Using DAML-S for P2P Discovery

Massimo Paolucci[1], Katia Sycara[1], Takuya Nishimura[1,2], Naveen Srinivasan[1]

[1]Canegie Mellon University, USA

[2]Media Technology Development Division, SONY Corporation, Japan

{paolucci,katia,nishi,naveen}@cs.cmu.edu

***Abstract*—Mechanisms for Web services Discovery proposed so far have assumed a centralized registry that collects information about all the Web services available at any given time. Centralized registries are performance bottlenecks and may result in single points of failure. In this paper, we propose an alternative architecture based on a P2P connection between Web services and we show how to perform capability matching between Web services on the Gnutella network.**

***Index Terms*— P2P, DAML-S, Discovery, Capability Matching**

I. Introduction

Discovery of Web services is becoming a hot topic as Web services become more widespread. Much of the work on Web services discovery is based on centralized registries such as UDDI [18] or the DAML-S Matchmaker [13][14]. An architecture based on a centralized registry assumes that every Web service coming on line advertises its existence and its capabilities/functionalities with the registry; and that every service requester contacts the registry to discover the most appropriate Web service and gather information about it. Centralized registries are effective since they guarantee discovery of services that have registered. On the other hand, they suffer from the traditional problems of centralized systems, namely they are performance bottlenecks and single points of failure. In addition, they may be more vulnerable to denial of service attacks. Moreover, the possible storage of vast numbers of advertisements on centralized registries hinders the timely update, as changes in the availability and capabilities of providers change. These problems can be partially alleviated through replication of servers, to mitigate against single point of failure and performance bottlenecks. Leasing mechanisms may force providers to refresh their records keeping the registry up to date. Yet, it is still an open question whether centralized registries will scale up to the needs of Web services.

Peer-to-Peer (P2P) computing provides an alternative that does not rely on centralized services; rather it allows Web services to discover each other dynamically. Under this view, a Web service is a node in a network of peers, which may or may not be Web services. At discovery time a requesting Web service queries its neighbors in the network. If any one of them matches the request, then it replies, otherwise it queries its own neighboring peers and the query propagates through the network[1]. Such architecture does not need a centralized registry since any node will respond to the queries it receives. P2P architectures do not have a single point of failure; rather the high connectivity guarantees that the message reaches the provider. Furthermore, each node contains its own indexing of the existing Web services so there is no danger of a bottleneck effect. Finally, nodes contact each other directly, so there are no delays with the propagation of new information.

The reliability provided by the high connectivity of P2P systems comes with performance costs and lack of guarantees of predicting the path of propagation. Any node in the P2P network has to provide the resources needed to guarantee query propagations and response routing, which in turn means that most of the time the node acts as a relayer of information that may be of no interest to the node itself. This results in inefficiencies and large overhead especially as the nodes become more numerous and connectivity increases. Furthermore, there is no guarantee that a request will spread across the entire network, therefore there is no guarantee to find the providers of a service.

Because of their respective advantages and disadvantages P2P systems and centralized registries strike different trade-offs that make them appropriate in different situations. P2P systems are more appropriate in dynamic environments such as ubiquitous computing, while centralized registries are more appropriate in static environments where information is persistent.

In this paper, we explore a P2P approach to Web service discovery that relies on the Gnutella P2P protocol [3] and uses DAML-S [6] as service description language. Gnutella is a pure widely used P2P network principally for file sharing that does not rely on any centralized registry. DAML-S is a language for the description of Web Services that attempts to

[1] Message propagation is usually bound by a Time To Live (TTL) that limits the distance a message can travel.

bridge the gap between the growing infrastructure of Web Services based essentially on WSDL [2], UDDI [18], and BPEL4WS [4], and the Semantic Web [1]. Previous work on matchmaking using DAML-S described how to use DAML-S for capability matching among Web services [13] and how to apply such a matching process to empower a centralized registry such as UDDI with semantic capability matching for Web services [14]. The work presented here expands on those works by showing how DAML-S can also be used to perform capability based search in a P2P network.

The rest of the paper is organized as follows: first we provide basic background on the Gnutella protocol and DAML-S capability matching. We will then show how to exploit the Gnutella protocol for Web services discovery; in addition we provide a description of a Web service peer on the Gnutella network. Finally, we conclude with a brief literature review, discussion and future work.

