A DHT-based semantic overlay network for service discovery

Giuseppe Pirrò[a], Domenico Talia[a,b], Paolo Trunfio[a]

[a] *DEIS, University of Calabria, Rende (CS), Italy*
[b] *ICAR-CNR, Rende (CS), Italy*
{*gpirro,talia,trunfio*}*@deis.unical.it*

Abstract

The number of available Internet services increases every day. This trend demands distributed models and architectures to support scalability as well as semantics to enable efficient publication and retrieval of services. Two common approaches toward this goal are semantic overlay networks (SONs) and distributed hash tables (DHTs) with semantic extensions. SONs enable semantic-driven query answering but are less scalable than DHTs, which, in their turn, feature efficient but semantic-free query answering based on exact match. This paper presents a strategy and a system, called ERGOT, that combine DHTs and SONs to enable semantic-based service discovery in distributed infrastructures such as Grids and Clouds. ERGOT uses semantic annotations that enrich service specifications in two ways: (i) services are advertised in the DHT on the basis of their annotations, thus allowing us to establish a SON between service providers; (ii) annotations enable semantic-based service matchmaking, using a similarity measure between service requests and descriptions. An extensive evaluation of the system is presented and discussed. The experimental evaluation we carried out confirmed the efficiency of the implemented strategy in terms of both accuracy of search and network traffic.

Keywords: Semantic Web services, Semantic Overlay Networks, Distributed Hash Tables, Semantic Similarity.

1. Introduction

The main aim of the service oriented architecture (SOA) model is to allow developers and end-users to dynamically discover and assemble software modules (represented as services) to fulfill their needs. The SOA paradigm is therefore a viable model for the modular composition and reuse of third-party software components on a large scale. One of the stumbling blocks toward this goal, however, is the difficulty for potential users to discover new services of interest, particularly when such services are independently deployed by a large set of providers, on open large-scale computer networks.

Several approaches have been designed to provide effective service discovery, ranging from distributed architectures to mitigate centralization of service registries such as UDDI, to architectures leveraging Semantic Web technologies for providing semantic-based service descriptions and matchmaking. Regarding the former, most proposals for distributed and federated registries rely on peer-to-peer (P2P) protocols and distributed hash tables (DHTs). This is the case of DUDE, for example [1], which extends UDDI to multiple registries that form a federation with a DHT as a rendezvous point, as well as of the DHT-based Web service discovery system proposed in [31]. On the other hand, semantic-based service discovery is based on several techniques, e.g., information retrieval [9], rough set theory [19] or logic-based reasoning to infer semantic relations (e.g., exact, subsume) between a user request and a service profile [14]. Semantically rich service description annotation models,

such as OWL-S, SAWSDL, and WSMO provide the necessary underpinning for machine-processable annotations.

Both these areas suffer from some drawbacks. DHT-based search is efficient but is limited to exact terms, for instance the service name, and thus it does not fit well with semantic similarity search models, where service parts are described using multiple annotations. On the other hand, semantic service matchmaking usually requires logical reasoning, which makes it difficult to perform numeric result ranking. Therefore a definition of semantic similarity that takes into account the different annotations of a service request and description (e.g., operations, input, output) is mandatory. Finally, mapping the semantic search paradigm on a P2P network can be costly in terms of network traffic, if no attention is paid to how peers are connected with one another.

1.1. Contributions and Outline

The main goal of this work is to investigate and identify synergies between distributed service registries and semantic service discovery. Our hypothesis is that a P2P-based federation of registries can benefit from semantic annotations of service descriptions. Specifically, we propose to use P2P advertising of service descriptions to incrementally build a semantic overlay network (SON) between service providers. A SON is a virtual network designed to connect peers that are *semantically related*, that is, their respective content, service descriptions in our case, are *semantically similar*. The similarity between service descriptions is assessed on the basis of their annotations to ontology concepts.

Once a SON is available, it can be used to resolve service requests effectively, by ensuring that messages are routed between semantically relevant peers. This paper introduces a decentralized system based on these principles, called ERGOT (Efficient Routing Grounded On Taxonomy). ERGOT builds upon three main elements.

1. A DHT protocol, used to advertise semantic service descriptions annotated using ontology concepts. We assume that semantic annotations of service descriptions are expressed in W3C's standard SAWSDL[1].
2. A SON, which enables the clustering of peers that have semantically similar service descriptions. The SON is constructed incrementally, as a by-product of service advertising via a DHT.
3. A measure of semantic similarity between service descriptions, which overcomes the exact-term search limitations of DHTs. This measure combines the feature-based and information content-based similarity models [24].

ERGOT can be used in scenarios where a large number of services, published by different providers, need to be dynamically discovered and integrated into complex distributed applications. Examples include e-commerce and e-science applications, where the single components are available as services independently specified by their providers. This requires the availability of efficient mechanisms for semantic-based service matching and discovery, as needed, for instance, by workflow-based computing systems [8].

Consider for instance the case of a bioinformatics researcher who wants to make a comparison of genes from different species to find similarities between protein functions. To this end, the researcher could be interested in finding the widest possible set of sequence analysis services that may be used to support her/his task. Given the wide variety of sequence analysis methods and implementations, standard search methods based on service names and parameters would return only a small subset of all the relevant services available from the bioinformatics community. In this scenario, the ERGOT system could be used

[1]Semantic Annotated WSDL: `http://www.w3.org/2002/ws/sawsdl`

by service providers to advertise descriptions of their sequence analysis services annotated with ontology concepts, and by interested users to locate all the sequence analysis services that may be useful for the research they want to perform. As outlined before and detailed in the remainder of the paper, the combined use of DHT and SON approaches, together with a semantic similarity measure of service descriptions, enables ERGOT to perform this kind of semantic discovery task with high recall and low network traffic.

A preliminary version of the present work was given in [26]. In the present version we make the following contributions.

1. The various components of the ERGOT architecture are described in more detail. Besides, descriptions are accompanied by concrete examples.
2. An extensive evaluation campaign is conducted to compare ERGOT both with Chord and with unstructured P2P models in the manner of Gnutella. This latter evaluation is very important for showing the effectiveness of the SON, constructed by a content-similarity principle, as compared to the Gnutella topology, where links between peers do not take into account the semantics of contents.
3. A more detailed analysis of the related literature is conducted to underline the contribution of the present work, which to the best of our knowledge, is the first approach combining structured P2P network models and SONs.

The remainder of the paper is organized as follows. Section 2 gives a background on DHTs and SONs. Section 3 discusses related work. Section 4 presents an overview of ERGOT, introduces the notation used throughout the paper and describes the semantic similarity measure used to match service profiles. Section 5 discusses the implementation of the system. Section 6 provides an experimental evaluation of ERGOT. Finally, Section 7 concludes the paper.

2. DHTs and SONs: background and comparison

Distributed hash tables (DHTs) and semantic overlay networks (SONs) are the two technologies at the core of our proposal. We provide a brief summary of them here.

DHTs have gained recognition as a prominent network paradigm for data distribution and indexing, due to their scalability properties and efficiency in retrieving content. For the sake of explanation we are going to use the well-known Chord DHT model [33] as a reference. In Chord, each data item is associated with a key. Each peer can be used both to publish new data items on the network, using the *put(key, value)* primitive, and to submit search requests through the *get(key)* primitive. The hashing and routing mechanisms employed by Chord makes exact, key-based queries efficient. In Chord both network peers and data items are assigned an m-bit hash value (a node's *id* could be computed by hashing the peer's IP address). With this provision, the network has a ring topology consisting of at most 2^m nodes. Chord adopts a simple rule to assign values to nodes: a *put(key, value)* operation assigns *value* to the peer whose *id* is the *successor* of *key*. This is illustrated in Fig. 1, where *key 12* is assigned to node N_{13}. A *get(key)* operation can be initiated by any peer, and is routed around the ring with the help of a *finger table* associated with each peer. The finger table for a node contains the Chord *ids* of the node's neighbors. Neighbors are nodes located on the ring at exponentially increasing distance from a given node. This guarantees that any query is answered in $O(\log N)$ hops, where N is the number of nodes in the network [33]. In Fig. 1, the request posed by N_6 for *key 12* is routed at first hop to the node in N_6's finger table closest to (but not higher than) 12, that is, N_{10} which, in its turn, by looking at its finger table can easily find the peer responsible for *key 12* (i.e., N_{13}). DHTs only support key-based, exact queries, a serious limitation for semantic-based

searches where one is interested also in retrieving contents "semantically similar" to what is expressed in the request.

SONs provide a first step in this direction, although they do not directly address the problem. Originally proposed by Crespo and Garcia-Molina [7], the SON is a paradigm for organizing peers and enhancing content-based search. The main idea is that, by clustering the peers according to the semantic similarity of content they share, the clusters can then be exploited to speed up query routing while providing good recall. In particular, annotating contents with concepts drawn from a common ontology provides the necessary underpinning for partitioning an otherwise unstructured P2P network into SONs. An example of a SON is shown in Fig. 2, where contents shared by peers are semantically annotated. For instance,

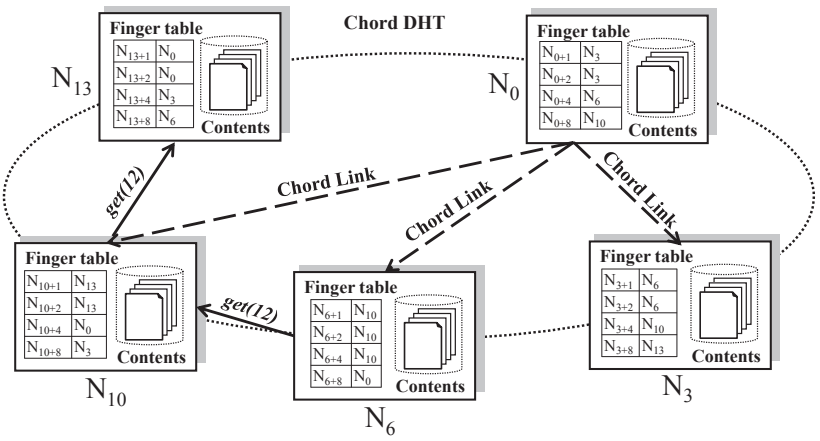

Figure 1: An example of Chord DHT

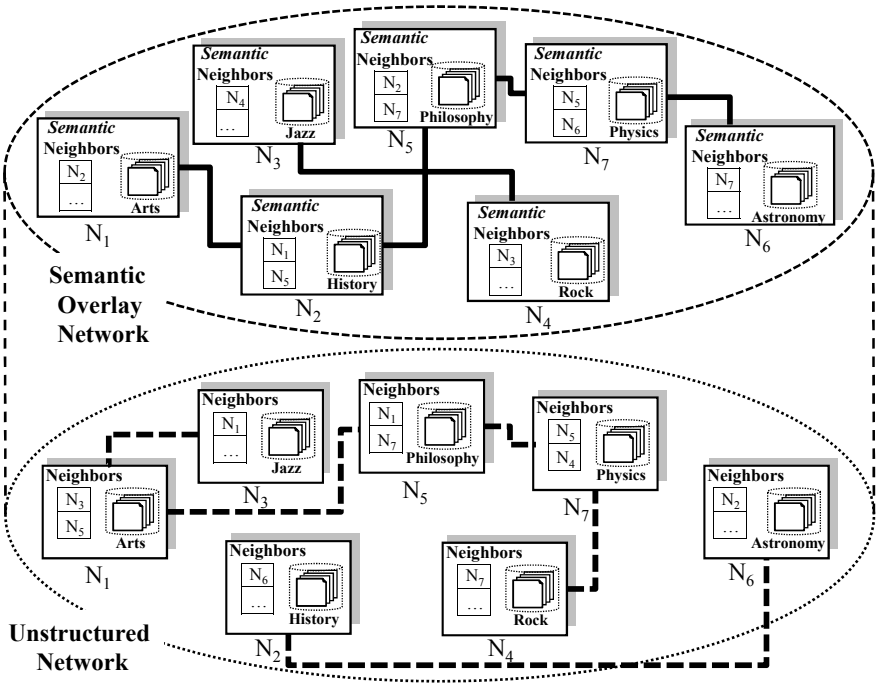

Figure 2: An example of SON

node N_4 shares contents related to *Rock* whereas N_2 has expertise in *History*. In the lower part of Fig. 2, nodes form an unstructured P2P network in the manner of Gnutella, where neighbor nodes are determined by querying some cache servers maintaining IP addresses of

