Grid-based Semantic Web Service Discovery Model with QoS Constraints

Kaijun Ren[1,3], Jinjun Chen[2],Tao Chen[1],Junqiang Song[1] ,Nong Xiao[1]
[1]*College of Computer, National University of Defense Technology, Changsha, Hunan 410073，P.R. China*
[3]*College of Science, National University of Defense Technology, Changsha, Hunan 410073, P.R. China*
renkaijun@nudt.edu.cn
[2]*Centre for Information Technology Research, Swinburne University of Technology, Melbourne 3122, Australia*
jchen@ict.swin.edu.au

Abstract

In this paper, we present a model for grid-based semantic service discovery by combining the expressive power of the present ontology language and the advantages of semantic web techniques. This model is an initial work to support dynamic workflow composition of semantically described services. Our model distinguishes other service discovery models by not only supporting semantic functional matchmaking, but also supporting flexible QoS ranking that differentiates similar services offered by different service providers in grid environments.

1. Introduction

In this paper, we presented a model for grid-based semantic service discovery with QoS constraints by combining grid technologies and the advantages of semantic web techniques. This model is an initial and necessary work to support automating the discovery, selection, and workflow composition of semantically described heterogeneous services, which offers the possibility of facilitating geographically distributed scientists to resolve complex scientific problems cooperately. Our model has the following important features which distinguish them from other work in this area.

- Efficiently cooperating with grid components, such as grid security, transaction mechanism, grid monitor;
- Presenting an efficient QoS computing model by cooperating OWL QoS ontologies to meet non-functional requirements;
- Including QoS ranking besides semantic matchmaking capability in service discovery engine based on the user-specified preference.

2. Semantic Functional Matchmaking

We extend the semantic matchmaking algorithm presented in [1, 2]. When a request is submitted, the request description is firstly transformed to OWL-S profile specification. Semantic annotation parse module will parse the request specification and split the related concepts and properties, such as service category, input parameters, output parameters. The referred service categories can do good to limit the scope of service search. These parsed concepts will be grouped request input list vector, and request ouput list vector named reqinputlist and reqoutputlist respectively. The reqoutputlist matching are firstly executed by the matchmaking algorithm, this is because the final matching result will fail if the request output matching fail. Then the reqinputlist matching will be executed if the reqoutputlist matching returns a valid degree of match. Our extended matchmaking algorithm will efficiently cooperate with the OWL inference engine and semantic UDDI API. OWL inference engine based on JESS(a rule-based engine)[3] is used to reason the relationship between the concepts stored in the specific domain semantic model, which the degree of match between two outputs or two inputs depends on. Semantic UDDI APIs are used to retrieve information from UDDI, which interact with the interfaces of OWL parser and OWL inference engine. Part of the matchmaking algorithm is given as follows:

```
public int outputMatch(Vector reqOutputsList , Vector
      advOutputsList){
 int globaldegree=3;//reflect the degree of global //semantic match. 3
is the largest match number
  for each i in reqOutputsList {
    int gooddegree=0;//reflect the local semantic //match.
   for each j in advOutputsList {
       int similarity=conceptmatch(i,j)//similarity //denotes the
      semantic distance in ontology //tree
     if(similarity>gooddegree)
        gooddegree=similarity;
      …
    }
If(globaldegree>distance_to_regular(gooddegree))
 //the semantic distance mapped to the semantic //match
 globaldegree= distance_to_regular(gooddegree);
 }
 …
 return globaldegree;
}
```

0-7695-3007-9/07 $25.00 © 2007 IEEE
DOI 10.1109/SKG.2007.118

In the above matching algorithm, the global degree of match is simplified into 4 levels (3, 2, 1, 0), namely Exact, Plug-in, Subsumption, Fail, and 3 denotes the highest level (Exact). Their formal definitions are as follows:

Exact(3): Service S exactly matches request R $\forall a_{input} \in$ advInputList of S, $\exists\, b_{input} \in$ reqInputList of R, st. $a_{input} \triangleq b_{input}$, here the symbol $\triangleq$ denotes terminological concept equivalence. Simultaneously, $\forall c_{output} \in$ reqOutputList of R, $\exists d_{output} \in$ advOutputList of S, st. $c_{output} \triangleq d_{output}$. The service I/O perfectly matches with the request with respect to logic-based equivalence of their formal semantics.

