Service-based Semantic Search in P2P Systems

Devis Bianchini, Valeria De Antonellis and Michele Melchiori
University of Brescia,
Dipartimento di Elettronica per l’Automazione, via Branze, 38
25123 Brescia - Italy
{bianchin|deantone|melchior}@ing.unibs.it

Abstract—Recently, Semantic Web Services have been increasingly adopted to search, access and manipulate information made available from autonomous and heterogeneous systems interacting in a P2P environment. Platform independency is obtained and the possibility of integrating different information sources over the Web is augmented. In P2P networks, characteristic aspects are the highly dynamicity and the absence of a common resource conceptualization. A centralized structure, where providers publish available services and requesters search for them, constitutes a performance bottleneck and a single point of failure; in P2P environments each provider acts as a peer on the network that supplies its own services and it is able to reply to incoming service requests. A semantic infrastructure to organize and support the automatic location of services without a centralized service registry is required. In this paper we present experimental evaluation of a reference architecture to support service-based semantic search, by means of a distributed service registry, DSR, composed of peers providing semantic-enriched services and semantic links between peers holding similar services; each service provider is also equipped with a service knowledge evolution manager, apt to update peer knowledge, and a semantic search assistant, apt to find services satisfying a user’s requested search.

Keywords—Semantic Web Services; Distributed Service Registry; semantic service discovery

I. INTRODUCTION

An ever-growing amount of information on the Web today is searched, accessed and manipulated through Web Services (e.g., search services for retrieving information about hotels in a given city, ranked by price or number of stars, services for checking the availability of a room and for booking it). The use of Web Services ensures platform independency, while the adoption of Semantic Web technologies makes possible the integration of heterogeneous information sources over the Web. Services are made available on the network by different providers. For efficient and effective service-based search, semantic service descriptions and discovery tools must be provided. A centralized structure, where providers publish available services and requesters search for them, constitutes a performance bottleneck and a single point of failure, only partially mitigated by replicated registries. P2P technologies support large-scale, decentralized applications, where each provider acts as a peer on the network that supplies its own services and it is able to reply to incoming service requests.

In this paper, we stress the experimental evaluation of SERVANT, a reference architecture to support service-based semantic search in P2P systems, that integrates in a comprehensive framework our previous work on service matchmaking [5] and P2P service discovery [6]. In particular, in [5] we experimentally evaluated advanced, semantic-driven service matchmaking techniques on a centralized service registry. In [6] we analyzed the advantages of building and maintaining a semantic overlay on top of P2P networks for efficient service requests propagation.

The SERVANT architecture supports the automatic discovery of services, available on an unstructured P2P network, apt to satisfy user requests for information searches, taking into account the highly dynamicity (networks keep changing their topology and their content) and the absence of a common resource conceptualization as characteristic aspects of P2P networks.

The SERVANT architecture is based on a distributed service registry, DSR, composed of peers providing semantic-enriched services and semantic links between peers holding similar services. Semantic links constitute the semantic overlay built on top of the P2P network. In particular, we organize the DSR on three layers: a logical layer, in which service providers are connected as peers in a P2P network; a semantic layer, where semantic service descriptions are added to the peer and the semantic overlay is maintained; a mapping layer, where mappings between similar services are defined to support service interoperability between peers. Moreover, each service provider is equipped with: (i) a service knowledge evolution manager, apt to update local peer knowledge (when new services are locally added) and network peer knowledge (establishing new semantic links when new peers join the network or new similar services are remotely published); (ii) a semantic search assistant, apt to find services satisfying a user request, to suggest alternative services, to propose related services for possible sequential invocation. A preliminary version of the SERVANT architecture has been presented in [7]. Here we extended the layered structure of the DSR with mapping information and we presented the experimental evaluation of the framework.

This paper is organized as follows: in Section II we present the semantic-enriched distributed service registry; in Section III, we describe the SERVANT semantic search framework; in Section IV experimental results and some implementation issues are discussed, while in Section V a comparison with existing approaches in literature is given. Finally, Section VI closes the paper.
II. SEMANTIC-ENRICHED DISTRIBUTED SERVICE REGISTRY

Figure 1 shows the three-layered structure of the Distributed Service Registry. Service providers supply their services on the P2P network. In Figure 1, three peers are considered, providing services in the travel domain: peers $P_X$ and $P_Y$ supply flight reservation services (AirLow and FlyCheap, respectively), while a third peer, say $P_Z$, provides a hotel booking service (HotelSearching). Each available service is described by means of its functional interface, where we distinguish among inputs, outputs and the categories that classify that kind of service (e.g., hotel booking, flight reservation). Inputs are to be provided by the user to access the underlying resources, while outputs represent information returned by the service. In Figure 1, inputs of the considered services are underlined.

