry Support

As mentioned, we cannot rely on WWW and DNS availability in dynamic environments, since disconnection from the Internet may sometimes be the case. Hence, regular XML Schema and ontology import mechanisms may have to be bypassed. To remove dependency on Internet availability, a repository for ontologies and XML Schemas is needed. Our registry network could fill this role, meaning that our architecture would need additional functionality and protocols.

4.7. Service Advertisement Publishing and Discovery

We will now look at how distribution of service advertisements can be done in our architecture. The challenge is to do this in a coherent way on LANs and WANs. We base our architecture on publishing service advertisements to registry nodes and keeping the advertisements close to the service nodes, letting queries pull them towards the interested clients.

For LANs, the normal mode is having one or more registry nodes on each LAN, and letting the other nodes use these registry nodes for advertisement and discovery. This is shown in the left part of Figure 3, and means that we advocate centralized or distributed topologies on the LAN level. The reasons for this are the properties of the topologies discussed in Section 3. Compared to broadcast or multicast, using unicast communication towards one or more central points to query for services can save a significant amount of total bandwidth. This is because it allows centralized control of the number of responses to queries. There will, however, be an additional cost of publishing and updating advertisements, compared to the decentralized approach. A registry on the LAN may or may not be connected to other registries inside or outside the local network.

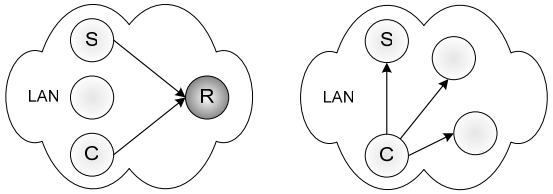

Figure 3. Two modes of LAN service discovery. Centralized/distributed with one/more registries, and decentralized without any registry.

In dynamic environments, registries may disappear abruptly, meaning that we could end up without any available registry nodes. Without a registry, a service node offering a service and a client node looking for services of that kind will not be able to find out about each other, even though they are located close to each other on the same local network. If no registry is available, using decentralized LAN service discovery could ensure that local services still can be discovered. This is shown in the right part of Figure 3. The use of a decentralized discovery is a fallback solution to allow local service discovery in the case where no registry nodes are present, which can occur in dynamic environments. We base such a mechanism on locally scoped multicast, where queries and possibly advertisements are multicasted on the local network. This is not bandwidth efficient, but it should make it possible to find out about available services on the LAN in all cases. Interestingly, decentralized service discovery is the same mechanism as the one used for registry discovery on LANs, which we described above.

At the WAN level, the preferred way to announce and discover services on a wide area scope in our system will be to find a connection point to the registry network. This is done through the mechanisms for registry discovery described above. By connecting to the registry network, it is possible to transparently query all other registries on the WAN, which gives a complete overview of the service network. This is shown in Figure 4. In the case where no information about other registry nodes on the WAN can be obtained, it is not possible to perform WAN discovery.

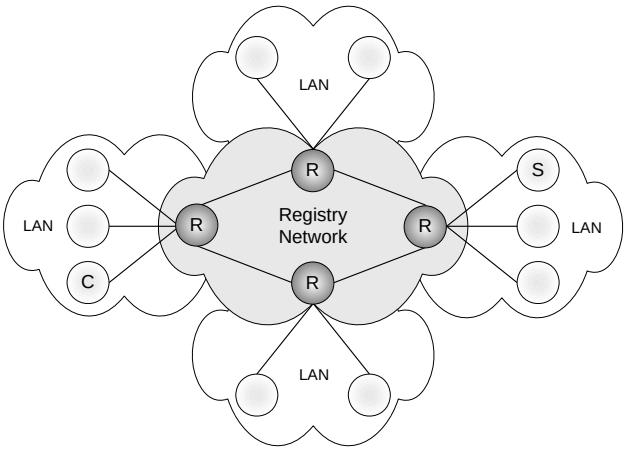

Figure 4. Connecting service discovery mechanisms from different LANs into a WAN registry network for service discovery.

