FACULTEIT DER NATUURWETENSCHAPPEN, WISKUNDE EN INFORMATICA

Web Services Discovery on Gnutella (decentralized P2P networks)

MSc Thesis in
Grid Computing

Supervisor: Adam Belloum
Assistant: Dmitry Vasyunin

Konstantinos Krommydas
(6/2006)

Abstract

A Web Service can be discovered by clients and applications in a network, by what it does and how it does it. For that purpose, well-defined analytic descriptions of Services and Requests must be available together with a corresponding matching algorithm, incorporated preferably in a totally decentralized communication protocol. A system providing all these features requires some verifications and evaluation. In this thesis we propose a simulation framework which can be used to achieve this goal based on GnutellaSim, we also extend one of the most commonly used semantic matching algorithms using elements from the OWL-s description language, allowing a wider range of matching possibilities.

Table of Contents

1. Introduction……..…….5
   1.1 Motivation…….……..…….....5
   1.2 Research Questions……..…..……...6
   1.3 Goals……..7
   1.4 Outline…….…….……..8
2. State of the Art……...…9
   2.1 Web Services Description, Semantics & Matching……10
      2.2.1 Web Services Description Language (WSDL)…… …… …..……10
      2.1.2 Simple Object Access Protocol (SOAP)…… …… …...……10
      2.1.3 Resource Description Framework (RDF)…… …… ….....11
      2.1.4 Ontologies…….. …….11
      2.1.5 Web Ontology Language (OWL)…….. ……..…12
      2.1.6 OWL-S: Semantic Markup for Web Services…… …...13
      2.1.7 Semantic Matching…… ……....18
   2.2 Peer-to-Peer Networks, Gnutella and W.S. Discovery……21
      2.2.1 Centralized P2P networks & Web Services Discovery……..…….…….....22
      2.2.2 Decentralized P2P networks & Web Services Discovery…….....22
      2.2.3 The Gnutella Protocol v0.4……...……...…….....23
      2.2.4 W.S. Discovery with DAML-S on Gnutella……...……...…….....…….25
   2.3 Network / P2P / Gnutella Simulation…….……..26
      2.3.1 The Network Simulator ns-2……...……...……...……...……26
      2.3.2 GnutellaSim……...……...……...…….....28
      2.3.3 Other Simulators……..……...……...…….....31
      2.3.4 PDNS - Parallel/Distributed NS……...……...……...……..31
3. Design……...33
   3.1 Extended Framework Design……..…….……...33
      3.1.1 Extra Service Attributes to use for Discovery……...……34
      3.1.2 Extending the Matching Algorithm……35
   3.2 Simulated Network's Structure & Content……..……...36
      3.2.1 Ns-2 Structure……..……36
      3.2.2 GnutellaSim Peers Structure……...……...…….....37
      3.2.3 Web Services distribution among Peers……...……...…….....37
   3.3 GnutellaSim modifications for W.S. Discovery……...38
      3.3.1 Operation of the Original Probabilistic File Discovery Simulator……..38
      3.3.2 Design of Changes towards W.S. Discovery Enhancement……...……41
      3.3.3 Operation of the Semantic W.S. Discovery Simulator……..……..43
4. Simulation……..……..47
   4.1 Testing……..……...47
      4.1.1 The Test Ontology & Test-set of Advertisements……...…….....48
      4.1.2 Testing for all Query Hit types……..……..……49
   4.2 Simulation……..…….50
      4.2.1 Setup……..……50
      4.2.2 Performance Evaluation and Comparison to original GnutellaSim.....… ……..……...50
5. Conclusions, Similar & Future Work…….……..…59
   5.1 Conclusions…….……59
   5.2 Comparison with Similar Work……..……60
   5.3 Future Work…….……...61
References……63
Appendix A……..……...65

CHAPTER 1 Introduction

Web Services have the dynamic to transform the internet from a file and information sharing platform to a full-application remote-execution environment. But to search for a web service is much more difficult than searching for a file, because it is also much more complicated to describe services, making descriptions of similar services (or even the same one) semantically different. Also, web services are not necessarily known by their name to the users, as (e.g.) music files do. Thus, the following requirements must be satisfied in order to ease the Web Services dynamic discovery: a) finding a precise and as complete as possible way for describing web services, b) designing a specific network protocol that eases their discovery from other computer nodes in the same network.