Table 1: Comparison between DHTs and SONs

	DHTs	**SONs**
Content Placing	Fixed, based on hashing	Each peer is responsible for its content
Content Retrieval	Exact lookup	Flexible (based on similarity)
Network Structure	Fixed	Variable
Lookup cost	O(logN)	Variable
Topology Management	Stabilization protocol	Semantic links need to be rearranged
Semantics	None	Exploit a semantic artifact (e.g., an ontology)

online nodes. In this case it may happen that peers sharing semantically similar contents are not connected with one another. For instance although N_3 and N_4 share contents related to *Rock* and *Jazz* respectively, which are semantically related since they are two musical genres, they are not directly connected. In the top part of Fig. 2, a SON is shown in which peers determine their neighbors according to a criterion of semantic similarity between their respective contents. In this case, N_3 and N_4 have a link that can result in being very useful in performing content lookup. In [7] the effectiveness of these ideas as compared to the Gnutella flooding-based approach is discussed.

Table 1 provides a comparison of the relative merits of DHTs and SONs. DHTs are very scalable, and they guarantee efficient lookup, but only for exact queries where the "identity" of the content one is looking for is known a priori. SONs, on the other hand, are more flexible as they can perform semantic-based query answering, but their performance is affected by the overlay topology. Moreover, an intrinsic initial cost for building the SON has to be taken into account.

Although DHTs and SONs have been (separately) exploited in recent service discovery initiatives (see Section 3), no attempt has been made to combine the two approaches even though several considerations suggest that useful synergies can be found.

3. Related Work

Service discovery has been addressed from different, not necessarily disjoint, perspectives thus giving birth to several strands of research. Some of these focus on mitigating centralization by adopting decentralized architectures while other focus on defining semantically rich data models (e.g., OWL-S, SAWSDL and WSMO). In the remainder of this section we provide a comprehensive overview about service discovery initiatives from a network-architecture perspective. Table 2 summarizes the systems analyzed and compares them with ERGOT in terms of network architecture, support of semantics, ranking, and underlying techniques.

3.1. Centralized Service Discovery

Centralized service discovery architectures have as primary concern that of providing accurate mechanisms to select services relevant with respect to a user request. In this context, different techniques have been proposed. Woogle [9] leverages information retrieval techniques to efficiently exploit text contained in service descriptions. ROSSE [19] exploits rough set theory to identify some dependences between service properties and is able to compute the *Lower* and *Upper* approximation of services that match a user request. Matchmaker [14] was one of the first semantic matchmakers for Web services. Through reasoning, it allows one to define different semantic relations (e.g., exact, subsume, plug-in) between a user request and a service profile. ERGOT differs from ROSSE and Woogle since it exploits ontologies to annotate services and performs numerical ranking by exploiting semantic annotations of the various service parts. As for Matchmaker, ERGOT is able to perform numerical ranking of results, which can be very useful for better distinguishing between services related to a user request.

Table 2: Comparison among service discovery systems

	Architecture	**Semantics**	**Ranking**	**Techniques**
	Centralized			
Woogle [9]	Registry-based	No	Yes	Ad-hoc clustering technique
ROSSE [19]	Registry-based	No	Yes	Rough set theory
Matchmaker [14]	Registry-based	Yes	Yes	Reasoning
COMPAT [3]	Registry-based	Yes	Yes	Semantic matching
URBE [27]	Registry-based	Yes	Yes	Semantic similarity
	Decentralized			
DUDE [1]	DHT	No	No	Prefix Match of service names
Schmidt et al. [31]	DHT	No	No	Hilbert Space Filling curves
Emekçi et al. [10]	DHT	No	Yes	Behavior-based search
SPiDeR [28]	DHT	Yes	Yes	Behavior-based search
Vu et al. [34]	DHT	Yes	Yes	Ontology partitioning
Li et al. [20]	DHT	Yes	No	Semantic prefix routing
Zhu and Hu [35]	DHT	Yes	Yes	Locality preserving hashing
ATLAS [12]	DHT	Yes	No	RDF-based
Papazoglou et al. [23]	Super-peer-based	Yes	No	Concept-lattices-based matching
Sapkota et al. [29]	Super-peer-based	Yes	No	Similarity among peers
Meteor-S [17]	Super-peer-based	Yes	No	JXTA-based
Paolucci et al. [22]	Gnutella-based	Yes	No	Controlled flooding
Hypercube [30]	Ad-hoc topology	Yes	No	Efficient broadcast
WSPDS [13]	SON	Yes	No	Similarity among peers
Bianchini et al. [4]	SON	Yes	No	Ontology-based matchmaking
ERGOT	**DHT+SON**	**Yes**	**Yes**	**Similarity-based routing**

For the purpose of our analysis, we will review the COMPAT and URBE systems in more detail since they share some features with ERGOT. COMPAT [3] exploits an ad hoc ontology framework based on description logics and a thesaurus to enable semantic-similarity-based service discovery. In particular, service profiles are defined by exploiting both a service ontology, which allows to group services with similar features, and a domain ontology used to annotate operation names, input and output. The system supports ranking of results by computing, through the thesaurus, the semantic similarity between a request and a service profile. UDDI Registry By Example (URBE), proposed by Plebani and Pernici [27], is a Web service retrieval algorithm based on the evaluation of similarity between Web service interfaces expressed in WSDL. URBE defines a function $fSim$ that returns the similarity degree between two Web services. To evaluate the similarity of the elements that constitute a Web service interface, $fSim$ relies on two main functions: a name similarity function $nameSim$ which compares two names included in the Web service interfaces, and a data similarity function $dataTypeSim$ which evaluates how similar two data types characterizing the parameters in the Web service interfaces are. A semantic extension of the similarity algorithm, URBE-S, copes with the case of Web service interfaces semantically enriched by SAWSDL annotations. URBE-S defines a function $fSim^s$ which, instead of comparing the names of the service elements using $nameSim$, compares the related annotations using a function $annSim$ that receives as input two annotations and returns their similarity according to the way in which they are related in a reference ontology.

Although ERGOT also exploits similarity, there are several underlying differences between these systems. The base similarity measure between concepts has a different underlying strategy. URBE, in its variant URBE-S, exploits a path-based strategy whereas ERGOT uses a combination of features and information content. As for COMPAT, ERGOT uses a similar approach even if COMPAT combines similarity with deductive strategies and adopts Dice-based similarity aggregation strategies. ERGOT uses similarity not only in the process of service ranking but also in routing queries towards semantically similar neighbors. Moreover, similarity estimation in ERGOT is the main pillar allowing one to partition

the space of service providers in clusters of semantic similar peers. Finally, ERGOT is a decentralized service registry whereas both COMPAT and URBE are centralized.

3.2. Decentralized Service Discovery

Decentralized service discovery architectures have been proposed to cope with the pitfalls (e.g., scalability, fault tolerance) of centralized ones. To reach decentralization different approaches can be adopted, based on different underlying network architectures.

3.2.1. Approaches using DHTs

DUDE [1] extends the UDDI centralized service discovery mechanism by allowing multiple registries to form a federation with a DHT as a rendezvous point. Service information (on the basis of the service name) is distributed between the participants. The DHT querying mechanism limits the scope of the queries only to relevant registries. In [31], Schmidt and Parashar propose a DHT-based Web service discovery system. A service description is viewed as a set of points in a multidimensional space identified by the possible keywords found in service descriptions. In order to map the multidimensional space to DHT keys, the authors exploit Hilbert space filling curves (HSFCs), which ensure that the locality in the multidimensional space will be preserved after the reduction. However, this jeopardizes the hashing mechanism of the original DHT thus leading to load imbalance. In practice, with the dimensional reduction, data elements are not uniformly distributed in the index space, i.e., certain keywords can be more popular and hence the associated index subspace can be more populated. To cope with this issue the authors propose two load balancing mechanisms: load balancing at node join and load balancing at runtime.

The system proposed by Emekçi et al. [10] uses a DHT to publish and search for Web services based on their behavior and reputation. Each Web service is defined in terms of its implementation (I), service (S) and request (R): I can be considered as the BPEL description of the Web service behavior and is represented as a finite automaton; S is also a finite automaton describing the set of interactions observed by the users of the Web service; R is the set of finite automata corresponding to other Web services that are required by the current Web service to complete. Web services are published and searched through their finite automata representation. The core of this mechanism is based on hashing the finite automata onto a DHT. The hashing technique hashes the regular expression representation of a finite automaton to determine its location on the DHT. The DHT is then used to find a matching between service automata of the stored services with the request automata included in the query messages. The system includes also a reputation model used for ranking the search results.

SPiDeR [28] organizes nodes into a P2P structured network based on super-peers (SPs). Each peer is connected to an SP with which it interacts to perform operations of service advertising and discovery. Discovery is performed by using three different techniques based on keywords, categories, and behavior respectively. The system features a reputation component to perform quality of service (QoS) ratings of Web services.

The system proposed by Vu et al. in [34] combines ontologies, DHTs and a reputation mechanism based on trusted agents to perform service discovery. One of the key features of this system is the partitioning of a shared ontology, with which describing and querying for services in concept groups. Each concept group is summarized by a Bloom filter to enable quick concept-membership checking. Hence, a service is described as a set of unordered concepts each of which is represented by the Bloom filter of the group to which it belongs. To map service descriptions in the underlying DHT a special hash function is applied to the concatenation of all the keys in the description. A QoS component allows rating the quality of a Web service and is used to rank results.