We have identified several research problems from working on this architecture. The architecture is based on the idea that if a registry is available on the LAN, all publishing and querying on the WAN level goes through this registry. In the case where there are two or more registry nodes locally, this may lead to redundant queries being forwarded on the registry network. In other words, a registry node should take into account the existence of other registry nodes locally. There must be some coordination between local nodes so that, at any time, only one node (or a predefined number of nodes) acts as the gateway to the WAN-level registry network. This can be based on the idea that if two registries can discover each other through multicast, they are on the same network segment. Of course, this only works when LAN multicast is available. Strategies for forwarding advertisements or queries are part of the subject registry cooperation, which is discussed below.

4.8. Service Reconfiguration and Advertisement Removal

We have mentioned that the service advertisements in the multi-registry should mirror the state in the service network as well as possible. This can be hard in dynamic environments, where the topology changes rapidly. To do this, we need to remove service advertisements representing obsolete services from the system. This is also the case if services fail, meaning that we cannot rely

on services deregistering from the registry themselves. Instead, some kind of leasing mechanism must be incorporated into the architecture. Such a mechanism is based on periodic confirmation of service availability and has been used in Jini [8] and JXTA [3]. Lack of such mechanisms is a major problem with today's technologies for Web Service discovery when being deployed in dynamic environments. To prevent non-existent services from being discovered, aliveness information should be used to delete old service advertisements from the registry. While this is a responsibility of each registry node, the service nodes will be responsible for renewing their leases with their associated registry node at regular intervals.

4.9. Registry Cooperation

As already touched upon, there are several unsolved research problems in the area of registry cooperation on the WAN level. We split them in two broad categories: forwarding and coordination. While closely related, forwarding deals with sharing information between registries, while coordination deals with principles for configuring and managing the registry network dynamically. The key role of the registry network is to forward queries and advertisements between registry nodes on different LANs. Several different strategies for doing this can be used, including increasing the reach of a query gradually in several rounds, random walks, or broadcasting in the registry network. This has been researched in the area of peer-to-peer computing. Loop avoidance must also be taken care of. Flexible query and advertisement forwarding between registries is another goal of our research. Is it possible to design generic protocols that can carry various payloads, described by different ontologies and registry information models?

Registry coordination and signalling is important in dynamic environments. As mentioned earlier, we use the term *registry signalling* to describe the exchange of management information that is carried out between registries. This could for instance be to check aliveness of other registries, to share information about other registry nodes in the system, and to send out summary information about the advertisements present in a registry. Moreover, when bootstrapping a registry network, dynamic assignment of registry node responsibility is a challenging problem. Some nodes may be more willing to take on the role as a registry node than other nodes. To prevent all nodes from taking on the registry node role, a policy may have to be used for negotiating who will be assigned such a role. Such a policy could for instance include something like "try to maintain three registries on each LAN." Since we target dynamic environments, such negotiation may happen on a regular basis, depending on changes in the registry network state.

4.10. Service Discovery Protocols

We now describe the architectural components in the registry discovery system. These are illustrated in Figure 5. As we want to build on existing efforts, we differentiate between basic, already existing Web Service messaging protocols and our own discovery protocols.