II. Gnutella

Gnutella is both a file sharing mechanism and an asynchronous message passing system that allows users to locate and share files across the Internet. Each Gnutella node (servent) acts as both a ‘‘SERVer” and a ‘‘cliENT”. Gnutella servents use their message passing system to perform two types of operations. First, they exchange messages with other servents that are available on the network so that they can maintain, or increase their level of connectivity to the overall Gnutella Network. Secondly, they exchange messages to search for specific files that might be available from other servents. This messaging system is primarily composed of binary packets of information, and text strings that represent search requests. File exchange is based on the HTTP protocol, and uses the same mechanisms used in the retrieval of content from web servers.

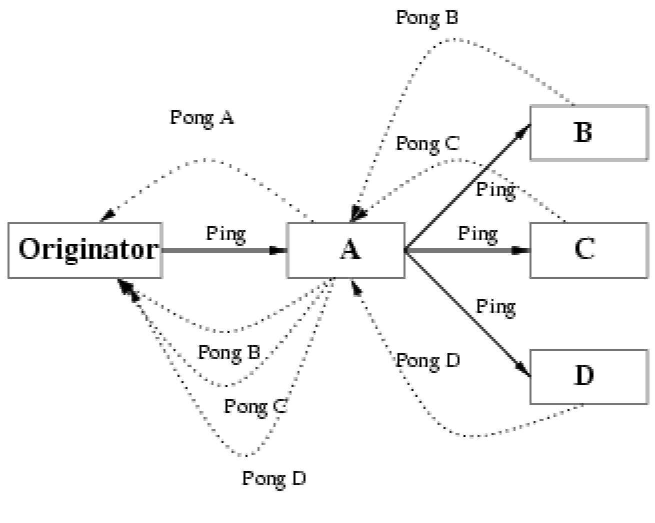

Figure1: Propagation of Ping and Pong messages

In order to discover other servents in the Gnutella Network, Servents use a PING/PONG process. PING messages are sent in hopes of receiving PONG messages that contain host, port, number of files, and kilobytes shared from other servents on the Gnutella Network. As shown in Figure 1, each servent that receives a PING performs two operations: first it sends a PONG back along the same path from which the message came, so that eventually the PONG will reach the originating servent; second it forwards the PING to other servents with a reduced Time to Live (TTL). As soon as the TTL reduces to 0, the message is no longer forwarded and it ceases to propagate. Because of the high degree of connectivity between servents on the Gnutella Network, a PING may hit up to an exponential number of servents in its travels from servent to servent

The search mechanism of Gnutella uses the same message passing process utilized to PING other servents. A QUERY message that is sent to the Gnutella Network contains a number representing the minimum acceptable communications link speed for file downloads, and a string representing the content that is being sought. In typical Gnutella servents, the search string will be tokenized before a servent’s local file system searches for filenames that match any of the string’s keyword tokens. If a local file exists that matches one or more of the words in the query string, its information will be formed into a response to the QUERY packet. If more than one file matches a pattern, the servent can reply with multiple responses embedded in the same message. The HIT message that is sent back contains information about the system’s link speed and the name and size of each matching file. It also contains an integer index value to help map the request into the local file system’s storage.

III. DAML-S

DAML-S defines a DAML [5] ontology for the description of Web Services. A Web Service has a *Service Profile*, a *Process Model* and a *Grounding*. The Service Profile describes what the service does, i.e. the services functionality. For example, Amazon provides browsing of book data bases, provides selling of books etc. The Process model provides a description of the workflow of the service, i.e. the steps through which the service accomplishes its functionality. The Process model in addition, provides the inputs, outputs, preconditions and effects that are required for proper interaction of a service requester with the service provider. Finally, the Grounding provides a mapping of the interactions between the requester and provider to actual message exchange patterns.