Li et al. [20] proposed a structured P2P routing system that extends the Plaxton mesh to support decentralized semantic service discovery. Service advertisements and service requests are described using a service ontology such as OWL-S. Given a semantic service description, information such as input, output, preconditions and effects are extracted as ontological concepts. These concepts are encoded to a series of binary strings, called the service's characteristic vector (CV). Each peer maintains a routing table and a neighborhood table, which are extended from those of the Plaxton mesh. When a peer receives a request, it first converts the service description to its CV, and calculates the semantic similarity between the CV and peers in the neighborhood table to see it the CV falls within the range of semantic distance covered by the neighborhood table. If so, the message is forwarded directly to the destination; otherwise the routing table is used to find the entries that share the longest common prefix with the CV.

Zhu and Hu [35] designed a DHT-based semantic discovery system that shares some basic ideas with the system proposed by Li et al. [20] , even if it is focused on indexing and searching files rather services. The system represents each file and each query as a term vector (TV). A set of locality sensitive hashing (LSH) functions is used to produce a small number of semantic keys starting from the TV of a file or a query. The locality-preserving property of LSH functions ensures that the indexes of semantically close files and queries are placed, with high probability, into close nodes of the DHT. Discovery is performed in three steps: first, a small number of semantic keys are derived from the query's TV by using the same set of LSH functions using for file indexing; then, for each semantic key associated with the query, a DHT lookup is performed to retrieve the identifiers of the semantically close files from the node which is responsible for that key; finally, the requesting node merges the replies from all responsible nodes and generates a materialized view of the query results.

Finally, the ATLAS system [12] has been designed as a decentralized mechanism for resource discovery in S-OGSA [6]. ATLAS adopts a DHT-based architecture to publish and discover information about Grid resources in the form of RDF triples. ATLAS allows resolving conjunctive queries expressed in a logic language based on triple patterns.

3.2.2. Super-peer based approaches

Papazoglou et al. [23] used a P2P approach to build a federation of UDDI-enabled registries which operates in a decentralized fashion. Federations represent common interest groups of providers (peers) that band together to provide syndicated services to their customers. Two key concepts in a peer service syndication are publication and subscription. Publications are documents that name, describe and publish the existence of peers that act as service providers, while subscriptions also name, describe and publish the service requirements of peers that act as service requesters within a service syndication. For each syndication a specific peer acts as super-peer by providing an event-notification service that receives and stores the publications and subscriptions of the entire peer syndication. The super-peer organizes the publication space using a *concept lattice* and stores it in its own local registry. Similarly, the collection of subscriptions in a syndication are structured in a *subscription lattice*. This allows super-peers to determine subscription/publication matches and which subscribers are affected by a change to the publication lattice.

Sapkota et al. [29] proposed a system composed by a set of Semantic Web service (SWS) nodes that are clustered together based on similar service descriptions. A cluster consists of at least one super-node. Each cluster is maintained by one of the super-nodes, the cluster manager, which is dynamically chosen based on its availability, processing power, and so on. The cluster manager indexes the Web services registered with its clusters and facilitates inter-cluster communication. Matchmaking between service descriptions and user requests (both described using WSMO) is always evaluated locally first. In the case of partial match or no match, the request is forwarded to the cluster manager.

Meteor-S [17] supports semantic-based organization of Web services in a federation of registries. This system, developed in JXTA [11], is based on an unstructured P2P network and is mainly meant to organize service publications by identifying the most suitable registry to host a service description.

3.2.3. Approaches using other kinds of overlay

The WSPDS system [13] aims at constructing an overlay network of peers by comparing their Web service descriptions. Nodes create links by comparing the inputs and the outputs of their services by exploiting the matchmaker described in [14]. The similarity between a query and the peers to whom forward it is computed by the same matchmaker. Hypercube [30] imposes a deterministic shape on a P2P network by using a globally known ontology to determine the organization of peers in the network topology. The system proposed by Paolucci et al. [22] relies on the Gnutella protocol and uses OWL-S (formerly DAML-S) as the service description language. Every Web service (P2P node) contains a DAML-S description of its capabilities and the associated engines for parsing ontologies, as well as a P2P discovery module. Service requests generated at a given node are broadcast to the network through the P2P discovery module, which implements a controlled-flooding algorithm of the Gnutella protocol.

Bianchini et al. [4] proposed a P2P-based semantic driven service discovery (P2P-SDSD) framework that organizes peers into a semantic overlay. The semantic overlay can be seen as a continuously evolving conceptual map across collaborative peers that provide similar services, thus enabling effective similarity-based service search. The system defines optimization strategies for request propagation over the P2P network that keep the network overhead generated by the discovery process low. This approach is the closest to ERGOT even if there are several underlying differences. First, the SON is constructed in a different way. Bianchini et al. use a probe collaboration model with a bounded TTL to discover peers with semantically related services. In ERGOT, the semantic overlay is constructed as a natural process during the service publishing phase in the DHT. Besides, ERGOT also enables to pose a query via the DHT native algorithm, which can result in being very useful when there are not so many services published and then it is difficult to perform search based on semantic neighborness.

Although most of the analyzed approaches share some characteristics with ERGOT (as summarized in Table 2), none of them combine the efficiency of DHTs with the semantic support provided by SONs. In fact, on the one hand DHTs are very efficient for performing exact lookup but they do not provide any semantic support as a native feature; on the other hand, one of the main problems when building a SON is how to find interesting neighbors. ERGOT represents a new way of facing Web service discovery by exploiting publishing of semantic annotations in a DHT to build a SON.

4. Overview of the approach

By analyzing DHTs, we observed that peers can potentially discover other peers with semantically similar content by piggybacking on the P2P communication that occurs naturally during the content publishing activity in a DHT. Thus, semantic links can be established at no additional cost as a by-product of normal network activity. Hence, semantic links can be exploited not only to improve search performance, but also to perform semantic-based similarity search. This is because peers can ask their immediate semantic neighbors whether they are responsible for contents semantically close to a user request.

As an example, suppose a user is looking for a service that offers *car-selling* information. A possible service description request may include terms such as *price*, *car model*, *year* and so forth. Using Chord, relevant services would only be retrieved if their semantic profiles

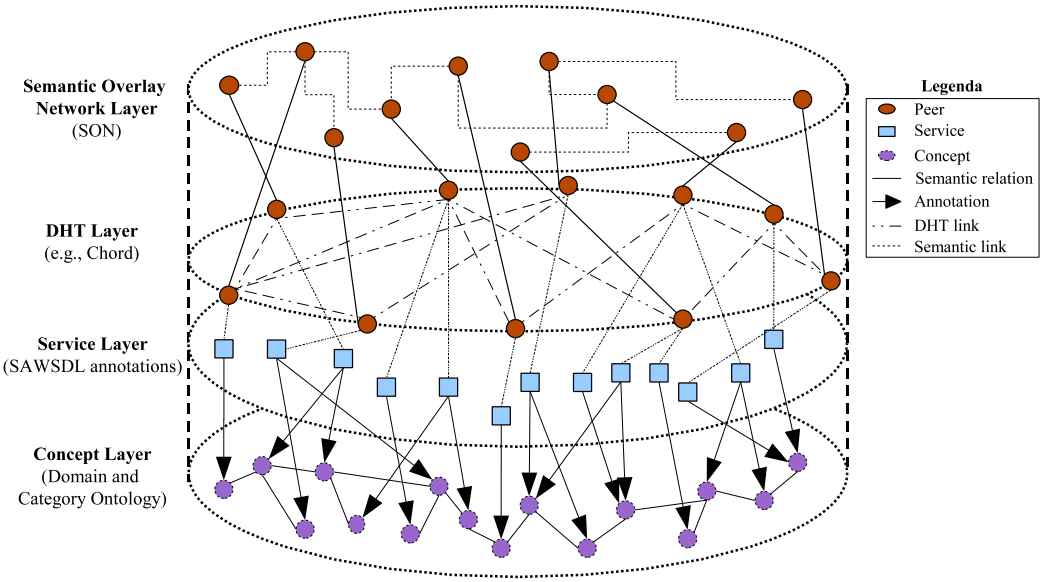

Figure 3: An overview of the ERGOT layers

include exactly the same keywords used in the query. However, it can be possible that in the specifications "related" terms are used instead, for instance, *automobile* in lieu of *car* and *car type* in lieu of *car model*. In this case, ERGOT's semantic similarity approach will still be able to return relevant services, while Chord would fail to find any matches. We have explored the viability of these ideas in the specific case where peers are autonomous and distributed semantic service registries.

Fig. 3 provides an overview of the different layers in the ERGOT architecture. The lowest layer (i.e., *concept layer*) includes the concepts in the ontology used for the annotations, which are connected by semantic relations such as *isa* and *part-of*. The *service layer* contains the services annotated via SAWSDL to the concepts in the *concept layer*. As can be noted, each service can be annotated to different ontology concepts. The *DHT layer* (Chord [33] in the case of ERGOT) is responsible for service profile publishing according to their annotations on the concept layer. The publication of services allows us to construct a SON between service providers (i.e., *SON layer*). Service requests can be posed both at the SON layer and at the underlying DHT layer. In the rest of this paper we elaborate more on these aspects.

4.1. Assumptions

The ability to match service descriptions to user searches is at the cornerstone of any service discovery model. Purely syntactic approaches, based on service name similarity, have been extended in several directions, using either ontology-based semantics [14] or information retrieval techniques [9] as discussed in Section 3. On the other hand, when the matching is performed using logic-based reasoning, the match status can itself be described using a simple classification model, e.g., *exact*, *plugin*, *subsume*, and *fail* [18]. A quantitative, numerical assessment of these concepts, however, has been proved to be difficult [5] for classes like *subsume* or *plugin*, making it hard to accurately rank the matches. To tackle this aspect some initiatives, such as OWL-MX [15], and WSMO-MX [16], have recently been proposed. Other initiatives such as the work by Vu et al. [34] or the SPiDeR system [28], address result ranking by only relying on QoS indicators, and do not extend to the fine-grain semantic description of the service itself.

ERGOT exploits an ontology-based measure of semantic similarity between service descriptions and service requests. In particular, it exploits both coarse-grain service function-

ality descriptions and at a finer level, when available, annotations on individual elements of a service signature. We use two sources of knowledge for these annotations: a high-level *Category Ontology* (CO) for service description and a *Domain Ontology* (DO) that accounts for semantic data type description. For the purpose of this paper, we assume that services have been already annotated according to the SAWSDL approach. The SAWSDL specification defines how to add semantic annotations to various parts of a WSDL service specification (e.g., input and output message structures, interfaces and operations). In order to enable semantic annotations, SAWSDL defines attributes that can be applied to both WSDL and XML Schema elements. SAWSDL provides a generic annotation reference mechanism that can be applied to abstract definition of a WSDL Web service. For instance, via SAWSDL it is possible to annotate WSDL interfaces and operations with categorization information that can be used to publish a Web service in a registry such as UDDI. Semantic annotations usually refer to concepts defined in one or more ontologies.