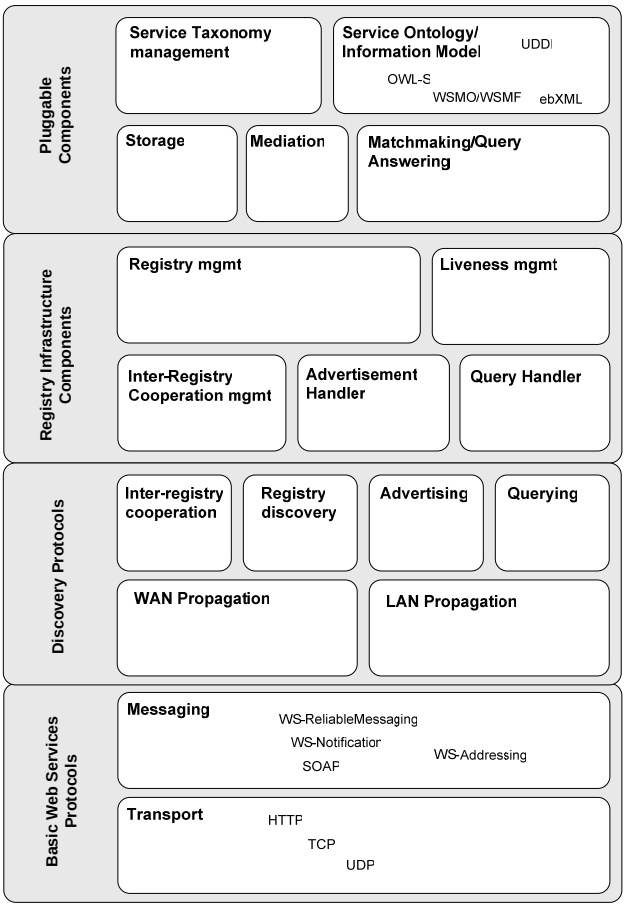

Figure 5. The components of our service discovery architecture.

The Basic Web Service protocols are SOAP and Web Service messaging protocols (WS-Notification [9], WS-Addressing [10]), as well as bindings to different transport protocols (HTTP, UDP etc.). We will use these existing protocols as bearers of our own discovery protocols. This enables reuse of protocol implementations and a level of interoperability at the transport and messaging levels.

Moving further up our stack, we find our initial attempt to make generic protocols that carry service advertisements and queries. Also, the protocols for bootstrapping the registry network are located here. By making the advertisement and query protocols generic, our goal is to allow the use of different registry

information models and service description ontologies in the case of Semantic Web Services. By offering this generic architecture, it should be possible to develop and improve the service advertisement distribution system separately from various service description efforts and to facilitate reuse instead of having to develop a distribution system along with each new service description effort.

The two first building blocks of the discovery protocols are the propagation protocols for LANs and WANs. They deal with transport of service advertisements and queries, registry discovery and inter-registry cooperation messages. The LAN version uses multicast, whereas the WAN version uses the registry network.

Clients of the service discovery system will first need to use the registry discovery protocol. Discovery of registries is done via registry advertisements (passive registry discovery) and queries for available registries (active registry discovery), which are the responsibilities of this building block. It uses the propagation protocols for transport.

The advertising building block is the protocol for advertising services. It has a flexible payload, since it must be able to carry various service description formats. A mechanism for dealing with aliveness information must be included here. The same goes for the querying component, which should be as generic as possible. It mainly uses the WAN propagation mechanism, but can use LAN propagation for decentralized service discovery using multicast on the LAN. For WAN service discovery, there must be some way to limit the reach of queries in the messages used here.

The inter-registry cooperation building block must provide mechanisms for the problems introduced when discussing registry cooperation above, and is responsible for tying the registry network together.

4.11. Registry Infrastructure Components and Pluggable Components

Moving higher up the stack in Figure 5, Registry Infrastructure Components are responsible for implementing the discovery protocols. Handling advertisements and queries therefore are natural tasks. The actual query evaluation and storage of advertisements is dispatched to the pluggable components at the level above. We envision standardized interfaces, preferably Web Service interfaces for registering plug-ins and receiving notifications. Another important task that must be handled on this level is removal of expired advertisements.

The pluggable components in the architecture are specific to the service description language and information models used. Examples of service description efforts can be found in Section 5. Different matchmaking algorithms and service taxonomy evaluation components are registered as listeners to the registry infrastructure components on the level below.

It is subject to further research to explore what is possible in this area. For instance, algorithms for topology management and negotiation or mediation algorithms could be plugged in, given that it is feasible to design a generic interface to the lower level.

5. Related Work

There is much related work in this area, ranging from current Web Service discovery mechanisms to Semantic Web Services to peer-to-peer computing. We discuss these topics briefly with respect to our goal, which is to discover Semantic Web Services in dynamic environments.