Plug-in(2): Service S plugs into request R $\forall a_{input} \in$ advInputList of S, $\exists\, b_{input} \in$ reqInputList of R, st. $a_{input} \succ b_{input}$, here the symbol $\succ$ denotes terminological concept subsumption. Simultaneously, $\forall c_{output} \in$ reqOutputList of R, $\exists d_{output} \in$ advOutputList of S, st. $c_{output} \ni d_{output}$, here the symbol $\ni$ denotes d_{output} is the direct child(immediate sub-concept) of c_{output} in concept tree. Service S is expected to return more specific output data whose logically defined semantics is exactly the same or very close to what has been requested by the user.

Subsumes(1): Request R subsumes service S $\forall a_{input} \in$ advInputList of S, $\exists\, b_{input} \in$ reqInputList of R, st. $a_{input} \succ b_{input}$, simultaneously, $\forall c_{output} \in$ reqOutputList of R, $\exists d_{output} \in$ advOutputList of S, st. $c_{output} \succ d_{output}$, This match is weaker than the plug-in and the returned output is more specific than requested by the user, so it relaxes the constraint of immediate output concept subsumption.

Fail(0): Service S does not match with request R according to any of the above rule.

In the above matchmaking algorithm, the set of services which fall under Fail category do not match the service request (in terms of functional aspects), we ignore them for the rest of the service selection process and only consider the services which belong to Exact, Plug-in and Subsumption categories. The algorithm also assumes the worst case scenario. The function of conceptmatch(i,j) reflects the semantic distance between concept i and j in the same ontology tree, and another function distance_to_regular(gooddegree) maps the semantic distance to the global degree of match. Actually, the all matching results (inputs, outputs, etc.) should be taken into account together so that the final semantic matching result can be concluded.

3. QoS Computing Model

Semantic service matchmaking can find functional similar grid services by making use of inferring capabilities of OWL ontologies. However, with the increasing number of functional similar services, semantic matchmaking has no capability of selecting the best service to meet the requirements of user while ensuring the quality of service. Therefore, only semantic ranking is not enough, and other nonfunctional properties of services such as price, reputation and reliability should be computed and ranked. Unfortunately, although QoS-based service selection and ranking have been a hot topic research area[4-7],it's hard to come up with a standard QoS model that can be used for all services in all domains. This is because QoS is a broad concept that can encompass a number of context-dependent nonfunctional properties. Moreover, when evaluating QoS of web services, we should also take into consideration domain specific criteria[5]. Since QoS computing and evaluating become very important in the presence of multiple grid services with overlapping or identical functionality, our semantic service discovery engine integrates QoS ranking module. In order to facilitate users to evaluate the QoS of grid services, objective QoS criteria to distinguish one service from another is needed. Our QoS computing model is primarily composed of the following three aspects: OWL QoS ontology, QoS information collection, and QoS ranking model.

3.1 OWL QoS ontology

OWL QoS ontology is used to provide a common understanding of QoS parameters and their semantics between providers and consumers by reasoning their properties. The ontology simply consists of basic concepts to define QoS parameters and the relationship between them.

In our QoS model we define a generic set of QoS parameters, which is extensible, and can be customized to include domain-specific QoS parameters. Figure 1 shows the general QoS ontology framework. QoS parameters are divided into four categories: network (bandwidth, delay, and etc.), system(availability, reliability, performance, capacity, and etc.), task(cpu_used_for_task, memory_used_for_task, waiting_time, and etc.), extensions(transaction, security, reputation, and etc.). Service providers and customers can extend QoS parameter types.