In this paper we concentrate on the Profile module of DAML-S that provides capability information which is used during the discovery process. DAML-S describes capabilities of Web Services by the inputs they require, the outputs they produce, the pre-conditions that must hold for the service to take effect and the post-conditions, i.e. the effects that executing the service will have. For example, the inputs to a book selling service could be the ISBN number of the desired book, and a credit card number; the precondition is the existence of enough amount of money in the credit card account. The output of the service is an invoice, and the post condition the sending of a book to the book purchaser. In addition, the Profile describes contact information and accessibility conditions for the service (e.g. only employees of the US Federal Government can access the service), and functional parameters, i.e. parameters describing service quality, such as accessibility, reliability, etc. As another example, consider a travel booking Web Service. Travel booking services usually require departure and arrival information as inputs and produces a fight schedule and a confirmation number as output. The effects of the Web service are the booking a flight, the generation a ticket, and charges to the credit card.

While DAML-S is just a Web Services representation and therefore does not imply any form of processing, it is relatively easy to implement a matching algorithm to recognize which Web Services advertisements match a given request. There is at least one such matching engine [13] that takes advantage of the underlying DAML logic to infer the logic relations between the input and outputs of the request, with the input and outputs of the advertisements. While a complete description of this algorithm is outside the scope of this paper, the main idea is that the outputs of the request should be subsumed by the outputs of the selected advertisements, this condition guarantees that the selected Web Services provide the expected information. Furthermore, the matching engine ranks the advertisements on the basis of their input matching, where, inputs match if inputs of the request subsume the inputs of the advertisement. This condition selects services that the requester can invoke since the requester and provider inputs and outputs (partially) match.

IV. P2P DAML-S MATCHING

The core of our proposal is to combine the DAML-S matching with the Gnutella QUERY process and use the basic Gnutella protocol for Web services discovery. Our proposal is based on A2A [10] which describes how to locate basic MultiAgent infrastructure components on the Internet using the Gnutella P2P network. A2A exploits Gnutella connectivity schema that allows its servents to discover other servents over wide area networks. By enabling Agents and infrastructure components of the RETSINA Multi Agent System (MAS) [17] to act as servents on the Gnutella Network, A2A takes advantage of a fabric of wide-area connectivity that is already in existence and widely deployed. The result is that whenever an agent needs to locate a service provider, it sends a QUERY request through the Gnutella Network. As QUERY requests fan-out over the Gnutella Network, being sent from servent to servent like any file request A2A servents providing RETSINA infrastructure functionality recognize a request for a service that they provide, and reply with HIT messages. However, these HIT responses do not provide information about where to find a file, but the address where service providers can be reached. The complete protocol is described in figure 2

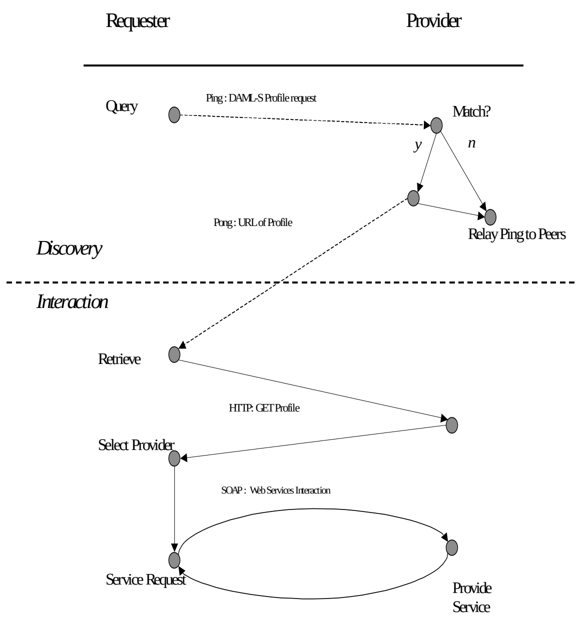

Figure 2. Protocol of Web services discovery and interaction

Web services that adopt the Gnutella based protocol proposed above should be able to verify whether they can satisfy the functionality that they receive as well as managing their interaction with their requesters and providers. In the Semantic P2P architecture we are implementing, this means that every node should contain DAML-S description of its capabilities and the associated engines for parsing ontologies, as well as a P2P discovery module. The resulting architecture is shown in figure 3. The architecture is composed of three modules that are activated in sequence. The first module is a *DAML parser*, based on the Jena parser [11], that reads DAML ontologies and DAML-S specifications off the Web, translates them in a set of predicates, and passes them to the *DAML-S Processor*. The DAML-S Processor is based on the JESS theorem prover [8]; which is used to implement a DAML inference engine [9] and the DAML-S semantics. The last layer of the architecture defines the ports that the Web service uses to interact with the rest of the World. In our architecture the Web service has two ports, one to manage *Webservice Invocation* and interaction with other Web services, the other to interact with the P2P world and perform *P2P Webservice Discovery*. Finally, the DAML-S Processor interacts with the *Application* that decides which Web services to look for on the P2P network, and the information to be exchanged during the interaction with other Web services.