We can split existing technologies for Web Service discovery into LAN discovery and WAN discovery (WAN registries can be used on LANs as well). For WAN discovery, both UDDI [1] and the ebXML [5] registry make it possible to use multiple registries or to let a registry consist of several nodes, which can help achieve robustness and scalability. In UDDI, either replication between registry nodes or a hierarchical model may be used. The ebXML solution supports a non-hierarchical multi-registry topology, facilitating federated queries. Both technologies are likely to support semantic search to some degree in the future. For LAN service discovery, WS-Dynamic Discovery [11] is based on local-scoped multicast, which is similar to the LAN part of our service discovery architecture. A Discovery Proxy is also specified to reduce the burden on the network. WS-Dynamic Discovery facilitates registry discovery and local service discovery based on URI matching. However, current technologies target different domains, and no relationship between them exists. This basically means that one technology must be chosen for dynamic LAN discovery and another one to reach out on the WAN. This can introduce an unnecessary additional round-trip, e.g. if a Discovery Proxy is used to find a UDDI registry. Further, the service advertisement expressiveness for these technologies is not very rich and has no support for semantics. Yet another problem with using these technologies is that if a service goes down unexpectedly, its advertisement will stay in the registry because there is no aliveness information in a UDDI or ebXML registry and the Discovery Proxy of WS-Dynamic Discovery. As such, they are not very well suited for dynamic environments.

There are several efforts in the area of Semantic Web Services research, especially OWL-S [12], FLOWS [13], WSMO [14], WSDF [15] and WSDL-S [16]. Rich, semantic descriptions of services are important to

facilitate automated service discovery in dynamic environments, but leave the challenge of service advertisement distribution to the existing Web Service standards, specifically UDDI.

The peer-to-peer research community has developed a host of different schemes for distributing resource advertisements in a distributed way. Some of the efforts, like JXTA [3], are based on generic protocols that can carry various advertisements. Structured systems (based on e.g. distributed hash tables) do not support semantic query evaluation at intermediate nodes. On the other hand, our notion of a registry network has many similarities with unstructured hybrid peer-to-peer architectures, where registry nodes are similar to super-peers. Especially the support for dynamic changes in connectivity of peers is an advantage, as reliance on DNS and WWW is not realistic for the environments we target. There are, however, several important differences between peer-to-peer systems and our registry network. One difference between peer-to-peer-systems and Semantic Web Service discovery is that the number of advertisements will be lower than in peer-to-peer-systems. Also, reasoning about Semantic Web Services may be too complex for certain clients, which demands service selection support at the registries.

A few research efforts have looked at some of the requirements identified in Section 2. In [17], the Gnutella overlay network is used to advertise semantic service descriptions. However, there are problems related to query response control, bandwidth usage, and registry discovery. In [18], Universal Plug and Play (UPnP) is used to carry semantic service descriptions. However, UPnP only works on a single LAN. Further, in [19], the registries for publication are divided according to service advertisement content, which is not acceptable in dynamic environments where registries may fail. In [20], a peer-to-peer infrastructure for Semantic Web Services based on a hypercube scheme is presented. GridWine [21] is an overlay network built on top of a distributed hash table. Both distributed hash tables and hypercubes are vulnerable to a high churn rate, which is typically the case in dynamic environments. Finally, [22] presents a super-peer based solution that has no aliveness information and no LAN discovery.

6. Conclusion and Further Work

To conclude, current Web Service discovery mechanisms do not support our requirements well enough. The opportunities enabled by semantic service descriptions in dynamic environments deserve a better infrastructure for service advertisement distribution. Our hypothesis is that a coherent and robust multi-registry system for Semantic Web Service discovery on both LANs and WANs can help overcome these problems. In the future, we plan to implement an infrastructure based on the conceptual architecture described in this paper. To learn more about different strategies, experimentation with different parameters will be important.

7. Acknowledgements

This work was sponsored by the Norwegian Defence Research Establishment (FFI).

8. References

[1] OASIS. UDDI Version 3.0.2. Clement, L., Hately, A., von Riegen, C., and Rogers, T. Oct. 2004.

[2] Gagnes, T., Plagemann, T., and Munthe-Kaas, E., "Discovering Semantic Web Services in Dynamic Environments," *European Conference on Web Services (ECOWS)*, 2005.

[3] Gong, L., "JXTA: A Network Programming Environment," *IEEE Internet Computing*, vol. 5, no. 3, May/June 2001.

[4] Object Management Group. Trading Object Service Specification, version 1.0. 2000.

[5] OASIS. ebXML Registry Services and Protocols Version 3.0. Fuger, S., Najmi, F., and Stojanovic, N. May 2005.