In case of a large availability of services in a distributed environment, to overcome the limitations of a centralized service registry, peers $P_X$ and $P_Y$ can store their services and requesters can look for them, each service provider should be able to autonomously receive incoming requests, compare them against services provided by other services and reply to the service requester; moreover, in case of mismatch or to collect further search results, service providers should be able to forward the requests to the other providers.

Consider the case of Alice, a traveller who wants to organize a week holiday in one of the European cities and is searching for a low-cost flight. She has not chosen the destination yet and her constraints regard the cost of the holiday and the departure and return dates. First, she looks for available low-cost flights and, on the basis of the offers, decides which city to visit. Secondly, she has to find a cheap accommodation for the final destination of the journey. To find an answer, she joins the P2P network and submits her request. Let suppose the request reaches peer $P_X$. The peer $P_X$ replies with the AirLow service to the service request. For an effective comparison between service descriptions, matchmaking techniques based on a semantic description of services are applied. Various approaches addressed the problem of providing a semantic annotation of service functional interface (WSMO [8], OWL-S [9], WSDL-S [1]). In our approach we adopt WSDL-S. Each peer refers to an OWL-DL peer ontology, that provides concepts and semantic relationships used for semantic annotation of service I/O parameters in the WSDL-S document. A suitable tool is available and described in [15] for eased service semantic annotation.

Since we do not constrain nodes in the DSR to adopt a common shared conceptualization, different peer ontologies are used on different nodes. Peer ontologies could refer to the same concept using different terms, that are synonyms or more/less specific. To bridge the gap between slightly different terminologies, peer ontologies are extended with a thesaurus of terms related by terminological relationships to the names of ontological concepts. The following terminological relationships are considered: (i) synonymy (SYN), established between two terms that can be used interchangeably in the domain (e.g., departureCity SYN fromCity); (ii) narrower/broader term relationship (NT/ BT), established between a term $t_1$ and another term $t_2$, where $t_1$ has a more generic (resp., specific) meaning than $t_2$ (e.g., roomType NT roomInformation); (iii) related term relationship (RT), established between two terms whose meaning is related (e.g., Room RT Stars). The thesaurus is built from the WordNet lexical system\(^1\). In this way, it is possible to extend matchmaking capabilities when looking for correspondences between semantic services.

described in terms of different peer ontologies. A detailed explanation of how the thesaurus is built and used in combination with OWL-DL ontologies has been presented in [5].

After comparison of service request and local available semantic service descriptions on peer $P_X$, the request must be forwarded on the P2P network: (i) to suggest other similar services, in the same category, provided on different peers (e.g., service FlyCheap on $P_Y$); (ii) to suggest possible related services (accepting as INPUT what is OUTPUT of the required one), such as hotel booking facilities (e.g., service HotelSearching on $P_Z$). Such a decentralized service discovery approach does not suffer of a performance bottleneck and a single point of failure, as for a centralized service registry. Moreover, the possible storage of a vast number of service advertisements on a central registry hinders the timely update, in particular for dynamic environments such as the P2P one. Finally, centralized architectures may constrain to an agreement on the adopted conceptualization for semantic description of shared resources. On the other hand, flooding of service requests may increase the network overload. Furthermore, there is no guarantee to reach all the suitable services and providers on the network. To mitigate these limitations, besides local knowledge about its own services, each provider maintains network knowledge in terms of semantic links to other peers in the DSR (semantic neighbors), where similar services are registered. Local and network knowledge constitute a semantic overlay built and maintained over the logical P2P network. In particular, we distinguish between functional similarity links (FSL), denoted with $S_1 \approx S_2$, that relate similar semantic service descriptions (e.g., services AirLow and FlyCheap), and coupling similarity links (CSL), denoted with $S_1 \rightarrow S_2$, to assert that outputs of $S_1$ are related to the inputs of $S_2$ (e.g., services AirLow and HotelSearching). Semantic links between semantic service descriptions constitute the semantic layer of the DSR (Figure 1). Furthermore, for each semantic link, service I/O mappings are established as explained in the next section. I/O mappings form the upper layer of the DSR (mapping layer).

The resulting peer architecture is shown in Figure 2. Each peer is provided with a Knowledge Evolution Manager, that is the component in charge of updating the peer local knowledge (when new services are locally provided) and network peer knowledge, by discovering semantic links when peers join the network or services are remotely published. The knowledge collected and organized on each peer is exploited in the Semantic Search Assistant during the service discovery process. The Knowledge Evolution Manager and the Semantic Search Assistant interact with the peer knowledge through the Peer Network Manager, that contains a DOM parser to read the WSDL-Ss of locally available semantic service descriptions and to load and update the Peer Network Knowledge. The communication on the P2P has been implemented through the JXTA API\textsuperscript{2}. An excerpt of the Peer Network Knowledge representation is shown in Figure 3 for peer $P_X$. For each locally available service, different kinds of information are provided: (i) the URL for the service invocation; (ii) the reference to the corresponding WSDL-S document, locally stored; (iii) the reference to service categories, extracted from standard taxonomies (such as UNSPSC\textsuperscript{3}), where coarse-grained concepts are used to denote the service kind (e.g., hotel booking, flight reservation); (iv) the representation of semantic links starting from that service. Semantic links are labeled with matching information and I/O mappings. Details are provided in the next section, where we explain how semantic links are established, described and exploited within the SERVANT architecture.