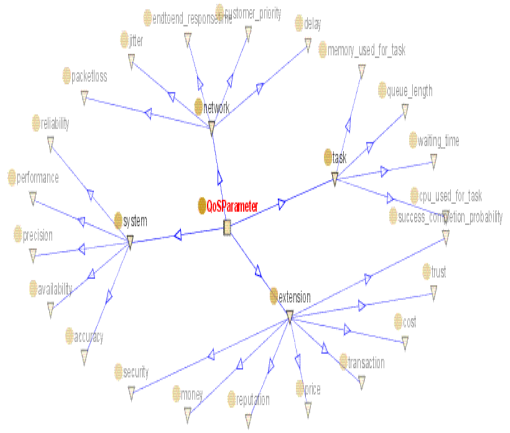

Figure 1. General QoS Ontologies

3.2. QoS Information Collection

In our QoS evaluation model, QoS information is divided into two categories, namely obtained QoS information and computed QoS information. Obtained QoS information such as cpu_capcity, memory_capcity, scalability, availability, cost, response_time, network_bandwith, number_of_processes generally can be captured directly from the provider side, the client side, and network which services depend on. Obtained QoS information needs to be updated whenever they change. Computed QoS information is the information which needs to be computed based on the obtained QoS information. Here we give the following three examples about computed QoS parameters:

Reputation: the reputation of a service s is a measure of its trustworthiness. It mainly depends on end user's experiences of using the service s. Different end users may have different opinions on the same service. The value of the reputation is defined as the average ranking given to the service by end users, i.e., $q_{reputation}(s_j) = \sum_{i=1}^{n} rank(c_i) \Big/ n$, where $rank(c_i)$ is the jth end user's ranking on a service's reputation, n is the number of times the service has been graded.

Average_response_time: the average response time of a service s is computed by dividing the summation of the response times of s submitted by different consumers by the actual throughput of s, i.e., $q_{avr\_restime}(s_j) = \sum_{i=1}^{n} time(c_{ij}) \Big/ n$,where $time(c_{ij})$ is the response time of s_j reported by consumer c_i and n is the number of times the service has been invoked.

QoS information can be collected from the following sources, service providers, user's feedback and active grid execution monitoring.

3.3. QoS Computing and Ranking

To allow for a uniform measurement of service qualities independent of units, the matrix Q needs to be normalized. Before normalizing matrix Q, QoS criteria need to be divided into two groups. The first group g_1 are for the case where the increase of $q_{i,j}$ ($q_{i,j} \in Q$) benefits the service requester, for example, g_1 =(reliability , reputation). Conversely, the second group g_2 are for the case where the decrease of $q_{i,j}$ benefits the service requester, i.e. g_2 =(response time). The following equation 2 is the uniform measurement.

$$q'_{i,j} = \begin{cases} \dfrac{q_{i,j}}{\dfrac{1}{n}\sum_{i=1}^{n} q_{i,j}} & j \in g_1, q_{i,j} \in Q \\ \dfrac{\dfrac{1}{n}\sum_{i=1}^{n} q_{i,j}}{q_{i,j}} & j \in g_2, q_{i,j} \in Q \end{cases} \quad (1)$$

The correction of the equation 2 is proved as follows:

Proof: $\forall j$,if $j \in g_1$, $q_{i,j} > q_{i',j}$, $q_{i,j}, q_{i',j} \in Q$,then $q'_{i,j} > q'_{i',j}$,conversely, if $j \in g_2$, $q_{i,j} > q_{i',j}$, then $q'_{i,j} < q'_{i',j}$.

The equation 1 shows that each element of the normalized matrix Q will play the same part for the QoS ranking without having to consider the different measurement unit and the different direction (positive and negative) of QoS criteria.