Figure 3 also shows the different roles that DAML-S rules play in the architecture. Rules for DAML-S Process model and Grounding are used to control the

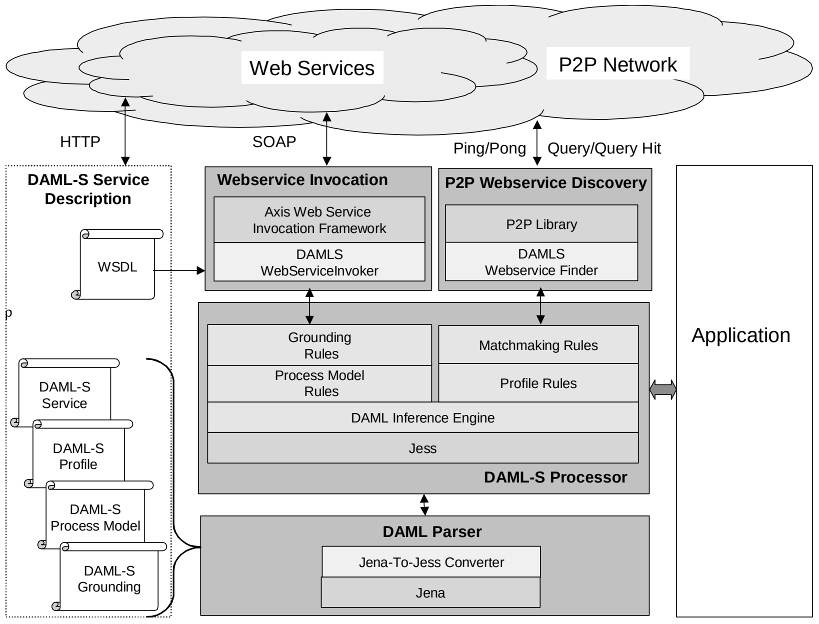

Figure 3. Architecture of P2P DAML-S based Web service

interaction with other Web services, while rules for Profile and Matchmaking are used to manage the discovery and location of providers. When the Application decides to look for a provider with a given functionality ϕ it asks the DAML-S Processor to generate a request ρ for it and broadcast a query for ρ on the P2P network. When DAML-S Web services that act as servents on the Gnutella network receive the query, they attempt to match ρ with their own capabilities using their own matching rules. If a match is detected they respond with a reply signaling to the original requester that they are potential providers. Upon receiving the replies the P2P module of the requester asks the Application to select a provider and initiates the interaction using the provider's Process Model and Grounding specifications.

V. Discussion

In this paper we outlined how Web services can combine the discovery process provided by P2P networks and specifically by Gnutella with the DAML-S representation of Web services capabilities exploiting the semantics of DAML ontologies to provide a capability matching. The result of this work is that Web services that use DAML-S can enter a P2P network such as Gnutella as peers participating not only in distributing Pings and Pongs or Queries and Replies, but also discovering providers of the services they seek or requesters of the services that they provide.

The idea of using P2P for discovery of Web services has already been explored by in the HyperCup project [15]. The goal of HyperCup is to develop an overlaying structure on the P2P network that allows efficient discovery while reducing the overhead related with unbounded ping/pongs and query/reply that is characteristic of Gnutella. Unfortunately, HyperCup reduces the P2P graph to a tree, which on the one hand guarantees that each node is pinged at most once, but on the other hand introduces weaknesses that P2P wants to remove: the failure of one node prevents the visibility of the rest of the tree. Discovery in HyperCup is performed by classifying nodes in the P2P network with concepts in service ontologies. For example, ontologies can represent concepts such as *Buying services* or *Selling services*, then use these ontologies to classify nodes in the P2P network so that through the ontology we can find *Buying Web services* or *Selling Web services.* Unfortunately service ontologies are hard to come by since they have to provide a concept for each type of function, and ultimately they straightjacket different services under the same concept. The approach followed in this paper is to provide a schema for representing any service and an inference mechanism that maps between representations.