\textsuperscript{2}https://jxta.dev.java.net/.

\textsuperscript{3}The United Nations Standard Products and Services Code: http://www.unspsc.org/.
III. SEMANTIC SEARCH FRAMEWORK

A. Service matchmaking

In [5] we proposed an hybrid matchmaking model to compare semantic service descriptions based on peer ontologies and terminological knowledge (thesaurus). The rationale behind the application of hybrid techniques in the scenario considered here is that we need two kinds of matching information to label semantic links, in order to support efficient and effective service discovery: (i) a deductive matching model, exploiting equivalence/subsumption checking in the peer ontology, is used for classifying the kind of match between the service request and the service advertisement, to check if the latter truly or partially satisfies the former; (ii) a similarity based model quantifies the degree of overlapping between two semantic service descriptions through the evaluation of name affinity between terms used in the descriptions, on the basis of the terminological knowledge. Specifically, to establish the affinity between names, a weight $\sigma_{rel} \in (0, 1]$ is associated with each kind of relationship in the thesaurus: (i) for synonymy, we pose $\sigma_{SYN} = 1.0$, because synonymy is generally considered a more precise indicator of the affinity between names; (ii) for narrower/broader term relationship, we pose $\sigma_{BT/NT} = 0.8$; (iii) for related term relationship, we pose $\sigma_{RT} = 0.5$. Two generic terms $t_1$ and $t_2$ can be related by one or more chains of relationships: we call path of length $m$ between two terms $t_1$ and $t_2$, denoted with $t_1 \rightarrow^m t_2$, a finite ordered sequence of $m$ relationships. The strength of $t_1 \rightarrow^m t_2$ is the product of the weights of all the relationships belonging to the path, that is, $\rho(t_1 \rightarrow^m t_2) = \prod_{k=1}^{m}(\sigma_k) \in (0, 1]$. Since between two terms there can exist more than one path, the one with the highest strength is chosen. The Name Affinity coefficient between $t_1$ and $t_2$, denoted by $\text{NA}(t_1, t_2)$, is defined as follows:

$$\begin{cases} 
\max_m(\rho(t_1 \rightarrow^m t_2)) & \text{if } t_1 = t_2 \\
\max_m(\rho(t_1 \rightarrow^m t_2)) & \text{if } t_1 \neq t_2 \\
0 & \text{otherwise} 
\end{cases}$$

Only name affinities that are equal or greater than an affinity threshold $\alpha$ are considered, where $\alpha$ is set to filter out not relevant affinity values. The choice of actual values of $\alpha$ and the other thresholds used in the matchmaking process is done during a training phase [5].

Matching techniques are used to: (i) establish a functional similarity link or a coupling similarity link between a local semantic service description and a remote one, as shown in the following; (ii) during the service discovery phase, to find suitable search services for a given request.

Given two semantic service descriptions $S_1$ and $S_2$ to be compared, service categories are initially exploited to filter out not matching services: if there is not at least a category for $S_1$ that is semantically related (through subclass-of or equivalent-to relationships) to a category for $S_2$, or vice versa, the hybrid matchmaking algorithm is not applied and the two semantic service descriptions mismatch.

Deductive matching techniques are applied to classify the kind of match $\text{MatchType}(S_1, S_2)$ between $S_1$ and $S_2$, following general guidelines in the literature [11]. A formal definition of these kinds of match can be found in [6].

- **Exact**, to denote that $S_1$ and $S_2$ have the same capabilities, that is, they have: (i) equivalent output parameters; (ii) equivalent input parameters.
- **Extends**, to denote that $S_2$ offers at least the same capabilities of $S_1$, that is: (i) an equivalent or more specific output parameter in $S_2$ is found for each output parameter of the $S_1$; (ii) an equivalent or more general input parameter in $S_1$ is found for each input parameter of the $S_2$; the inverse kind of match is denoted as **Restricts**; the rationale behind the **Extends** match is that $S_2$ at least fulfills the functionalities of $S_1$ if it provides all the $S_1$ outputs, but, on the other hand, $S_1$ must be able to provide all the inputs needed for the execution of $S_2$.
- **Intersects**, to denote that $S_1$ and $S_2$ have some common capabilities, that is: there exists a pair of I/O parameters, one from $S_1$ and one from $S_2$, that are related in any generalization hierarchy in the peer ontology.
- **Mismatch**, otherwise.