1.1 Motivation

It is insufficient to search for Web Services through generic engines: they do not ensure that the results will be services and not merely static web sites. Instead, a dedicated environment, where only services would be offered, is required. But such an environment would need some extra attention on how to interact with the user: He/she is by now accustomed with the file sharing/searching concept, where only a keyword or a short description is enough to characterize a song, a movie, an image or a book. The title of the offered/requested object, its main subject or its classification could quite successfully return some search hits with just a lexicographical matching of meta-data. On the other hand, a Web Service (W.S.) is an "action", an "activity", therefore it does not refer to one static object but to a collection of objects that change "state" (e.g. a selling service causes change of ownership for the sold product). And with such a complex nature, no title and no short description can be objective enough.

As an example, songs have only one title and they mostly belong to only one big music style. But an internet-based department shop (which can have *any* name-title) sells products of many (hundreds of?) genres, so we must not expect all of them to fit somehow in the short meta-data description and, in a second phase, to be found with a simple lexicographical check. Adding to that, how can someone spot immediately a web service which sells *Station Wagons* when the service is only described as a *Vehicle Selling Service?*

Moreover, especially in most commercial cases, the web service provider needs to ensure that his service can be reached by a maximum number of clients. In other words, anyone who wants to use it should be able to reach it easily and quick. This is also the desire of the client himself, plus the fact that he probably wants to have a range of hit answers on his search in order to choose the most appropriate or in case some of them don't work. If we translate these to the physical level, the dedicated environment we mentioned above should be supported by a network (protocol) that allows large popularities of nodes, both of web service providers and clients.

The difficulty with allowing big number of nodes in a network is the complexity of managing them and serving the enormous message-handling load that is produced. This means that we would need a large number of server nodes, a wide distribution of control.

1.2 Research Questions

a) Is there a way to describe Web Services in a more conceptual manner which would allow their searching and matching using methods beyond the typical lexicographical method?

The step beyond the lexicographical idea is to give an objective meaning to the terms that describe the W.S.

Terms start having a meaning when they are related to other terms. One can find several attempts to create a web service semantics description language. The most popular (used in works [2], [3], [5], [6], [11], [12] and [20]) is the XML-type *DARPA Agent Markup Language for Services* (DAML-S, [4]) based on the *DARPA Agent Markup Language with Ontology Inference Layer* (DAML+OIL, [7]). This works on top of the so common *Web Services Description Language* (WSDL, [28]) for empowering its description with semantic terms, which WSDL is lacking in (I/O, pre-conditions, post-conditions, effects of execution, etc. of the service). It organizes these terms in semantic hierarchies called ontologies, where (e.g.) a "parent" node of the output type named "vehicle" has as "children" cars and motorbikes. Recently, DAML+OIL upgraded to the *Web Ontology Language* (OWL, [9]) and so DAML-S to *Web Ontology Language for Services* (OWL-S, [10]).

A step beyond that is the research in [5], where we read about how we can use DAML-S to implement an algorithm for searching web services. They use the input and output entities of each service to match requests and advertisements. If those entities have some close relation in some ontology a hit is considered with gradual ranking: first rank for output and input full match, second for output match and input parent-child relation, then for input match and output parent-child relation etc.

b) Is there a type of network that can host lots of nodes reliably and at the same time perform a satisfactory control on the flow of the messages?

The last 5 years we notice a considerable effort in research, on how to combine web services discovery and peer-to-peer networks. This effort is characterized by two main trends: the use of structured-centralized p2p protocols, or the use of unstructured-decentralized protocols. There are arguments for supporting both cases.

Centralized p2p networks include servers (brokers) that advertise web services with a description and a way to invoke them. The service providers have to register to the network. Their advantage is that it is very easy to perform exact matching with stored keywords, while searching for a web service. On the other hand, such networks are condemned to have limited population of nodes and a low performance on partial matching queries [1]. They also suffer from single-point failure [2].

Decentralized p2p networks have no need of registries and each node has each own index of services on other nodes (the ones it knows) [2]. Thus, they can include a big population of nodes. But, they don't ensure a hit because the distribution of a query is always limited (by lack of time or lack of links to other nodes) [1].