An example of WSDL specification along with semantic annotations is shown in Fig. 4. The service has an operation called *Order*, which takes as input a *customerNo* and an

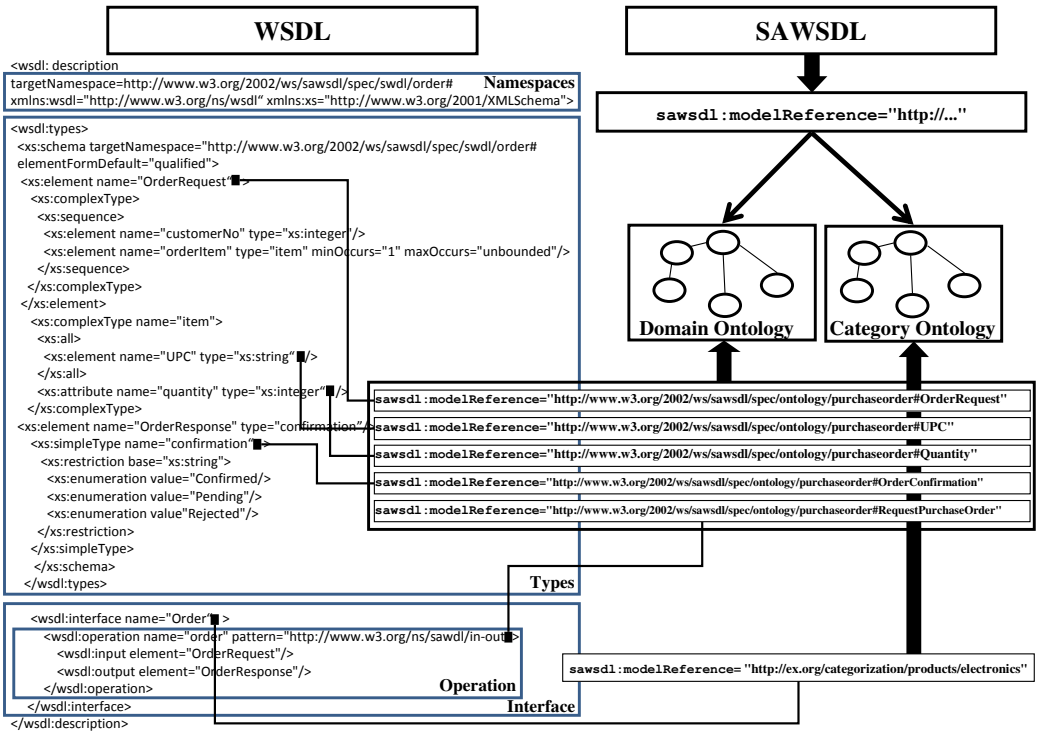

Figure 4: Example of WSDL interface with SAWSDL annotations

Item, that, in its turn, has a *UPC* and a *Quantity*, and returns as output the state of the order that can be *Confirmed*, *Pending* or *Rejected*. SAWSDL annotations are expressed through the `sawsdl:modelReference` primitive. It can be noted that SAWSDL allows *Types*, *Operations* and *Interfaces* to be annotated. In our previous example, *Types* and *Operations* are annotated with concepts from a *Domain Ontology (DO)*, which contains specific concepts in the knowledge domain of the Web service.

On the other hand the *Interface* is annotated with concepts in a *Category Ontology (CO)* (specifically to the category */products/electronics*). This latter annotation is very useful for summarizing the goal of a service and is similar to the categorization mechanism featured by UDDI where it is possible to categorize services by using some standard taxonomies.

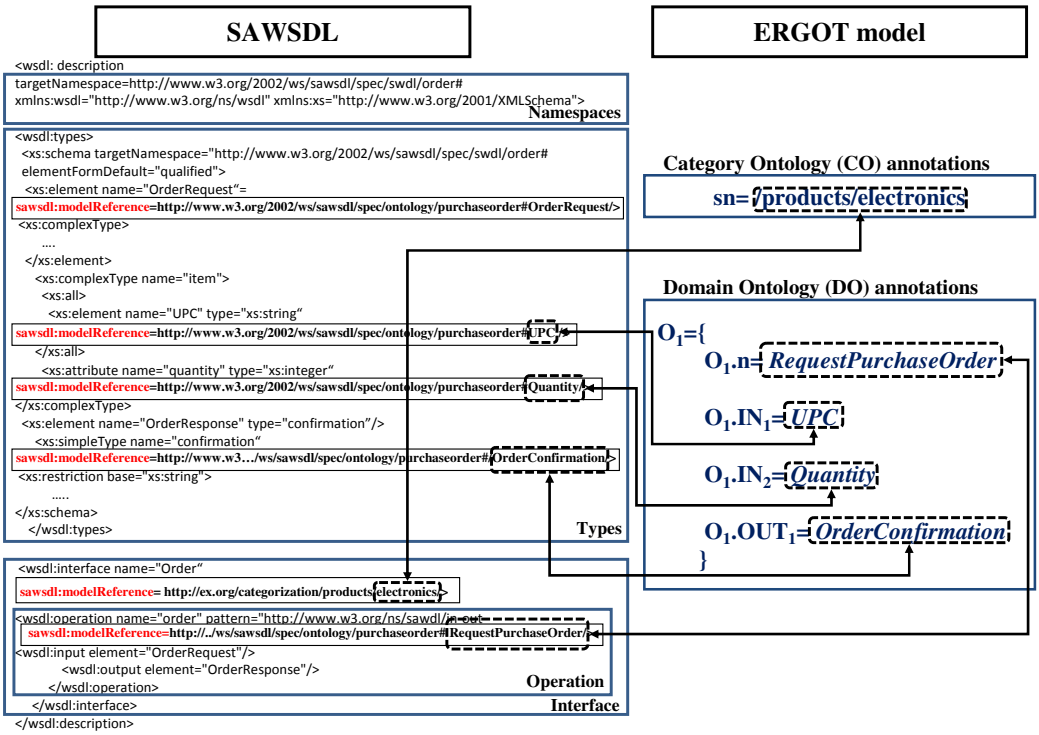

Figure 5: Example of annotated service interface with its representation according to the ERGOT model

4.2. Notation

This section introduces our notation for representing the elements of a Web service interface. We refer to these as a *service profile*. The elements of a service profile will be considered during the similarity evaluation. We introduce the following representation of a service specification.

- A service profile $P = \langle sn, \mathcal{O} \rangle$ consists of a service name sn and a set $\mathcal{O}$ of operations.
- An operation $O \in \mathcal{O}$ has a name $O.n$ and a set $O.IN$ and $O.OUT$ of inputs and outputs: $O = \langle n, IN, OUT \rangle$. For example, $P.sn$ is the service name, and $P.\mathcal{O}_i.IN_j$ is the j-th input of the i-th operation.

Fig. 5 shows an example of annotated service interface along with its representation according to the ERGOT model. Annotations can be associated with sn, n, and each input and output, denoted $ann(x)$ for a generic element x. As mentioned above, we use one or more CO concepts for describing the overall function of a service, and DO concepts, when available, to annotate elements of its internal structure. It is also possible that $DO \equiv CO$. In the general case we assume that $ann(P.sn) \subseteq$ CO, $ann(O.n) \subseteq$ CO, $ann(P.\mathcal{O}_i.IN_j) \subseteq$ DO and $ann(P.\mathcal{O}_i.OUT_j) \subseteq$ DO. From the requester perspective, a service request $R = \langle \mathcal{C}, \mathcal{O} \rangle$ matches the structure of a service profile, but ontology concepts representing user requirements for matching services appear in lieu of service component annotations. ERGOT enables users to pose service requests in a similar manner as the *Query by Example* (QBE) approach does for databases. For instance, users can define the characteristics of the wanted Web services via a user-friendly interface by navigating and choosing relevant concepts. Then, the QBE approach converts the user input into a request to be sent across the network.

In particular, requests are built by using concepts $\mathcal{C}$ from the CO to express and summarize service functionality requirements, while the $\mathcal{O}$ structure can be used to specify semantic types for operations input and output, as well as specific operations names. A request R

posed by a user is evaluated against a collection of service profiles $\mathcal{P} = \{P_1 \dots P_n\}$ each peer advertises on the network, by matching each of the concepts found in R's structure with the corresponding annotations in each profile $P \in \mathcal{P}$, using the semantic similarity function $RPsim(R, P)$, which will be described in Section 4.4. Function $RPsim(R, P)$ is defined inductively on P's structure, using the requirement specifications found in R.

The level of precision of a request may vary, from simply a set $\mathcal{C}$ with no constraint on the service operations, to a request for any operation that satisfies a given set of input or output types, to a full set of requirements that include named operations with given input and output types. Algorithm 1 summarizes the main steps performed by each peer receiving the request. Function *RPsim* (see Section 4.4) returns a value in the interval

Input: R: service request ;
T_h: similarity threshold ;
Output: $\mathcal{P}_\mathcal{R}$: ordered list of services that match the request ;
1 $\mathcal{P}_\mathcal{R} := \emptyset$;
2 **foreach** $P \in \mathcal{P}$ **do**
3 $sim := RPsim(R, P)$;
4 **if** $sim \geq T_h$ **then**
5 $\mathcal{P}_\mathcal{R} := \mathcal{P}_\mathcal{R} \bigcup P$;
6 **end**
7 **end**
8 **return** $\mathcal{P}_\mathcal{R}$;

Algorithm 1: Matchmaking performed by each peer

$[0, 1]$. The higher the result of *RPsim*, the higher the similarity between the request and the profile. Function *RPsim* enables result ranking, which is an important factor to be taken into account in an open and large-scale network where it is mandatory to differentiate between services belonging to different service providers. The output of each request is the list $\mathcal{P}_\mathcal{R}$ whose elements are ordered according to their similarity to the request R.

Note that even though category annotations do not contribute directly to the ranking of results, they are used a priori to route queries via the SON to the peers that keep information about the services that are more similar to the categories expressed in the request (see Section 4.4).