In case of partial match ($S_2 \text{ Restricts} \text{Intersects} S_1$), a similarity-based matching model is used to quantify the degree of match $\text{Sim}_{IO}(S_1, S_2)$ between semantic service descriptions through the following coefficient:

$$\text{Sim}_{IO}(S_1, S_2) = \frac{\sum_{i,j} \text{NA}(i_n, j_n)}{\|\text{IN}(S_1)\| + \|\text{IN}(S_2)\|} + \frac{\sum_{k} \text{NA}(o_{out_k}, o_{out_k})}{\|\text{OUT}(S_1)\| + \|\text{OUT}(S_2)\|} \in [0, 1]$$

where: $i_n \in \text{IN}(S_1)$, $j_n \in \text{IN}(S_2)$, $o_{out_k} \in \text{OUT}(S_1)$, $o_{out_k} \in \text{OUT}(S_2)$. **Exact** and **Extends** matches correspond to the case $\text{Sim}_{IO}(S_1, S_2) = 1.0$, while **Mismatch** corresponds to the case $\text{Sim}_{IO}(S_1, S_2) = 0.0$. $S_1$ and $S_2$ are recognized as matching if the kind of match is not **Mismatch** and $\text{Sim}_{IO}$ is equal or greater than a threshold. The functional similarity link is defined as a 4-uple:

$$\langle S_1, S_2, \text{MatchType}(S_1, S_2), \text{Sim}_{IO}(S_1, S_2) \rangle$$

If a match between $S_1$ and $S_2$ is not recognized, services are analyzed with respect to the degree of interdependency between them. On the basis of the output parameters in the first service that are recognized to have semantic correspondences with some of the input parameters of the second service, their degree of coupling is measured. The evaluation relies on the following coupling coefficient ($\in [0, 1]$):

$$\text{Coupl}_{IO}(S_1, S_2) = 2 \times \frac{\sum_{i,j} \text{NA}(out_i, in_j)}{|\text{OUT}(S_1)| + |\text{IN}(S_2)|}$$

where: $out_i \in \text{OUT}(S_1)$, $in_j \in \text{IN}(S_2)$. $S_1$ and $S_2$ are recognized as coupled if $\text{Coupl}_{IO}(S_1, S_2)$ is equal
or greater than a threshold. The **coupling similarity link** is defined as a 3-uple:

\[
(S_1, S_2, \text{CouplIO}(S_1, S_2))
\]

**Example.** Services \(S_1X\) and \(S_2Y\) in the running example are similar. Indeed, we see that the input SeatPreferences of \(S_2Y\) does not correspond to any input in \(S_1X\). As a consequence exact and extends match conditions are not satisfied (since \(IN(S_2Y) \not\subseteq IN(S_1X)\)). On the other hand, \(S_2Y\) provides outputs FromCity and AirplaneModel that are not provided by \(S_1X\). Semantic correspondences between all the other I/O parameters of the two services are recognized, so \(S_2Y\) \textbf{INTERSECTS} \(S_1X\) (partial match) is established. Similarity-based evaluation of \(S_1X\) and \(S_2Y\) is therefore performed and gives a value \(Sim_{IO}(S_1X, S_2Y) = 0.61\).

**B. Service Knowledge Evolution Manager**

**Network Knowledge Harvesting.** For establishing semantic links, semantic service descriptions in different peers have to be compared by applying service matchmaking techniques. Periodically, the knowledge evolution manager starts a knowledge harvesting process to possibly establishing new semantic links. Since in \textsc{servant} no central coordination mechanisms or structures are defined, each peer is autonomous and has the ability of identifying similar services registered on different nodes by means of network knowledge harvesting. Specifically, at predefined time intervals, a peer \(P_X\) sends a probe request for each locally available service that is not yet source of a semantic link. The probe request contains the semantic description of the service. A peer \(P_Y\), receiving the request, applies the service matchmaking techniques between the probe request and each local semantic service description and, for each similar service, a reply is sent back containing: (i) a reference to the matching service, (ii) the corresponding matching information, that is, the kind of match and the similarity value, and (iii) the semantic correspondences between I/O parameters. \(P_X\) collects received information and establishes a semantic link towards \(P_Y\).