[6] Huhns, M. N. and Singh, M. P., "Service-oriented computing: Key concepts and principles," *IEEE Internet Computing*, vol. 9, no. 1, pp. 75-81, 2005.

[7] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. Web Services Description Language (WSDL) 1.1, W3C Note. Mar. 2001.

[8] Waldo, J., "The Jini Architecture for Network Centric Computing," *Communications of the ACM*, vol. 42, no. 7, July 1999.

[9] OASIS. Web Services Base Notification 1.3 (WS-BaseNotification). Graham, S., Hull, D., and Murray, B. July 2005.

[10] W3C. Web Services Addressing, W3C Member Submission. Box, D. and Curbera, F. Aug. 2004.

[11] Microsoft Corporation Inc. Web Services Dynamic Discovery (WS-Discovery). Oct. 2004.

[12] OWL-S Coalition. OWL-S: Semantic Markup for Web Services, W3C Member Submission. 2004.

[13] Grüninger, M., Hull, R., and McIlraith, S. A., "A First-Order Ontology for Semantic Web Services," *W3C Workshop on Frameworks for Semantics in Web Services*, 2005.

[14] Fensel, D. and Bussler, C., "The Web Service Modeling Framework WSMF," *Electronic Commerce: Research and Applications*, vol. 1 pp. 113-137, 2002.

[15] Eberhardt, A., "Ad-hoc Invocation of Semantic Web Services," *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2004.

[16] Akkiraju, R., Farell, J., Miller, J. A., Nagarajan, N., Sheth, A., and Verma, K., "Web Service Semantics - WSDL-S," *W3C Workshop on Frameworks for Semantics in Web Services*, 2005.

[17] Paoalucci, M., Sycara, K., Nishimura, T., and Srinivasan, N., "Using DAML-S for P2P Discovery," *Proceedings International Conference on Web Services*, 2004.

[18] Masuoka, R., Labrou, Y., Parsia, B., and Sirin, E., "Ontology-Enabled Pervasive Computing Applications," *IEEE Intelligent Systems*, vol. 18, no. 5, pp. 68-72, September/October 2003.

[19] Verma, K., Sivashanmugam, K., Sheth, A., and Patil, A., "METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services," *Journal of Information Technology and Management*, 2004.

[20] Schlosser, M., Sintek, M., Decker, S., and Nejdl, W., "A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services," *Proceedings of the International Conference on Peer-to-Peer Computing*, 2002.

[21] Aberer, K., Cudré-Mauroux, P., Hauswirth, M., and Van Pelt, T., "GridVine: Building Internet-Scale Semantic Overlay Networks," *Proceedings of the International Semantic Web Conference*, 2004.

[22] Thaden, U., Siberski, W., and Nejdl, W., "A Semantic Web based Peer-to-Peer Service Registry Network," *Technical Report, Learning Lab Lower Saxony*, 2003.

Appendix C Discovering Semantic Web Services in Dynamic Environments

Discovering Semantic Web Services in Dynamic Environments[1]

Tommy Gagnes, Thomas Plagemann, and Ellen Munthe-Kaas

University of Oslo, Department of Informatics,
P.O. Box 1080 Blindern, 0316 Oslo, Norway
{Tommyg, Plageman, Ellenmk}@Ifi.uio.no

Abstract. Web Services are rapidly taking over many of the tasks previously solved by using traditional middleware, also in more peer-to-peer-like, dynamic environments where services and registries are transient. With the advent of Semantic Web Services, rich service description is possible, facilitating dynamic discovery of services. However, our position is that current technologies for Web Service discovery are not sufficient as an infrastructure for Semantic Web Service discovery in such dynamic environments. We explain this by analyzing general requirements for Semantic Web Service discovery in dynamic environments and the shortcomings of recent research results with respect to these requirements. We give a brief overview of our ideas on how to fulfill these requirements and outline work towards an architecture for semantic web service discovery in dynamic environments.

1 Introduction

Web Services discovery mechanisms were initially designed for classical Internet environments where the network topology and availability of hosts is relatively static. However, as Web Services technology is applied in more dynamic environments, discovery of web services becomes more challenging. Such environments have significantly different characteristics than the original environment Web Services discovery mechanisms were designed to operate in. By discovery of Semantic Web Services in dynamic environments we mean finding available Web Services in surroundings with continuous change in service descriptions and topology, which also means that services and registries can be transient. This means that the architecture for service advertisement distribution must be based on a new set of requirements.