4.3. A running example: looking for a convertible

The concepts described in the previous sections are detailed through the scenario depicted in Fig. 6. Here, an example of service request in the *car-selling* domain is expressed by using ontology concepts drawn from both a domain ontology and a category ontology. The user is interested in finding Web services that given a convertible model and a year return price and description of some convertible. According to the QBE approach, this query can be posed by simply picking, from both the DO and the CO, the ontology concepts that better approximate what the user has in mind. Fig. 6 depicts a portion of the DO in the *car-selling* domain taken from an online *Vehicle Sales* ontology[2], which has been extended for the purpose of the example. Here some concepts such as *Vehicle*, *Car*, *Price*, and so forth are defined. In particular, a *Car* is defined to be a kind of *Motor Vehicle*. On the other hand, a *Convertible* is represented as a subclass of *Car*. Fig. 6 also reports a portion of CO extracted from the North American Industry Classification System (NAICS) published by the US Census[3]. Here, the category *Wholesale trade* is defined of which, *Merchant wholesalers durable goods* is a subcategory. By digging more into the categorization we can also find an *Automobile Merchant* category, which seems very appropriate for summarizing

[2]The ontology is available online at `http://www.heppnetz.de/ontologies/vso/ns`
[3]More details are available at `http://www.census.gov/eos/www/naics`

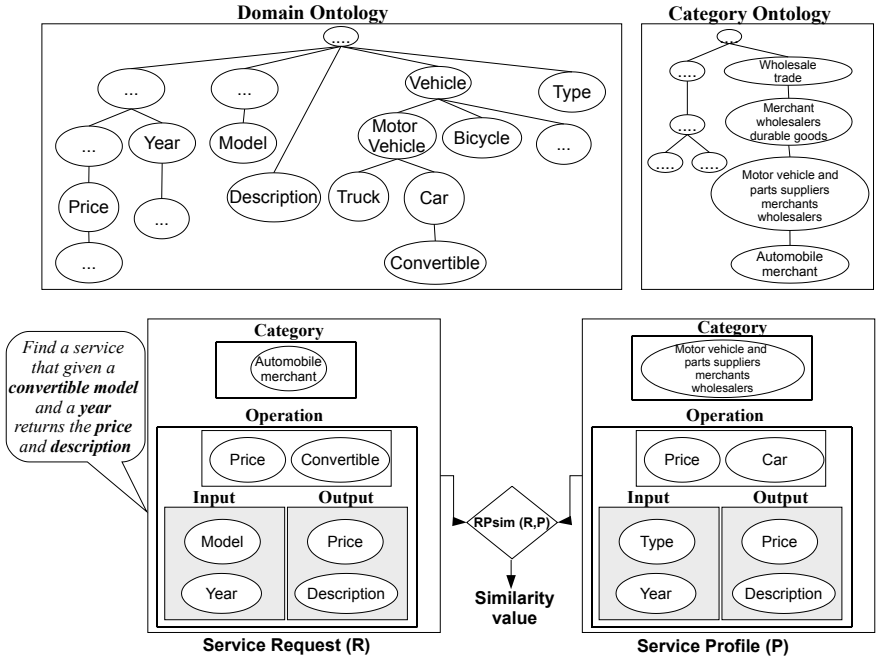

Figure 6: Example of service annotation and request

the information needed by the user. It can be noted that both the user request and the existing service profile, stored in a peer, use ontology concepts to specify their respective semantics. When the service request, routed through the network, reaches the peer hosting the service profile, the latter via the *RPsim* function (see Section 4.4) computes a similarity score between the request R and the service profile P.

4.4. On matching service profiles

We now build the definition of service profile similarity, denoted $RPsim(R, P)$, from the bottom up, starting with a measure of similarity between two concepts c_1, c_2 from the same ontology, proposed in our previous work on semantic similarity in ontologies [24]. This similarity measure has been extensively evaluated in several scenarios including both general and domain-specific ontologies. Therefore, its evaluation per se is out the scope of this paper.

Let $msca(c_1, c_2)$ denote the most specific common ancestor of two concepts c_1, c_2 within the ontology. The similarity between c_1, c_2 is defined as

$$Csim(c_1, c_2) = \begin{cases} 3 \cdot IC(msca(c_1, c_2)) - IC(c_1) - IC(c_2) \text{ if } c_1 \neq c_2 \\ 1 \text{ otherwise} \end{cases}$$

where $IC(c)$ is the *information content* of a concept c [32]. Intuitively, it quantifies the information carried by a concept, in terms of the number of its hyponyms (i.e., subconcepts) in the ontology. Formally,

$$IC(c) = 1 - \frac{log(hypo(c) + 1)}{log(C)}$$

where function *hypo* returns the number of hyponyms of a given concept c and C indicates the total number of concepts in the ontology. As an example, the similarity measure between the concepts *Convertible* and *Car* in the portion of ontology depicted in Fig. 6 is 0.84, which is reasonable since a convertible is a kind of car.

The definition of $RPsim(R, P)$ is inductive on P's structure, and it accounts for the varying specificity of the request R, as anticipated earlier. Initially, let us consider a fully specified request for a single operation type, i.e., $R = \langle \mathcal{C}, \mathcal{O} \rangle$ where $\mathcal{O} = \{O\}$, and a DO concept c as well as a set of input and output type requirements IN, OUT associated with O, i.e., $O = \langle c, IN, OUT \rangle$. This request structure is matched to a profile P inductively, by matching the annotations of each operation $P.O_i$ to $R.O$ and taking the similarity value of the best-matching operation. In turn, matching one operation structure $P.O_i$ requires matching the set $P.O_i.IN$ (respectively, $P.O_i.OUT$) to set $R.IN$ (respectively, $R.OUT$), and operation names $P.O_i.n$ to $R.n$.

The similarity of input semantic types, $Tsim(P.O_i.IN, R.O.IN)$, is the sum of the best matches between all possible pairs $(ann(x), y)$ with $x \in P.O_i.IN$, $y \in R.O.IN$:

$$Tsim(P.O_i.IN, R.O.IN) = \sum_{y \in R.O.IN} \max_{x \in P.O_i.IN} Csim(ann(x), y)$$

The same function applies to output types. This function rewards services that contain an operation whose input (resp. output) types best match the concepts in $R.O.IN$ (resp. $R.O.OUT$). Matching of $P.O_i.n$ to $R.O.c$ is a straightforward application of the $Csim$ function:

$$Nsim(P.O_i.n, R.O.c) = Csim(ann(P.O_i.n), R.O.c)$$

The overall similarity between $P.O_i$ and $R.O$ is the weighted sum of the three components just defined:

$$\begin{aligned} OPsim(P.O_i, R.O) = \; & \alpha \; Nsim(P.O_i.n, R.O.c) + \\ & \beta \; Tsim(P.O_i.IN, R.O.IN) + \\ & \gamma \; Tsim(P.O_i.OUT, R.O.OUT) \end{aligned}$$

This last definition accounts for the possibility that the user request contains incomplete requirements, as in this case, the similarity value associated with a missing annotation is just zero. Thus, in particular a minimal request may only include a concept for the operation name, or for the input/output semantic data types. The three parameters α, β and γ are used to weight the contribution of operation names inputs and outputs.

Finally, the overall similarity between P and $R.O$, i.e., when R contains requirements for one single operation, is simply the best similarity value over all operations in $P.\mathcal{O}$:

$$RPsim(R.O, P) = \max_{O_i \in P.\mathcal{O}} OPsim(R.O, O_i)$$

We extend this function to the case where R specifies requirements for multiple operations $R.\mathcal{O}$, simply by adding up the similarities values that result from the best matches between each $O_j \in R.\mathcal{O}$ and each $O_i \in P.\mathcal{O}$:

$$RPsim(R, P) = \sum_{R.O_j \in R.\mathcal{O}} \max_{O_i \in P.\mathcal{O}} OPsim(O_j, O_i)$$

The $RPsim(R, P)$ similarity function considers the best matches between the request and the operations described in a service profile. The evaluation of the similarity measure itself as well as the comparison with related approaches to service matchmaking (see for instance the URBE approach [27]) is an interesting topic for future work but it is beyond the scope of this paper. Here the focus is in presenting a new model of overlay network exploiting the efficiency of DHTs and the accuracy of SONs. Returning to the running example introduced in Section 4.3, the similarity between the request and the service profile depicted in Fig. 6 is 0.87. As it can be noted, the semantic similarity measure, even if the concepts used to express the request and annotate the service profile are not the same, is able to find

a match. In fact, it recognizes, for instance, that the concept *Car*, used to annotate the service profile, is similar to the concept *Convertible* used in the request. The same reasoning applies for the category annotations where the concept *Automobile Merchant* is used in the request whereas the service profile is summarized by an annotation to the *Motor vehicle and parts suppliers merchants wholesalers* concept.

4.4.1. A note about multiple ontologies

In the current implementation, ERGOT assumes that peers share a common ontology. On the one hand, sharing a common ontology can be realistic, in the case, for instance, of category ontologies. In fact, it is unlikely that each service provider will build its own category ontology since there are several already available generic categorization systems such as NAICS, which are constantly maintained. On the other hand, sharing a common ontology can be unrealistic since in distributed scenarios it is also common to have different representations for the same knowledge domain. This raises the problem of finding correspondences between ontologies (also known as ontology matching), which is a very active research area. Although we do not address the problem here, it can be noted that an algorithm such as the *Semantic Coordinator (SECCO)*, described in our previous work on ontology matching in distributed environments [25], can be adopted.

5. System Implementation

This section discusses in more detail the architecture of ERGOT from an implementation-oriented perspective. In particular, here we describe the mechanisms related to service publishing and discovery.

5.1. Publishing semantic service profiles

Consider a service profile $P = \langle sn, \mathcal{O} \rangle$, and let $C_P \subseteq$ CO be the set of its service-level annotations. P is published by invoking the standard DHT primitive *put(key, value)*, using each concept $c_P \in C_P$ as the key. Thus, publishing requests are of the form $put(c_P, P)$. In addition, however, we enhance the DHT behavior to enable the construction of semantic links between peers. Semantic links between peers build the SON, which can be exploited to facilitate service discovery. In this section we explain the publication algorithm and describe the process of building semantic links.

For each published profile P we distinguish between its *hosting peer* $hp(P)$, i.e., the peer that requests the publishing of P; and the *responsible peer* $rp(P)$ to which P is assigned according to the standard DHT routing algorithm. To recall, in general a DHT defines a mapping function $p = publish(k)$ that maps a key k to a peer p (since in our experiment we have used Chord, we assume this function to be $publish(k) = successor(k)$). This means that, since we are using CO concepts as keys, each peer is responsible for a subset of the concepts in the CO:

$$concepts(rp) = \{c \in \text{CO } s.t.\ publish(c) = rp\}$$

Algorithm 2 shows the publishing mechanism described above. Each service profile gets published according to its annotations to the *CO* and the responsible peer stores the whole service profile along with the *id* of the publisher id_p (i.e., the hosting peer $hp(P)$).

5.2. The Semantic Annotation Table

In ERGOT, each peer p maintains data it is responsible for in a *semantic annotations table* (SAT). If $c \in concepts(p)$ (i.e., p is responsible for c), then p's SAT records all peers that host service profiles that are annotated using c.

$$SAT(p) = \{hp(P) s.t.\ c \in ann(P.sn) \cap concepts(p)\}$$

Input: P: the set containing k service profiles ;
P_i: a generic service profile with $1 \leq i \leq k$;
$C_{Pi} : ann(P_i.sn) \subseteq CO$;
id_p: id of the *hosting peer (hp)* ;

```
1 foreach P_i ∈ P do
2 |   foreach c ∈ C_Pi do
3 |   |   h := hash(c) ;
4 |   |   put(h, ⟨id_p, c, P_i⟩) ;
5 |   end
6 end
```

Algorithm 2: Publishing service profiles

In practice, each peer keeps a record of peers that publish service profiles with the specific semantic annotations they are responsible for along with the service profiles themselves.