Peer knowledge evolution occurs at the semantic layer through the probe collaboration mechanism, but it is initiated at the P2P logical layer, where, in order to guarantee up-to-date connections between peers an Overlay Management Protocol (OMP) is used, which defines specific procedures to join, leave and modify the logical layer. Given the inner dynamism of P2P networks, a shuffling-based OMP is chosen in order to allow more effective information diffusion among peers [20]. This kind of OMP arranges the logical layer as a graph in which each peer is directly connected to a small portion of the whole peer population, that is, its logical neighbors. The shuffling protocol is quite simple: peer \(P_X\) keeps changing the set of its logical neighbors by randomly contacting one of its current neighbors; then \(P_X\) and the contacted peer exchange some of their logical neighbors. The obtained result is an inexpensive overlay membership management, in the sense that any joining or leaving of peers is quickly and efficiently detected without overloading the network. Moreover, the underlying OMP enables the identification of new semantic links at the semantic layer. This avoids the creation of clusters of peers that are not connected through semantic links to any other peer in the network. Such clusters would constitute pitfalls for any service request reaching one of those isolated peers.

**Service I/O Mappings.** Once a semantic link between a semantic service description \(S_1X\) on peer \(P_X\) and a semantic service description \(S_2Y\) on peer \(P_Y\) has been set, a domain expert (i.e., the peer administrator) is assisted in setting I/O mappings between inputs/outputs of \(S_1X\) and \(S_2Y\). The purpose of building and maintaining I/O mappings is to allow to propagate the invocation of a local service to remote ones transparently and with the minimal involvement for the users. The overall objective is to make it possible to invoke more services at a time to answer a user’s requested search and provide user with the results of these invocations. In particular, mappings are based on I/O semantic correspondences collected and stored by a peer during the Network Knowledge Harvesting phase. Correspondences are proposed to the domain expert, on request, to establish I/O mappings between two linked services. If a similarity link is set between a local and a remote service, I/O mappings can be established among their inputs (respectively, outputs). In particular, I/O mappings among services \(S_1X\) and \(S_2Y\) allow to transform \(S_1X\) input into a valid (possibly partial) input for linked service \(S_2Y\). Similarly, output results produced by the remote service \(S_2Y\) can be mapped (possibly partially) to \(S_1X\) output parameters to be presented to the user according the output schema of local service \(S_1X\).

An I/O semantic correspondence between parameters \(io_{ix}\) of \(S_1X\) and \(io_{iy}\) of \(S_2Y\) is defined as follows:

\[
\langle io_{ix}, io_{iy}, NA(io_{ix}, io_{iy}), \text{dom}(io_{ix}), \text{dom}(io_{iy}) \rangle
\]

where the correspondence is established if \(NA(io_{ix}, io_{iy})\) is greater than a given threshold; \(\text{dom}()\) is the concrete domain of a parameter, i.e., string, date, integer or other XSD type according to the WSDL specification. Correspondences are obtained when matchmaking is applied to \(S_2Y\) and the probe query containing the description of \(S_1X\).

Concrete domains of \(io_{ix}\) and \(io_{iy}\) are evaluated by the service knowledge evolution manager to determine their compatibility.

Compatibility of two built-in primitive (i.e., atomic) XSD domains can be concluded due to the internal representation of values. We refer to this kind of compatibility as basic-level compatibility. For instance, \(float \Rightarrow \text{double}, \text{int} \Rightarrow \text{long}, \text{token} \Rightarrow \text{string}, \text{integer} \Rightarrow \text{string}\), where \(\Rightarrow\) denotes domain implication. If basic-level compatibility is established between \(io_{ix}\) and \(io_{iy}\), a mapping function is built by the system and proposed to the domain expert that can validate it or not. As an example, given the semantic correspondence \(\langle \text{FlexibilityDays}, \text{Tolerance}, 0.6, \text{integer}, \text{string} \rangle\) and \(\text{integer} \Rightarrow \text{string}\), a mapping function is built to convert...
the integer value of *FlexibilityDays* into a string value for *Tolerance*. A mapping function is expressed by an XQuery expression.

Generally speaking, we note that a parameter can be present in zero, one or more I/O semantic correspondences of a pair of services. That is, a parameter of a service can be related to zero, one or more parameters of the linked service. Given a set of I/O semantic correspondences between a local and a remote linked service, the domain expert (DE) is required to manage the following cases:

- **correspondence one-to-one and basic-level compatibility**: the DE should validate the parameter correspondences and the mapping function proposed by the system or define a different mapping function;
- **correspondence one-to-one without basic-level compatibility**: the DE should validate the parameter correspondences and define a suitable one-to-one mapping function or define a different mapping function;
- **correspondence one-to-many**: the DE should validate the parameter correspondences and define a suitable one-to-many mapping function or define a different mapping function.

Since setting I/O mappings from a peer $P_X$ to $P_Y$ requires the intervention of a DE, it is necessary to avoid loosing this kind of information even if the peer $P_Y$ temporarily disconnects or if a remote service on $P_Y$ does not reply to a given number of successive requests and therefore have to be considered as not available. In these cases, the corresponding semantic links with associated mappings are marked as invalid and are not used for request forwarding, but they are not removed from $P_X$. Periodically, $P_X$ will check the status of the remote provider/service and if necessary the status of semantic links is restored to valid.