Consider a crisis management scenario, where members from different agencies carry with them networked devices that run different applications and services that support them in doing their tasks. Shared upper-level ontologies for information exchange and service taxonomies have been standardized. Ideally, all participants working in the geographical area of the crisis, as well as people working elsewhere, should be able to cooperate by providing and discovering services to exchange information in an ad-hoc way. In such a scenario, it is critical that all relevant services can be

[1] This work was sponsored by the Norwegian Defence Research Establishment (FFI).

discovered, and that a precise selection can be made among available services. This is a typical example of a dynamic environment where Semantic Web Services could be used for automatic selection of services. However, we argue that today's solutions for discovering available semantic web service descriptions, building largely on current Web Services discovery mechanisms, are not satisfactory in this respect. Our research interest can therefore be formulated as follows: In a dynamic environment where services and possibly registries are transient, how can we facilitate opportunistic, automated discovery of available and relevant Web Services in a coherent way?

2 Requirements for Semantic Service Discovery in Dynamic Environments

Below, we present the most important general requirements imposed on a system for discovery of semantic web service descriptions in dynamic environments.

1. To minimize continuous manual configuration, there should be automatic registry discovery on LANs and WANs. Dynamic configuration of a registry network as well as different registry cooperation strategies should be possible.
2. Query responses should mirror the current state in the service network. This can be hard in dynamic environments, where topology changes rapidly. Service discovery should be possible in a coherent and transparent way on LANs and WANs.
3. To enable discovery of available services in dynamic environments, robustness against registry and service failure or disappearance is important. This means that a centralized solution will not suffice, and that old service advertisements should be removed to prevent discovery of obsolete services.
4. To find the best-suited services for given tasks, selection of the best services among many others, based on semantic descriptions, should be possible.
5. In wireless environments, it is important to minimize resource (e.g. bandwidth) usage and to prevent receiving too many responses to queries. This can happen if we have a completely decentralized solution, where potentially all nodes can answer queries independently of each other. To relieve constrained clients, the opportunity to allow service selection support in registries is important.
6. Service discovery should work in environments disconnected from the Internet. Dependency on DNS and World Wide Web should be avoided.

In the following section, we will examine recent research and standardization efforts to see how they support our list of general requirements.

3 Critical Review of Recent Research and Standardization Efforts

We can split existing technologies for Web Service discovery into LAN discovery and WAN discovery (WAN registries can be used on LANs as well). For WAN discovery, both UDDI [1] and the ebXML registry [2] support finding services by name, type, binding, and according to a taxonomy. It is possible to use multiple registries or to let a registry consist of several nodes, which can help achieve robustness and scal-

ability. In UDDI, either replication between registry nodes or a hierarchical model may be used. The ebXML solution supports a non-hierarchical multi-registry topology, facilitating federated queries. Both technologies are likely to support semantic search in the future. For LAN service discovery, WS-Dynamic Discovery [3] is based on local-scoped multicast. A Discovery Proxy is also specified to reduce the burden on the network. WS-Dynamic Discovery facilitates registry discovery and local service discovery based on URI matching.

However, current technologies target different domains, and no relationship between them exists. This can introduce an unnecessary additional round-trip, e.g., if a Discovery Proxy is used to find a UDDI registry. Further, the service advertisement expressiveness for these technologies is not very rich and has no support for semantics. Yet another problem with using these technologies is that if a service goes down unexpectedly, its advertisement will stay in the registry because there is no aliveness information in a UDDI or ebXML registry and the Discovery Proxy of WS-Dynamic Discovery. As such, they are not very well suited for dynamic environments.

There are several efforts in the area of Semantic Web Services research, especially OWL-S [4], FLOWS [5], WSMO [6], and WSDL-S [7]. Rich, semantic descriptions of services are important to facilitate automated service discovery in dynamic environments, but leave the challenge of service advertisement distribution to the existing web service standards, specifically UDDI.