Fig. 7 shows the SAT for peer N_{10}, the successor of the key obtained by hashing the `Automobile Merchant` concept to which service profiles SP_3, hosted by peer N_3, and SP_1, hosted by peer N_{13}, are annotated. In this case we have that $rp(SP_1)$=N_{10}, $rp(SP_3)$=N_{10} and $hp(SP_3)$=N_3 and $hp(SP_1)$=N_{13}. Note that in the SAT, hosting peers are grouped according to their common CO concept, `Automobile Merchant` in this case, and the service profiles according to their hosting peers. Note that it is common for a peer to be responsible for more than one concept, e.g., N_{10} is responsible for both `Automobile Merchant`, and `Hardware Merchant` and therefore *publish*(`Automobile Merchant`) = *publish* (`Hardware Merchant`). We use the information in the SATs to establish semantic links between peers. Suppose

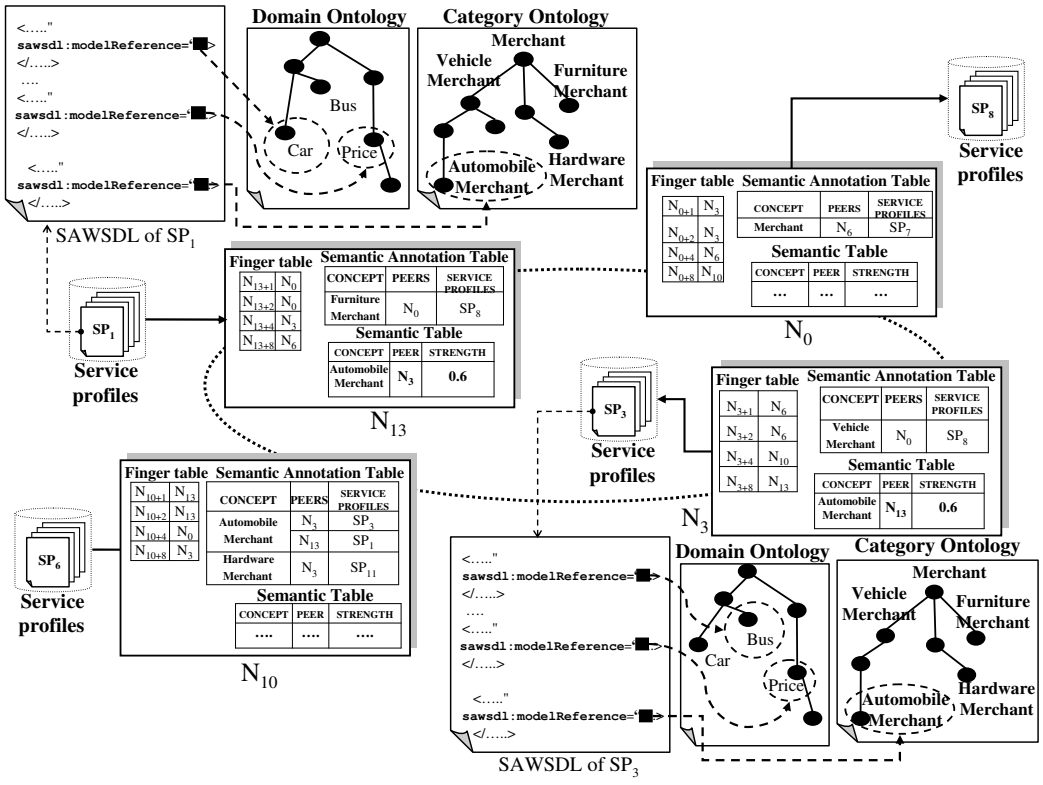

Figure 7: Construction of semantic links

that the hosting peer $p = hp(P)$ publishes P using $put(c, P)$; when the request is routed to $p' = rp(P)$ (through the usual finger table mechanism), p' now responds by sending to p the entries of its SAT that correspond to the concept c used as key. The entries in the SAT sent by p' contain the set of service profiles annotated so far to c. In the general, we refer

to the peers included in the SAT entries sent from p' to p as *semantic neighbors* $sn(p, c)$ of p on the concept c:

$$sn(p, c) = \{hp(P) \; \forall \; P \text{ s.t. } c \in ann(P.sn) \wedge c \in concepts(p')\} \subseteq SAT(p')$$

The peers in $sn(p, c)$ host services that have been annotated using the same category concept c used to describe P. Algorithm 3 describes the procedure followed by the *responsible peer* (p' in the above-mentioned example) when receiving messages containing semantic service profiles. Note that these messages are routed by the underlying DHT protocol, Chord in the case of ERGOT.

Input: A message of the form $\langle id_p, c, P_i \rangle$ routed by the DHT protocol ;
Output: SN: set of ranked peers that have published service profiles on c ;
1 $rank := 0$;
2 $insertInSAT(\langle id_p, c, P \rangle)$;
3 $SN := \emptyset$;
4 **foreach** $s = \langle id_{p'}, c', P' \rangle \in SAT$ *s.t.* $c' = c$ **do**
5 $\quad rank := RPsim(P', P)$;
6 $\quad SN := SN \bigcup \{\langle id_{p'}, rank \rangle\}$;
7 $\quad notifyUpdate(id_{p'}, id_p, c, rank)$;
8 **end**
9 $replyToPublisher(id_p, SN, c)$;

Algorithm 3: Semantic neighbors discovery

The *responsible peer* ranks by semantic similarity (see line 5) peers that have already published service profiles on the concepts it is responsible for and sends back rank results to the publishing peers (see line 9). This finer grained similarity assessment allows one to distinguish among service profiles annotated to the same CO concepts and investigate to what extent the publisher peer advertises a service profile similar to existing ones. Note that the SAT for p is populated incrementally each time it publishes a new service profile, with no additional routing as we are piggybacking on the original DHT routing strategy (only one new message back from $rp(P)$ to $hp(P)$ is needed).

In the example depicted in Fig. 7, when N_3 publishes the service profile SP_3 annotated to `Automobile Merchant`, N_{10} informs N_3 that N_{13} owns a service annotated to the same concept. Services belonging to the same category can be further differentiated by applying $RPsim$ on their respective profiles thus taking into account operations with their input and output. For instance in Fig. 7, even if N_{13} and N_3 publish two service profiles (i.e., SP_1 and SP_3) annotated to the same CO concept (i.e., `Automobile Merchant`), the two services can be differentiated since they are annotated to different DO concepts, that is, Car and $Price$ and Bus and $Price$, respectively.

5.3. Building semantic links

A peer p establishes semantic links $p \rightarrow ng$ with each neighbor $ng \in sn(p, c)$. Not all of the peers in $sn(p, c)$ are interesting neighbors, however. It may be the case that one of these peers hosts many services, only one of which is annotated using c. As the semantic links are used to create a SON, which is used for semantic similarity search, installing a link to such a neighbor would actually be misleading for many of the searches. We deal with this variability in two main ways. First, we limit the extent of the set $sn(p, c)$ received by p', by using a predefined threshold on the minimal number of services that each $ng \in sn(p, c)$ annotates using c. And second, we associate a measure of *strength* to each new semantic link $p \rightarrow ng$ based on the semantic similarity function between the services hosted by p and ng described in Section 4.4.

Fig. 7 shows the new *semantic table* (ST), maintained by each peer in addition to its finger table and the SAT discussed earlier. The ST in the SON can be seen as the

counterpart of the finger table for the DHT. In the example in Fig. 7, the ST for peer N_{13} stores the semantic link with peer N_3 on the concept `Automobile Merchant`. The strength of the link is 0.6 obtained by computing $RPsim(SP_3, SP_1)$. Note that the link is bidirectional, since in the ST of N_3 the link with N_{13} is also stored. Algorithm 4 describes the procedure followed to establish semantic links.

Input: A *replyToPublisher* $\langle id_p, SN, c\rangle$ message ;
$dimST$: the maximum size of the ST ;
Th_s: a threshold to accept a semantic neighbor ;
while $|SN| > 0$ **do**
  $\langle id_{p'}, rank_{(id_{p'},c)}\rangle := extractEntry(SN)$;
  if $rank_{(id_{p'},c)} \geq Th_s$ **then**
    if $|ST| = dimST$ **then**
      $\langle id_{\hat{p}}, \hat{c}, rank_{(id_{\hat{p}},\hat{c})}\rangle := \arg\min_{\langle id_{p''},c'',rank_{(id_{p''},c'')}\rangle \in ST} \; rank_{(id_{p''},c'')}$;
      if $rank_{(id_{p'},c)} > rank_{(id_{\hat{p}},\hat{c})}$ **then**
        $ST := ST \backslash \langle id_{\hat{p}}, \hat{c}, rank_{(id_{\hat{p}},\hat{c})}\rangle$;
        $ST := ST \bigcup \{\langle id_{p'}, c, rank_{(id_{p'},c)}\rangle\}$;
      end
    else
      $ST := ST \bigcup \{\langle id_{p'}, c, rank_{(id_{p'},c)}\rangle\}$;
    end
  end
end

Algorithm 4: Building semantic links

5.3.1. Semantic links maintenance

The construction of a SON brings benefits in terms of service discovery since it allows one to forward queries more efficiently. However, a SON brings an intrinsic cost of both construction and maintenance. ERGOT deals with both these aspects in the following way. The construction of the SON, as discussed earlier, occurs by piggybacking on the service publishing mechanism provided by the DHT with almost no additional cost. As for semantic link maintenance, ERGOT relies on a DHT, for which there exists a *stabilization protocol* to reorganize keys when peers join or leave the network. Therefore, without loss of generality, we can see semantic link maintenance in the SON as a similar process to the stabilization protocol. Indeed, a semantic link maintenance protocol can be run at the same time of the stabilization protocol by exchanging information between peer to check the value of a link or whether it is still valid.

5.4. Service query processing

The ERGOT architecture discussed so far enables service discovery based either on the SON, the underlying DHT overlay or a combination of both. In each case, we assume that a service query has the form

$$SQ = \langle id_{SQ}, id_p, R, TTL\rangle$$

where id_{SQ} is a unique id assigned to the query, id_p is the id of the peer starting the query, R is a request of the form $R = \langle \mathcal{C}, \mathcal{O}\rangle$ and TTL is the Time-to-Live, which represents the maximum number of hops that the query can traverse.

The semantic search strategy is straightforward: a peer p that receives a request begins by matching it within its local service profiles; it then forwards the request over its semantic links. The peer in order to establish to which semantic neighbors the query has to be forwarded, computes the similarity between the query and the semantic links in its ST. Algorithm 5 describes how queries are processed and propagated on the SON. Alternatively,

Input: $SQ = \langle id_{SQ}, id_p, R, TTL \rangle$;
Th_f: a forward threshold ;
Output: RES: set of services ranked w.r.t R ;

```
1  RES := ∅ ;
2  if notAlreadyAnswered(SQ.id_SQ) then
3      RES := RES ⋃ checkLocalAnswers(SQ.R) ;
4      if p.id ≠ SQ.id_p then
5          SQ.TTL := SQ.TTL − 1 ;
6          replyMsg(SQ.id_p, SQ.id_SQ, RES) ;
7      end
8      if SQ.TTL > 0 then
9          foreach c ∈ SQ.R.C do
10             foreach sl ∈ ST do
11                 if Csim(c, sl.c') · sl.rank > Th_f then
12                     forwardMsg(SQ, sl.id_p') ;
13                 end
14             end
15         end
16     end
17 end
```

Algorithm 5: Processing service query

the peer may choose to use the underlying DHT to route the request; this may happen, for example, when there are no semantic neighbors that satisfy the strength criteria. In this case, the request becomes a collection of standard $get(c_i)$, with $c_i \in R.\mathcal{C} \subseteq$ CO. This means that, for every CO concept c in a request, a *get(c)* operation is invoked in the DHT. The routing in this case involves $O(k \log N)$ hops where k is the number of concepts in the request R. Note that this approach relies on exact matching of concepts, and is similar to that adopted by the SPiDeR system [28]. However, in ERGOT, upon reaching the peer responsible for the concept in the request a similarity-based search can additionally be performed between the request and peer's service profiles. This allows service profiles to be ranked as described in Section 4.4.