Edutella [12] is a project whose goal is to apply semantic web technology to P2P network. The major concern of Edutella is the semantic discovery of contents, not web services. In addition to, Edutella uses their own RDF-based data structure (ECDM) for describing meta data. So the usage of meta data is limited, compared to ontology approach like DAML/DAML-S.

In this paper we assumed the use of the initial Gnutella protocol [3] which defined a flat P2P network in which every node participates in the message passing. Since then, work on the Gnutella protocol has recognized the need to introduce a structure to the network, and developed a new protocol in which some nodes, called Ultrapeers [16], assume the load of the connectivity of the network filtering messages for other nodes. The use of Ultrapeers does not invalidate our proposal since it does not modify the discovery functionalities used in this paper. Indeed, our architecture, and implementation, can easily be abstracted to any P2P protocol (including HyperCup).

Performance is a major concern of the architecture we proposed especially because it makes every node in the P2P network performs the work of a registry. It is easy to imagine situations in which the Web service would be swamped by the amounts of requests for any type of service. We are currently evaluating trade offs of effectiveness vs. cost in our implementation.

Bibliography

[1] T. Berners-Lee, J. Hendler, and O. Lassila.: *The semantic web*.: Scientific American, 284(5):34--43, 2001.

[2] E. Christensen, F. Curbera, G. Meredith, and S.Weerawarana.: *Web Services Description Language (WSDL)*: http://www.w3.org/TR/2001/NOTE-wsdl-20010315 2001.

[3] Clip2***:*** *The Gnutella Protocol Specification* ***V.0.4****:* www.clip2.com/GnutellaProtocol04.pdf

[4] F. Curbera, Y. Goland, J. Klein, Microsoft, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana: *Business Process Execution Language for Web Services, Version 1.0*: http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[5] DAML Joint Committee.: *Daml+oil language (march 2001)*: http://www.daml.org/2001/03/daml+oil-index.html 2001

[6] DAML-S Coalition.: *Daml-s: Web service description for the semantic web*: In ISWC2002.

[7] The Foundation for Physical Agents (FIPA): *FIPA ACL*: http://www.fipa.org

[8] E. Friedman-Hill: *Jess: The rule engine for the Java Platform*: http://herzberg.ca.sandia.gov/jess/

[9] Joe Kopena: *DAMLJessKB*: http://plan.mcs.drexel.edu/projects/legorobots/design/software/DAMLJessKB/

[10] Langley, B., Paolucci, M., and Sycara, K., *Discovery of Infrastructure in Multi-Agent Systems*: In Agents 2001 Workshop on Infrastructure for Agents, MAS, and Scalable MAS

[11] B. McBride: *Jena Semantic Web Toolkit*: http://www.hpl.hp.com/semweb/jena.htm

[12] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, T. Risch: *EDUTELLA: A P2P Networking Infrastructure Based on RDF*: WWW11

[13] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara.: *Semantic matching of web services capabilities*. In ISWC2002, 2002.

[14] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara.: *Importing the semantic web in uddi*. In Proceedings of E-Services and the Semantic Web Workshop, 2002.

[15] M. Schlosser, M. Sintek, S. Decker, W. Nejdl: A Scalable and Ontology-based P2P Infrastructure for Semantic Web Services: P2P2002 - The Second IEEE International Conference on Peer-to-Peer Computing

[16] A. Singla, C. Rohrs: *Ultrapeers: Another Step Towards Gnutella Scalability*: http://rfc-gnutella.sourceforge.net/Proposals/Ultrapeer/Ultrapeers.htm

[17] Sycara, K., Paolucci, M., van Velsen, M. and Giampapa, J., *The RETSINA MAS Infrastructure*. To appear in the special joint issue of *Autonomous Agents and MAS*, Volume 7, Nos. 1 and 2, July, 2003**.**

[18] UDDI: *The UDDI Technical White Paper*.: http://www.uddi.org/ 2000.