### C. The Semantic Search Assistant

When a new search $S_R$ is submitted to a peer $P_X$, similar services must be identified throughout the DSR. Service discovery is performed on the basis of semantic links identified by the knowledge evolution manager, to improve the scalability of the proposed system. The service discovery task is performed according to the following steps.

#### Semantic neighbors selection

The search request $S_R$ submitted to the peer $P_X$ is matched against the local semantic service descriptions and a list $MS(S_R)$ of matching service descriptions is obtained. The request $S_R$ is forwarded to the other nodes of the DSR to extend the list of matching services. To prune the set of peers to be investigated, thus avoiding time-consuming distributed service search, semantic neighbors that are related to services $S_iX \in MS(S_R)$ through functional similarity links are selected as request recipients. Moreover, candidate semantic neighbors are filtered according to different forwarding policies, that are based on the matching information labeling the semantic links.

According to a minimal policy, search over the network stops when matching services which fully satisfy the request have been found. Exhaustive policies are applied following the same rules, but the search does not stop when matching services that fully satisfy the request are found: the request $S_R$ is forwarded to semantic neighbors to find other services that could present, for example, better non functional features. In both forwarding policies, if no semantic neighbors have been found, a subset of logical neighbors to which the request is to be forwarded is randomly selected. This further reduces the problem related to the formation of disjoint subgraphs of semantic neighbors related by semantic links, as discussed in Section III-B. In [6] a detailed presentation of different forwarding rules based on functional similarity semantic links is provided. Note that without the organization of services through semantic links, the discovery process would rely on conventional P2P infrastructures and associated routing protocols for query propagation in the network (e.g., flooding). Exploiting the semantic links, it is possible to enforce query forwarding according to content similarities rather than to the mere network topology.

#### FSL-driven request forwarding

Once the candidate semantic neighbors have been selected, the search request $S_R$ is forwarded towards them in order to obtain required search results on the DSR. A token-based strategy is adopted to avoid cycles and network overloading [18]. The strategy is based on similarity degree labeling functional similarity links. First, for each functional similarity link $sl$ starting from a service $S_iX \in MS(S_R)$, the relevance of $sl$ for retrieving the request $S_R$ is evaluated as the harmonic mean combining the similarity $Sim_{IO}^{sl}$ labeling $sl$ with the similarity $Sim_{IO}(S_R,S_iX)$, that is:

$$ r_{sl} = \frac{2 \cdot Sim_{IO}^{sl} \cdot Sim_{IO}(S_R,S_iX)}{Sim_{IO}^{sl} + Sim_{IO}(S_R,S_iX)} \quad (7) $$

A number of tokens $A_P$ is assigned to the peer $P_X$ that has been reached by the service request $S_R$. These tokens are proportionally distributed among selected semantic neighbors, according to their relevance value; the number of tokens assigned to the semantic neighbor $sn$ with relevance value $r_{sn}$ is determined as:

$$ A_{sn} = \left\lfloor \frac{A_P}{\sum_{i=1}^{RSN} r_{sn_i}} \cdot r_{sn} \right\rfloor \quad (8) $$

Each semantic neighbor receiving the request checks for locally available matching services: if matches have been found, the semantic neighbor replies to $P_X$; from which the request started, it consumes a token and, if the remaining number of tokens is not zero, it repeats the forwarding procedure based on its semantic neighbors and the current token value. Search results are collected and ranked according to the similarity value. Since in an unstructured P2P network is not guaranteed how much time would be taken to search for a given item, a time-out is set on $P_X$ to collect the search results.

#### CSL-driven extension of results

Once the user selects one of the retrieved services, the services linked to
the selected one through coupling similarity links are proposed to the user. In the running example, if \( S_1 \rightarrow X \) is selected, the system exploits the semantic link \( (S_1 \rightarrow S_2) \) to propose the \( S_2 \rightarrow X \) service to the user as a service complementary to \( S_1 \rightarrow X \).

**Service invocation.** Once suitable matching services have been identified for a given service request, service invocation is performed on the DSR following P2P mappings. The user can select one of the search results \( S \) and P2P mappings starting from \( S \) are exploited to propagate the request towards other service providers.