Returning to our previous example, a peer could perform a request such as *get*(`Automobile Merchant`). The request, through the DHT routing, reaches node N_{13} responsible for the key associated with the concept `Automobile Merchant`. At this point, N_{13}, by using function *RPsim* can rank and return the services annotated to `Automobile Merchant` (i.e., SP_3, SP_1 and SP_{11}). Hence, services annotated to the same category are differentiated, via *RPsim*, since they possibly contain different annotations in terms of DO concepts.

6. Evaluation

ERGOT has been evaluated using a custom-made simulator written in Java. The simulator covers both the DHT publication phase and the SON construction. In the current implementation, the Chord DHT [33] is used. Fig. 8 shows the architecture of the ERGOT simulator. A core component of this architecture is the `Similarity Assessor`, which embeds the similarity functions described in Section 4.4. In this implementation, the WordNet ontology [21] has been used both as *Category Ontology* (*CO*) and *Domain Ontology* (*DO*). ERGOT offers both a *requester* and a *provider* perspective. Provider peers exploit the underlying DHT mechanism (i.e., the *put(id,object)* primitive) to publish services, whereas service retrieval can be performed by using the DHT, the SON or a combination of both.

6.1. Experimental setup and evaluation methodology

The experiments were focused on evaluating both accuracy-related and traffic-related performance parameters, which are summarized in Tables 3 and 4, respectively. For a

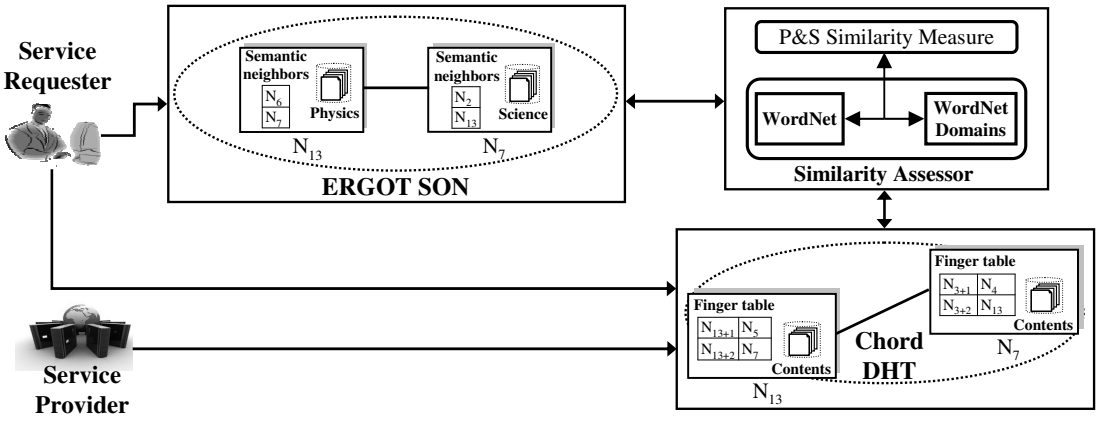

Figure 8: ERGOT simulator architecture

Table 3: Accuracy-related parameters

Parameter	Meaning
R_r^s	Relevant services retrieved
R_e^s	Existing relevant services
$\mathcal{R}^s$	Service recall = $\frac{R_r^s}{R_e^s}$

Table 4: Traffic-related parameters

Parameter	Meaning
R_c^p	Relevant peers contacted
N_c^p	Not relevant peers contacted
R_e^p	Existing relevant peers
$\mathcal{P}^p$	Peer precision = $\frac{R_c^p}{R_c^p+N_c^p}$
$\mathcal{R}^p$	Peer recall = $\frac{R_c^p}{R_e^p}$
$\mathcal{F}^p$	Peer F-measure = $\frac{2\times P_c^p \times R_c^p}{P^p+R^p}$

given query, the basic accuracy-related performance parameters are the number of relevant services retrieved (R_r^s) and the number of existing relevant services (R_e^s). The *service recall* ($\mathcal{R}^s$) is then calculated as the ratio between R_r^s and R_e^s. The basic traffic-related parameters are the number of relevant peers contacted (R_c^p), and the number of not-relevant peers contacted (N_c^p).

A peer is said to be relevant if it provides some service matching the query, whereas it is not relevant if it has been contacted but does not have any service relevant with respect to the query. Both R_c^p and N_c^p are compared to the number of existing relevant peers (R_e^p) to derive three performance indicators.

- *Peer precision* ($\mathcal{P}^p$): the ratio of contacted peers relevant to the query.
- *Peer recall* ($\mathcal{R}^p$): the ratio of the peers relevant to the query that have been successfully contacted.
- *Peer F-measure* ($\mathcal{F}^p$): the harmonic mean of peer precision ($\mathcal{P}^p$) and peer recall ($\mathcal{R}^p$).

Each query in each of the considered datasets (see Section 6.2) is executed multiple times (by 15% of peers) starting each time from a different peer. As the overall result we took the average values over all the executions.

The parameter values have been trained on a subset of the datasets considered, by choosing the combination of their values that maximized the achieved recall. There are four parameters involved in the evaluation. The parameters α, β, and γ are used to weight the contribution of operation name, inputs, and outputs, respectively. Moreover there is also the forward threshold t_h, which is used to decide if a request has to be forwarded to the semantic neighbors of the peer that receives it. As for the first three parameters, in all the evaluations we used $\alpha+\beta+\gamma=1$, with $\alpha = 0.65$. The relatively high value assigned to α was chosen after having observed that high values for this parameter produce the best results in terms of recall. The remaining weight (0.35) was equally divided between β and γ, since

in our experiments we did not observe a significant difference in terms of recall when giving a higher weight to β or to γ. The threshold t_h was set to 0.5 in all experiments since it provided a good trade-off between recall and network traffic.

6.2. Dataset and queryset generation

In order to evaluate ERGOT, a synthetic dataset of annotated services was built by exploiting the WordNet ontology and WordNet domain [2], which categorizes WordNet synsets into domains. The categorization into domains of WordNet concepts has been very useful since we noted that by exploiting a simple annotation generator (i.e., a random concept generator) it was very common to have service profiles annotated with disparate concepts, resulting in not reliable datasets. Specifically, for each element of the service profile, one or more ontology concepts were generated. Four different simulation configurations, summarized in Table 5, were exploited during the evaluation. This allowed us to evaluate the ERGOT behavior in scenarios characterized by different network sizes and application domains. These simulations are characterized by a different number of services, peers and size of the (ST). Note that in Table 5, $dim.ST$ represents the maximum number of entries that can be stored in the ST and not the number of semantic neighbors. In fact, the ST can include the same peer several times depending on the particular concept on which the semantic link has been created (see Algorithm 4).

Table 5: Simulation configurations

Scenario	#peers	#services	dim. ST	Domain
Sim1	500	1,000	30	Food
Sim2	1,500	3,000	150	Biology
Sim3	3,000	6,000	300	Commerce
Sim4	5,000	10,000	450	Finance

Table 6: Dataset generation parameters

Parameter	Range
# CO annotations	1-5
# of operations	1-3
# of inputs	1-3
# of outputs	1-3

Table 6 summarizes the parameters exploited to generate the service datasets used in the various simulation configurations. Services are assigned to peers according to a Zipf distribution with skewing factor equal to 0.5. An additional software module has been adopted for generating the service requests. Fig. 9 reports the architecture of the software module exploited to generate the dataset.

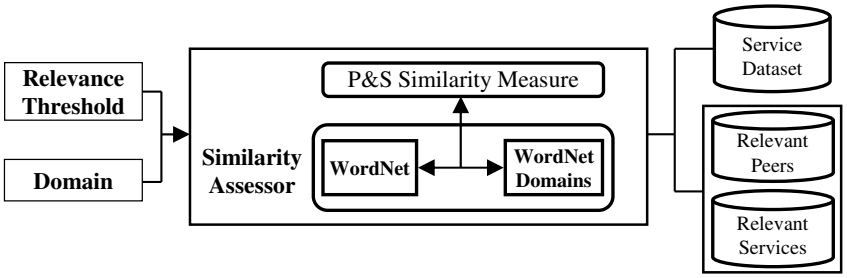

Figure 9: Queryset generation

Four different query sets, each containing 20 service requests, were constructed. Each query is referred to as qX-$simY$ in the following, where X indicates the number of the query and Y the simulation in which this query is exploited. Therefore, we generated 80 different queries, 20 for each simulation scenario reported in Table 5.

Each service request is generated and expressed in the same form of a service profile by specifying a `domain` and a `relevance threshold`. The relevance threshold enables us to construct the *relevance set*, that is, the set of services that are relevant with respect to the request. This set is constructed by resolving the request as if services were stored in a centralized way. Besides, for each relevant service in the relevant set, the set of `relevant`

`peers`, i.e., the set of (provider) peers where the services can be found, is also stored. This allows us to evaluate the number of relevant and not-relevant peers contacted during each request. Services are published on the DHT thus enabling the SON construction as described in Algorithm 4. Hence, each query in the queryset is resolved according to the ERGOT mechanism described in Algorithm 5.

6.3. Experimental results

In the following we analyze the performance of ERGOT, also compared with that of a Chord DHT and a Gnutella-like network.

6.3.1. Evaluating ERGOT solo

Figs. 10-13 depict, for each simulation configuration (see Table 5), the average service recall as a function of the TTL. The value of the TTL used in all the experiments presented hereafter is equal to 4. Indeed, in our experimental settings, this value permits us to achieve a good balance between network coverage and generated number of messages. Error bars represent the 95% confidence interval.