From these two approaches, the second is the newer and appears as more suitable for wide public use of web services. Such public accessibility becomes rather suppressed by "formal" procedures as registering and non-fully-dynamic network structures.

From the few p2p protocols that have an actual absence of any central controllers and brokers is the *Gnutella* [8]. The only network entities in it are the "servent" nodes (server-clients). Those exchange only ping-pong messages for finding neighbors, query messages for searching information, and queryhit messages to respond to successful queries. These queries "die" after some defined *Time-To-Live* (TTL).

1.3 Goals

The design and implementation in paper [2], is based on the algorithm described by paper [5], and considers the use of DAML-S for P2P discovery of Web Services on Gnutella. It requires a DAML-S description coder-reader and a discovery module present on each node. Our research aims to follow the main concept of paper [2], plus using the up-to-date OWL-S instead of the old DAML-S.

We will present the ways to express the partial activities of the services through OWL-S. We will also see how to describe preconditions & effects of the execution of the services, as well as service categories. Up to now we saw that OWL-S is more "compact" than its predecessor DAML-S and allows the extraction of such information easier and more precisely ([12]). This opens the way to extend the matching algorithm presented in [5].

In order to observe the behavior of such a system, we implemented this algorithm and embedded it in a network simulator. The peer-to-peer networks research has provided models ([36], [37]) and architecture designs ([38]), in which the value of simulations, as a validation tool for such systems, is highly pointed. GnutellaSim [17] is a file-sharing gnutella simulator on top of the very popular ns-2 [21]. We modified the core code of GnutellaSim as was appropriate for hosting web services. We chose sample ontologies (including the Kepler Workflow Components ontology [22]) to test the correctness and measure the performance of our new simulator, with advertised web services and service requests whose descriptions use terms that belong to these ontologies.

1.4 Outline

Chapter 2 provides all the necessary background for the achievement of the above goals. What technologies were used and why. Chapter 3 describes the design of a system that fulfills these goals, in a theoretical and partially in an implementation manner. The testing and simulation results of the implemented part are described in Chapter 4. We conclude and compare our work to similar ones and finally propose future directions in Chapter 5.

CHAPTER 2 State of the Art

In this chapter, we begin with presenting the appropriate background for the first research questions in section 1.2. Thus, we examine the progress done in describing Web Services non-semantically (briefly WSDL and the *Simple Object Access Protocol*-SOAP) and the first step towards semantics enhancement on the web, with the *Resource Description Framework* (RDF). Then, we introduce the *Ontologies* and the Ontology Web Language (OWL) which provide the necessary prior knowledge in order to present after them OWL-S for describing Semantic Web Services. This section (2.1) ends referring to the whole semantic matching concept as presented in paper [5].

In the second section we give the background for the second research question of 1.2, the network matter; what is a centralized and a decentralized (especially Gnutella) p2p network and how they can host Web Services. We review also the combination of W.S. Discovery with the Gnutella Protocol ([2]).

Section 2.3 involves with the simulation part of our project. We describe some network simulators of different kinds that could be used or not, briefly, and then we move our attention to *Network Simulator v.2* (ns-2) and *GnutellaSim*.

2.1 Web Services Description, Semantics & Matching

2.1.1 Web Services Description Language (WSDL)

WSDL describes W.S.s as sets of endpoints (*ports*) that communicate through messages, using XML as a basis, and allows interaction with these services. The endpoints are defined with abstract descriptions of operations (*portTypes*) and messages of some format, which follow some network protocol. The mechanism works for all protocols and encodings. It has two main parts: the *abstract definition* of the service and the messages, and the *binding* to the protocol and the message format. The elements of a WSDL document are:

- *Types*: a set of data types to be used in the exchanged messages, mainly in XML-Schema Datatypes (XSD), but extensible to other types also, including OWL classes
- *Message*: an abstract definition of the exchanged data consisting of logical parts. Each part has a message-typing attribute linking to some type (in XSD an *element* (for the name) and a *type*). The message name gives a unique name among all messages, while the part name, a unique name among all the parts of the particular message.
- *Operation*: an abstract description of an action executed by the service. It can be a *One-way* (the endpoint receives a message), *Request-response* (the endpoint receives a message and sends a correlated message), *Solicit-response* (the endpoint sends a message and receives a correlated message), *Notification* (the endpoint sends a message).
- *PortType*: an abstract set of operations corresponding to one or more endpoints, including the used messages.
- *Binding*: a network protocol and data format specification aimed for a particular port type
- *Port*: an endpoint as a combination of a binding and a network address
- *Service*: a collection of related endpoints [28].