A few research efforts have looked at some of the requirements identified in Section 2. In [8], the Gnutella overlay network is used to advertise semantic service descriptions. However, there are problems related to query response control, bandwidth usage, and registry discovery. In [9], Universal Plug and Play (UPnP) is used to carry semantic service descriptions. However, UPnP only works on a single LAN. Further, in [10], the registries for publication are divided according to service advertisement content, which is not acceptable in dynamic environments where registries may fail. In [11], a peer-to-peer infrastructure based on a hypercube scheme is presented. Hypercubes are vulnerable to a high churn rate. Finally, [12] presents a super-peer based solution that has no aliveness information and no LAN discovery. To conclude, current Web Service discovery mechanisms do not support our requirements well enough. The opportunities that semantic description of services gives us in dynamic environments demands a better infrastructure for service advertisement discovery.

4 A Coherent, Multi-Registry Semantic Web Service Discovery Architecture

Our research interest lies in developing a multi-registry service discovery architecture suited for semantic web service discovery in dynamic environments. The architecture will strive to address the requirements identified in Section 2. Its key elements are:

1. For automated service selection, semantic service descriptions are necessary.
2. To remove dependency on Internet availability, a repository for ontologies and XML Schemas is needed.

3. To help limited devices select services that are semantically described, mediation, composition and reasoning support in registries may be needed.
4. To avoid manual configuration of service and registry locations, the use of local-scoped multicast for local LAN service and registry discovery should be possible.
5. To enable remote service discovery between different LANs, a WAN level registry network system is needed. If no registry is available, fallback to decentralized LAN service discovery could ensure that local services still can be discovered.
6. To prevent non-existent services from being discovered, aliveness information should be used to delete old service advertisements from the registry.
7. The registry topology should be distributed, but not completely decentralized, so that clients can connect to other registry nodes in the case of a registry node failure. This also allows the use of a query response control mechanism, which is not possible in completely decentralized systems. Registry cooperation can enable the use of a single connection point to the service discovery system.
8. We aim to build a layered architecture that can be used with different registry information models, languages, registry cooperation and matchmaking strategies.

We are currently refining this architecture. In the future, we plan to implement an infrastructure based on the architecture, and to experiment with different parameters.

References

1. Clement, L., Hately, A., von Riegen, C., and Rogers, T. (eds.): UDDI Version 3.0.2, OASIS (2004)
2. Fuger, S., Najmi, F., and Stojanovic, N. (eds.): ebXML Registry Services and Protocols Version 3.0, OASIS (2005)
3. Microsoft Corporation Inc.: Web Services Dynamic Discovery (WS-Discovery) (2004)
4. OWL-S Coalition: OWL-S: Semantic Markup for Web Services, W3C Member Submission (2004)
5. Grüninger, M., Hull, R., McIlraith, S. A.: A First-Order Ontology for Semantic Web Services, W3C Workshop on Frameworks for Semantics in Web Services (2005)
6. Fensel, D., Bussler, C.: The Web Service Modeling Framework WSMF, Electronic Commerce: Research and Applications vol. 1 (2002) 113-137
7. Akkiraju, R., Farell, J., Miller, J. A., Nagarajan, N., Sheth, A., Verma, K.: Web Service Semantics - WSDL-S, W3C Workshop on Frameworks for Semantics in Web Services (2005)
8. Paoalucci, M., Sycara, K., Nishimura, T., Srinivasan, N.: Using DAML-S for P2P Discovery, Proceedings International Conference on Web Services (2004)
9. Masuoka, R., Labrou, Y., Parsia, B., Sirin, E.: Ontology-Enabled Pervasive Computing Applications, IEEE Intelligent Systems vol. 18, issue 5 (2003) 68-72
10. Verma, K., Sivashanmugam, K., Sheth, A., Patil, A.: METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services, Journal of Information Technology and Management (2004)
11. Schlosser, M., Sintek, M., Decker, S., Nejdl, W.: A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services, Proceedings of the International Conference on Peer-to-Peer Computing (2002)
12. Thaden, U., Siberski, W., Nejdl, W.: A Semantic Web based Peer-to-Peer Service Registry Network, Technical Report, Learning Lab Lower Saxony (2003)