Assuming that s_1, s_2, s_3 denote functional similar services after semantic matchmaking, and $R_s = (3,2,2)$ denotes the semantic match degree of s_1, s_2, s_3. In order to reduce the complexity of the problems, QoS criteria are simplified into the following three criteria: response time, reliability, reputation. Especially, reliability and reputation are divided into ten levels. The vector $q_{1v} = (50,9,8)$ means the quality information of s_1, Similarly, $q_{2v} = (30,8,8)$, and $q_{3v} = (40,7,9)$. We can obtain the following matrix Q.Each row in Q

represents a service s_i, while each column represents one of the QoS criteria (response time, reliability, reputation).

$$Q = \begin{bmatrix} 50 & 9 & 8 \\ 30 & 8 & 8 \\ 40 & 7 & 9 \end{bmatrix} \quad (2)$$

If each element in matrix Q is to be normalized using equation 2, we can obtain the following matrix Q'.

$$Q' = \begin{bmatrix} 0.8 & 1.13 & 0.96 \\ 1.33 & 1 & 0.96 \\ 1 & 0.88 & 1.08 \end{bmatrix} \quad (3)$$

Then, we can get the following total QoS ranking R_q of services. $w_q = (0.3, 0.3, 0.4)$ denotes the weights .

$$R_q = Q' * w_q^{\mathrm{T}} = \begin{bmatrix} 0.8 & 1.13 & 0.96 \\ 1.33 & 1 & 0.96 \\ 1 & 0.88 & 1.08 \end{bmatrix} * \begin{bmatrix} 0.3 \\ 0.3 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.963 \\ 1.083 \\ 0.996 \end{bmatrix} \quad (4)$$

Similar to the matrix Q, we can obtain the following matrix Q'' based on the R_s and R_q .Each row in Q'' represents a service s_i, while each column represents semantic ranking and QoS ranking.

$$Q'' = (R_s^{\mathrm{T}} \quad R_q) = \begin{bmatrix} 3 & 0.963 \\ 2 & 1.083 \\ 2 & 0.996 \end{bmatrix} \quad (5)$$

Due to the different measurement unit, Q'' should also be normalized to the following matrix Q''' by using the equation 1.

$$Q''' = \begin{bmatrix} 1.29 & 0.95 \\ 0.86 & 1.07 \\ 0.86 & 0.98 \end{bmatrix} \quad (6)$$

Finally, when the weights of semantic and QoS are assigned to w_s , i.e. , w_s = (0.7, 0.3), we can get the following final rank R_{total} of s_i (i =1, 2, 3).

$$R_{total} = Q''' * w_s^{\mathrm{T}} = \begin{bmatrix} 1.29 & 0.95 \\ 0.86 & 1.07 \\ 0.86 & 0.98 \end{bmatrix} * \begin{bmatrix} 0.7 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 1.188 \\ 0.923 \\ 0.896 \end{bmatrix} \quad (7)$$

Therefore, this demo shows that s_1 is the best service.

4. Acknowledgement

This research is supported partially by National "973" Foundation Research Plan of China (No.2003CB317008) and National Nature Science Foundation of China (No. 60573135 and No.40505023)

5. References

[1].Massimo Paolucci, K.S., Takahiro Kawamura. Delivering Semantic Web Services in WWW2003. 2003. Budapest, HUNGARY: ACM Press.
[2].Naveen, S. An Efficient Algorithm for OWL-S based Semantic Search in UDDI. in semantic web services and web process composition. 2005. USA.
[3].JESS. Java Expert Systems shell. in http://herzberg.ca.sandia.gov/jess.
[4].Hasan Davulcu, M.K., I.V. Ramakrishnan. CTR-S: A Logic for Specifying Contracts in Semantic Web Services. in WWW2004. 2004. New York, USA: ACM Press.
[5].Yutu Liu, A.H.H.N., Liangzhao Zeng. QoS Computation and Policing in Dynamic Web Service Selection. in WWW2004. 2004. New York, USA: ACM Press.
[6].Ran., S., A Model for Web Sevices Discovery with QoS. ACM SIGecom Exchanges, 2003. **4**(1): p. 1-10.
[7].Laila Taher, R.B., and Hazem El Khatib, QoS Information & Computation (QoS-IC) Framework for QoS-Based Discovery of Web Services. MOSAIC, UPGRADE Journal, 2005. **VI**(4): p. 55-66.