2.1.2 Simple Object Access Protocol (SOAP)

SOAP is an XML-based message exchange mechanism for Web Services. The W.S.s are considered as endpoints and a one-way communication is the simplest case covered by SOAP. Multiple communication and the actions to be executed from the endpoint when receiving a message can also be defined with more complex structures. SOAP has been adopted by many W.S. standards and is being used by WSDL for binding the abstract descriptions with concrete network protocols [29].

It operates on HTTP and can be also characterized as a protocol for accessing Web Services. It is platform and language independent, extensible, firewall "breaking" and a W3C standard since 2000 (proposed by UserLand, Ariba, Commerce One, Compaq,

Developmentor, HP, IBM, IONA, Lotus, Microsoft, and SAP). SOAP methods are HTTP request/responses written in SOAP rules. The elements in a SOAP message are:

- *Envelope,* which declares that the XML document is a SOAP message
- *Header,* for application specific information (e.g. authentication)
- *Body,* which includes the call and response information, the actual message content
- *Fault*, for describing errors that may occur during the message transfer [45]

2.1.3 Resource Description Framework (RDF)

RDF was written to describe resources that are available among the internet by modelling abstract concepts in a data format. It is a part of the W3C Semantic Web project (www.w3.org/2001/sw/) for the automation of use and exchange of machine-readable information.

There is an XML syntax available for RDF which enables the creation of *RDF graph triples*. Those *triples* define the resources in a *subject-predicate-object* scheme (e.g. "the input"–"has datatype"–"integer"). The *predicates* are a resource's features and express a relation between the *subject* and the *object* (which are *classes*). The resources may be divided into groups called *Classes,* which can be identified using Uniform Resource Identifiers (URIs) and described with *properties*. RDF *properties* are attributes of resources and correspond to traditional attribute-value pairs, or represent relationships between resources. *Resource* is the superclass of all the objects in RDF. *Literals* and *datatypes* are also used.

All RDF objects are expressed in RDF Vocabularies which can be defined with the RDF Schema. This Schema allows the instantiation of classes and properties that may be further used in order to describe classes, properties and other resources. This is what actually enables RDF with semantics, providing (e.g.) properties with a *domain* and a *range* attribute. Classes can have *subclasses* and properties *subproperties* [46, 47].

2.1.4 Ontologies

Ontologies define the terms that describe a specific domain of knowledge and the hierarchical relations that exist between them (*see fig.1*). In an IT sense, they encode this information in a computer-usable manner, so that it can be also extended to other domains (re-usability) [23].

They can be simple unstructured hierarchies or more complicated ones. For the semantic web, because of the given complexity of the domain, they should include 3 concepts: *Classes* (entities), *Relationships* and *Properties* (of the entities). Logic languages exist for expressing ontologies precisely and thus can be used to create an

artificial intelligence basis for executing semantic search and retrieval, decision support, speech and natural language understanding, knowledge management, intelligent databases, and electronic commerce.