**IV. EXPERIMENTAL EVALUATION**

**Implementation and experimental setup.** A prototype of the peer architecture has been implemented in Java on top of the JXTA P2P infrastructure. Moreover, experimental evaluation has been performed by implementing a simulator designed on the basis of the Neurogrid P2P simulator\(^4\) to evaluate effectiveness of the forwarding policies over the semantic service overlay. The simulator operates as follows. Firstly, it generates a P2P network: a set of nodes is instantiated and logical connections are randomly established, given a maximum number of outgoing edges for each node. The number of peers and the maximum number of connections for each node are parameters that can be set in a simulation run. Secondly, services are randomly assigned to nodes in the P2P network. A test set of semantically annotated WSDLs\(^5\) has been considered: it consists of 894 semantic Web services from 7 domains (education, medical care, food, travel, communication, economy, weapon) and 26 service requests. Requests and services are generated as a set of indexes and matching information are pre-computed and stored in a data set. Such a data set is exploited by the simulator to decide when to establish a semantic link, when a service positively matches a request and when to forward a request to another peer. Each index representing a service is then randomly assigned to one or more peers. In other words, matching evaluation in the simulator is performed by accessing the pre-computed results in the data set without invoking the matchmaker. The choice of using precomputed matching results instead of run time computation involving the matchmaker is motivated by our purpose of evaluating the behavior of forwarding policies rather than DSR performances in terms of response time. An experimental evaluation concerning the adopted hybrid matchmaker has been described in [5].

Performed simulations are in particular devoted to measure the generated traffic and the recall of SERVANT. By generated traffic we mean the total number of messages produced and forwarded on the network as a consequence of a request submitted to a peer. The recall is defined with respect to a service request and is measured according to the classical definition of Information Retrieval, that is, the ratio between the number of services actually retrieved by the semantic search assistant with respect to all the services relevant to the request. To establish the number of the services relevant to a request \( R \), the simulator just counts the number of services that match with \( R \) in the pre-computed data set. The experiments have been designed to evaluate: (i) the DSR scalability; (ii) performances of the implemented forwarding mechanism compared with the Gnutella one, that is based on flooding. Advantages derived from the exploitation of semantic links are balanced by the additional traffic generated for establishing the semantic overlay. Experimentation results on such traffic will be analyzed in the following.

**Scalability tests.** SERVANT scalability is analyzed in terms of generated traffic and recall as the number of peers grows. The lower the generated messages, thus reducing the network congestion, the better the scalability. Specifically, the simulation has been performed with the number of peers that varies in the range [100-1000], with increments of 180, the number of maximum connections for each node set to 5, the TTL value, denoting the number of times a message sent to a peer can be forwarded to other peers, set to 5. We report the results for generated traffic in Fig. 4(a) and for recall in Fig. 4(b). We distinguish the results for minimal and exhaustive policies. We can see that, as the number of peers grows, the traffic scales quite linearly. On the other hand, the recall varies in the range [0.66-0.90] for minimal policy and in [0.76-0.99] for exhaustive and decreases smoothly as the number of peers grows.

**Comparison with the Gnutella protocol.** The simulations we have run also compares the implemented forwarding policies with the well known Gnutella P2P protocol both in terms of efficiency and scalability. Actually, Gnutella is oriented to document discovery, but we have implemented in the simulations a service discovery process exploiting the Gnutella forwarding policy, that is based on a flooding strategy relying on logical connections between peers. The choice of a comparison with Gnutella is due to the fact that both SERVANT and Gnutella define an overlay network built on top of an unstructured P2P network. Apart from the architectural similarities, we have considered Gnutella since its message forwarding policy is well-known and it is frequently considered as a reference example.

The purposes of comparison are: (i) demonstrate the better recall of our distributed search with respect to Gnutella search; (ii) confirm that the use of the minimal and exhaustive request forwarding policies results in an improved scalability. In particular, SERVANT and Gnutella service discovery have been compared on the basis of the generated messages and the recall as the value of TTL parameter grows. As TTL increases, the number of peers reached by the request grows, so better recall is expected both for SERVANT and Gnutella. On the other hand, also the traffic increases. The simulation has been performed on a network with 500 peers and a maximum number of logical connections between peers set to 5.

In Fig. 5(b) we report the results of recall evaluation. SERVANT works better than Gnutella, but Gnutella

\(^4\)http://www.neurogrid.net.

\(^5\)http://projects.semwebcentral.org/projects/sawsdl-tc/.
tends to get good performances as TTL becomes high. This is due to the fact that in the considered network configuration the simulation works with 500 peers and about 5 connections for peer. In such a configuration, with TTL = 6, Gnutella is prone to flood the network reaching an high percentage of the peers and thus retrieving most of the services relevant to the request. For what concerns the generated traffic, the results are shown in Fig. 5(a). We note the good results obtained by both the SERVANT forwarding policies with respect to Gnutella. However, generated traffic due to the creation of the semantic overlay must be also considered for SERVANT forwarding policies.