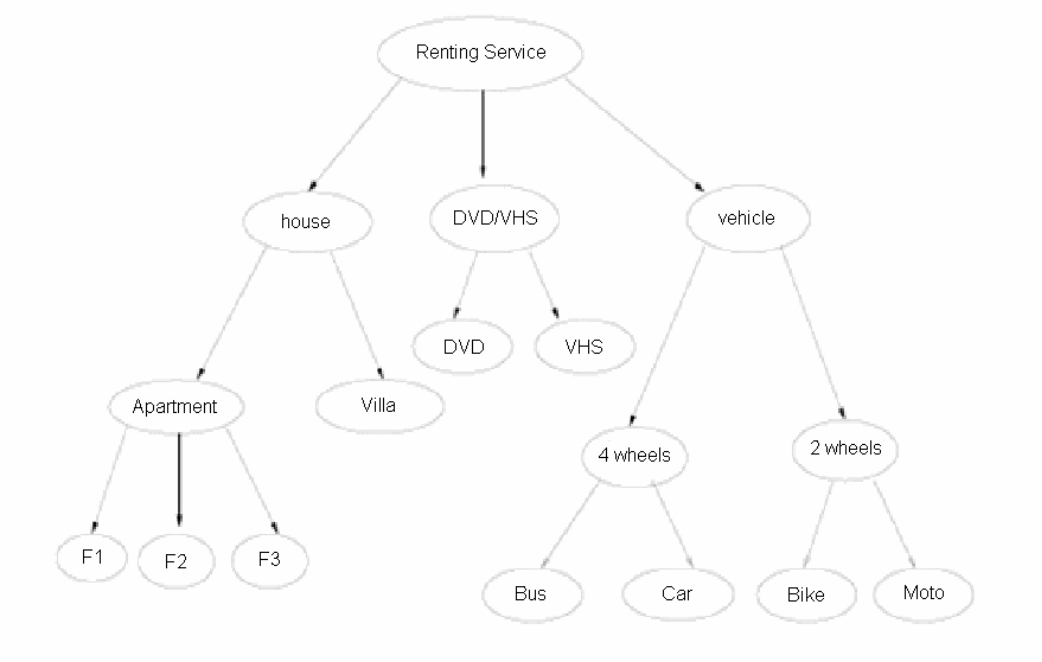

Fig.1 A Sample Ontology [3]

Ontologies give meaning to the meta-data and so go beyond XML and XML Schemas that just contain them without any semantic concept and where *"The same term may be used with (sometimes subtle) different meaning in different contexts, and different terms may be used for items that have the same meaning"* [23]. RDF and RDF Schemas define a class hierarchy with class properties but lack deeper ontology relationships, like class disjointness, cardinality, equality and enumeration of classes, staying at a more abstract approach of the semantics [9].

2.1.5 Web Ontology Language (OWL)

"OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web [26]*". "The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans* [9]*".* It allows distributed collection of information, providing relations between ontologies and explicit importing of ontologies from other ontologies [25]. It extends the vocabulary of XML, RDF and RDF-S with semantic concepts and contains also the additional class relations we mentioned at the end of 2.1.4, plus extra property types [9]. An OWL ontology is an RDF graph, a set of RDF triples [26].

Initial components of the ontology are XML declarations in RDF tags to define the namespaces with URIs, where also the XML, RDF, RDF-S and XSD datatypes are being referred [25]. The rest of an OWL document parts from class axioms, property

axioms and facts about individuals. Ontology version information is also kept, for the discrimination and reusability of older versions. Classes can be defined with:

1. a class identifier (a URI reference)
2. an enumeration of individuals that together form the class instances
3. a property restriction
4. the intersection of two or more class descriptions
5. the union of two or more class descriptions
6. the complement of a class description

Properties can be Object or Datatype properties [26]. Three versions of OWL are supplied, which with an increasing level of functionality are:

- OWL Lite supports only cardinality of up to first degree between classes and is considered enough for simple taxonomies. More precisely, it contains the RDF Schema Features for *classes, subclasses, properties, subproperties, domains* and *ranges,* enriched with OWL features for : *equivalence, similarity, distinction* and *intersection* of classes, and *equivalence, inversion, transition, symmetry* and *enumeration* of properties.
- OWL DL acquires all the language constructs with some case restrictions, but is compatible with description logics (see [24]) and thus can be supported by reasoning software. Analytically it extends OWL Lite with the axioms for *selection, disjointness, union, complement* and *data range* of classes.
- OWL Full supports all RDF structures (being more "open"), but for this reason cannot guarantee reasoning, so it remains at a theoretical-representational level [9].

Important features were added to OWL, which were missing from its predecessor DAML+OIL, fact that persuaded us to prefer the new version. These features include *symmetry* and *data range* for the present properties, while *Ontology Properties* are defined as well. Also, a significant number of functionally redundant components were removed, resulting to a more compact scheme [26].

OWL also allows the deleting of inconsistencies between ontologies and instances of classes. Moreover, supports ontology interoperability, by providing ways to relate concepts from different terminologies [23].

2.1.6 OWL-S: Semantic Markup for Web Services

OWL, as mentioned above, acquires the web content with very flexible semantic XML/RDF-based descriptions. However, as we concluded in the introduction, the complex nature of Web Services demands "special treatment", meaning apparently the