**Semantic overlay setup evaluation.** Beside the presented evaluation on scalability of the approach, the network overhead generated by a peer joining to the DSR must be considered. This overhead is due to the sent probe requests, one for each shared service. Probe messages are forwarded to other peers according to a flooding strategy, since semantic links are not set yet in this phase. To keep low the number of generated messages, probe requests are sent with a low TTL value (2 or 3). If no answer is received, probe requests are sent again with an incremented TTL. In Fig. 6(a) the total number of probe messages generated in the simulations discussed previously are plotted. In Fig. 6(b) is shown how the recall value change as the number of nodes increases, if different TTLs are used for probe messages. With a higher value of probe TTL, it is higher the number of semantic links established by each peer and this fact biases positively the recall.

V. RELATED WORK

Semantic service discovery on P2P architectures has been an important issue for the Semantic Web and several
approaches have dealt with this problem. Semantic-driven service discovery on a P2P network has been addressed in [14], where each peer provides its own service, described using DAML-S service ontology, and the Gnutella protocol is used to forward service requests on the network according to a flooding-based mechanism, thus avoiding the use of a centralized service registry. No semantic links are maintained between peers, while in SERVANT a semantic layer is built on top of the P2P infrastructure.

Other approaches use distributed architectures constituted by several UDDI registries properly organized to make more efficient request propagation. In METEOR-S [19] service descriptions are kept in UDDI Registries semantically enhanced with local domain ontologies, while a centralized registries ontology is adopted to classify peer registries. During the discovery process, registries ontology is browsed to find the proper node to which the request must be forwarded. A three-layer distributed UDDI registry, called ad-UDDI, has been also proposed in [10]. Each ad-UDDI registry is associated to one or more categories from a standard classification. All ad-UDDI registries are registered in a centralized server. Neighboring relationships between two ad-UDDI registries are established if they share at least one category. With respect to these approaches, SERVANT does not rely on a centralized structure for managing registries classification, for adding or removing a registry or for managing neighboring relationships, thus avoiding a system bottleneck and a single point of failure.

Centralized components have been avoided also adopting structured P2P solutions [2], [13], [17], that organize peers or shared services through fixed structures, like DHT (Distributed Hash Tables), which require more maintenance efforts and a common conceptualization or ontology to describe the resources at hand. With respect to these attempts, SERVANT does not assume a priori semantic agreement among peers and does not constrain to the adoption of fixed structures to organize resources.

Approaches like the ones described in [4], [16] organize peers into semantic communities and request forwarding is performed inside the borders of the communities or between semantically related communities. In [4] a P2P-based system to support efficient access to e-catalogs resident data is provided, where scalability issues are solved by organizing information space in communities. Each community is associated to a reference ontology and is related to other communities through semantic links, defined as mappings between ontologies. A query rewriting algorithm is used to determine which part of a request submitted to a community can be answered locally and which requires to be forwarded to other communities. Forwading policies are defined at community level, while in SERVANT different forwarding policies exploit semantic links at service level and associated matching information. An approach for service discovery based on a semantic overlay is proposed in [16], where offered services are semantically described according to peer local ontologies. Peers with similar ontologies are supposed to offer services in the same domain and are clustered to form SemSets. In each SemSet, a coordinator peer is elected. Coordinators of semantically related communities (e.g., Rental and Transport) are related through semantic relationships that are exploited for query routing between SemSets. With respect to SERVANT, this approach relies on SemSet coordinators (that is, a form of super-peers) for service discovery and adopts a coarser grain routing among communities rather than among peers.

OntSum [12] is an attempt to completely decentralize resource discovery that shares with SERVANT the assumption of peers endorsed with local ontology and an effective, semantic-based, request forwarding, but is not involved with service discovery.

In the RS2D approach [3] for semantic service discovery in P2P networks, each peer first determines the set of local services matching a request by use of a OWL-S matchmaker, then forwards the request to a selected set of neighbor peers from which it is expected to receive semantically relevant services and, at the same time,
minimizing the network traffic, evaluated according to a Bayesian model.

VI. CONCLUSIONS

In this paper, we presented experimental evaluation of SERVANT, a reference architecture for scalable semantic searching in P2P systems, that integrates in a comprehensive framework our previous work on service matchmaking techniques [5] and P2P service discovery [6]. The architecture supports service-based semantic search by means of a distributed service registry, composed of peers providing semantic-enriched services and semantic links between peers holding similar services; each service provider is also equipped with a service knowledge evolution manager, apt to update peer knowledge, and a semantic search assistant, apt to find services satisfying a user’s requested.

Experimentation results are encouraging, both in terms of performances and effectiveness of the discovery process. The proposed work features recall, scalability and different semantic forwarding policies as design goals. Open issues still regard the implementation of additional request forwarding policies that are adaptive with respect to the network conditions. Moreover, forwarding policies can be extended to consider complementary functionalities: if a peer is not able to satisfy a service request, more focused request re-direction should be provided, instead of randomly choosing a subset of peers connected at the logical P2P network overlay.

REFERENCES