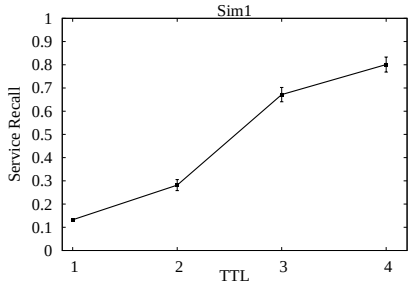

Figure 10: ERGOT: $\mathcal{R}^s$ vs. TTL in *Sim1*

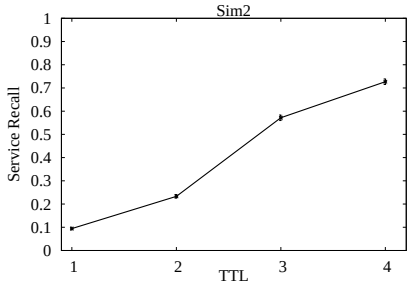

Figure 11: ERGOT: $\mathcal{R}^s$ vs. TTL in *Sim2*

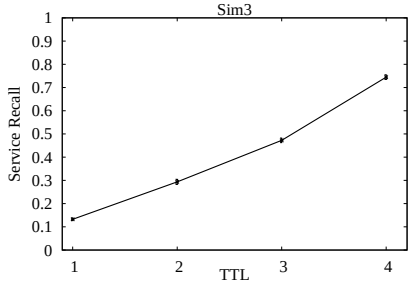

Figure 12: ERGOT: $\mathcal{R}^s$ vs. TTL in *Sim3*

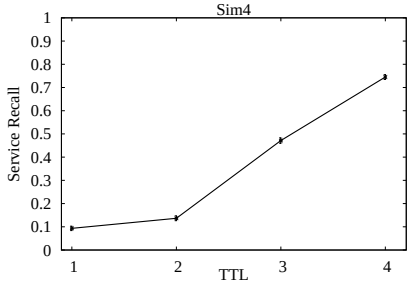

Figure 13: ERGOT: $\mathcal{R}^s$ vs. TTL in *Sim4*

As can be noted, the recall increases with increasing TTL. In particular, for all the simulations, it can be noted that with a TTL lower than 3 the recall does not reach significant values. This is because a low value of TTL does not allow one to contact semantic neighbors, which are relatively distant from the peers that propagate the query. Moreover, when TTL is equal to 3 it can be noted that with smaller simulation settings (i.e., *Sim1* and *Sim2*) the values of recall are above 0.50 (i.e., only half the relevant services are returned) whereas, with larger simulation settings, values reach a maximum of 0.47 (0.4725 for *Sim3* and 0.4715 for *Sim4*). More interestingly, with TTL equal to 4 all the simulations seem to converge toward very good values of recall ranging from 0.801 with *Sim1* to 0.745 with *Sim4*. This shows that with TTL greater than 3 the system performs well regardless of the size of the networks and number of services. Moreover, since the evaluations covered different domains, the performance does not seem to be dependent on a particular dataset.

Figs. 14-17 report the number of relevant (R_c^p) and not-relevant (N_c^p) peers contacted, out of all the existing relevant peers (R_e^p), for each simulation setting. As shown in the figures, N_c^p is low compared to R_c^p in all scenarios. For example, in *Sim1* the average value

of N_c^p over the 20 queries is 14.7, while the average value of R_c^p is 72.6. In *Sim2* the average values of N_c^p and R_c^p are 39.2 and 201, respectively. Similar results can be observed in *Sim3* and *Sim4*.

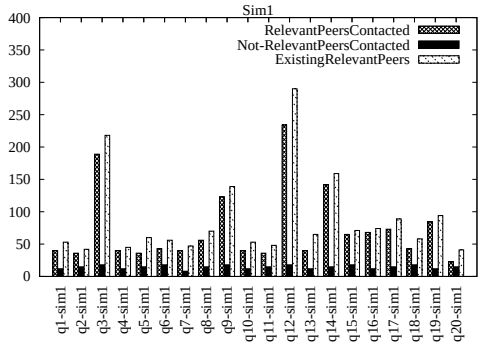

Figure 14: ERGOT: R_c^p, N_c^p and R_e^p in *Sim1*

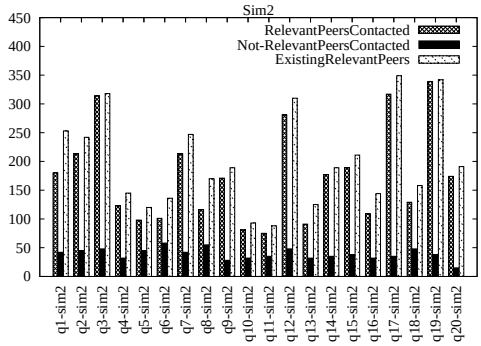

Figure 15: ERGOT: R_c^p, N_c^p and R_e^p in *Sim2*

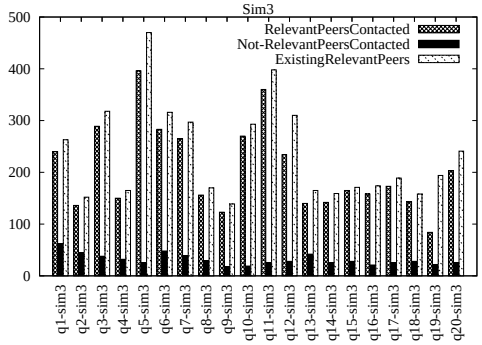

Figure 16: ERGOT: R_c^p, N_c^p and R_e^p in *Sim3*

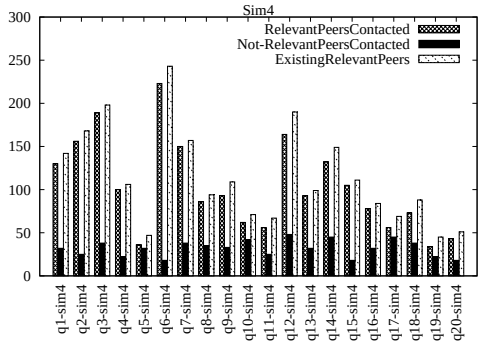

Figure 17: ERGOT: R_c^p, N_c^p and R_e^p in *Sim4*

Figs. 18-21 summarize the peer precision ($\mathcal{P}^p$), recall ($\mathcal{R}^p$) and F-measure ($\mathcal{F}^p$) parameters for the various simulations. As can be noted, ERGOT obtained notable results in all scenarios. In particular, in *Sim1*, where service profiles have been annotated to concepts in the *Food* domain, the values of $\mathcal{F}^p$, which combines $\mathcal{P}^p$ and $\mathcal{R}^p$, are always above 0.5. The lowest value, namely 0.58, is obtained for query *q20-sim1*. By analyzing the structure of this query in terms of relevant and not-relevant peers and services, we noted that *q20-sim1* is a very selective query in the sense that service profiles related to this query are few. In particular *q20-sim1* has the following structure: 1 *CO* annotation and 2 operations with 1 input and 1 output respectively. By recalling that the annotations to the *CO* (see Algorithm 4) are exploited by the semantic-based discovery process, in this case it is only possible to route the query via the only (or related) concept(s) expressing the *CO* annotation. This means that it is very difficult to route the request toward relevant peers since only few semantic links can be exploited. Despite this, the value of $\mathcal{P}^p$ is 0.61 and that of $\mathcal{R}^p$ is 0.56. On the other hand, the highest value of $\mathcal{F}^p$, namely 0.9, is obtained for the query *q14-sim1*. In this case, the request contains 4 *CO* annotations, and then it enables multiples semantic links to be exploited.

In *Sim2*, where the dataset contains concepts in the *Biology* domain, the lowest value of $\mathcal{F}^p$ is 0.68 obtained in query *q8-sim2* and query *q6-sim2*. Also in this case the structure of the query contained only 1 CO annotation. The highest value of correlation, that is 0.94, is obtained for query *q19-sim2*. In this case the query contained 4 *CO* annotations with 3 operations.

In *Sim3*, where the domain is *Commerce*, the lowest value of $\mathcal{F}^p$ is 0.56 obtained in query *q19-sim3*. In this case $\mathcal{P}^p$ is 0.79 and $\mathcal{R}^p$ is 0.43. This latter value indicates that the

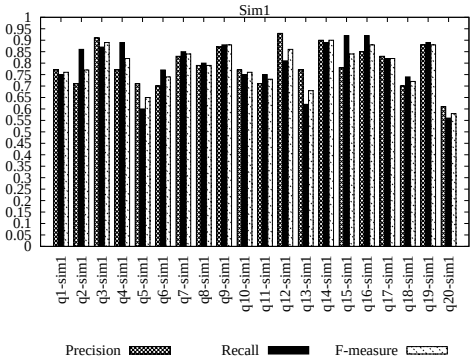

Figure 18: ERGOT: $\mathcal{P}^p$, $\mathcal{R}^p$ and $\mathcal{F}^p$ in *Sim1*

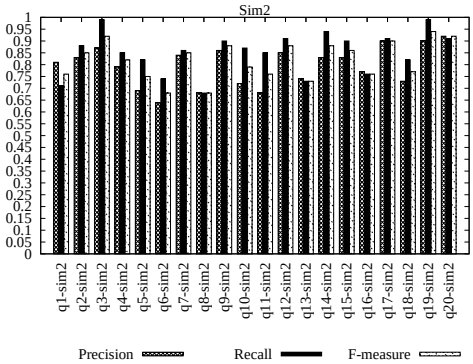

Figure 19: ERGOT: $\mathcal{P}^p$, $\mathcal{R}^p$ and $\mathcal{F}^p$ in *Sim2*

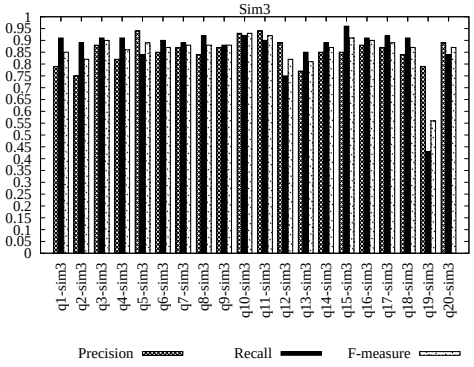

Figure 20: ERGOT: $\mathcal{P}^p$, $\mathcal{R}^p$ and $\mathcal{F}^p$ in *Sim3*

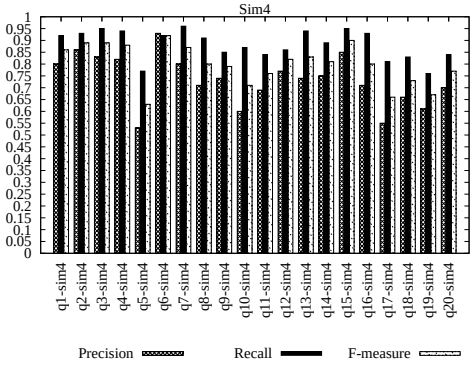

Figure 21: ERGOT: $\mathcal{P}^p$, $\mathcal{R}^p$ and $\mathcal{F}^p$ in *Sim4*

system correctly contacts 43% of all existing relevant peers. Despite this value, however, ERGOT is able to retrieve about 77% of all existing relevant services (see Fig. 20). The remaining part (i.e., 23%) is stored by the relevant peers not contacted. The highest value of $\mathcal{F}^p$ is 0.93 obtained in *q10-sim3* with $\mathcal{P}^p$ equal to 0.93 and $\mathcal{R}^p$ equal to 0.92.

Finally, in *Sim4* focused on the *Finance* domain, the lowest value of $\mathcal{F}^p$ is 0.66 obtained in *q17-sim4* where the value of $\mathcal{P}^p$ is 0.55 and that of $\mathcal{R}^p$ is 0.81. In this case, ERGOT obtained a low value of peer precision, meaning that the semantic path followed to forward the request contained several not-relevant peers, which causes a waste of bandwidth. However, about 81% of all relevant peers have been contacted. In this case, the highest $\mathcal{F}^p$ is 0.92 obtained in *q6-sim4*.

The discussion above results in two main conclusions. On the one hand, having carefully annotated service profiles provides better results since it enables us to construct a SON with more and useful semantic links. On the other hand, the performance of ERGOT does not seem to be affected when the size of the network increases. In fact, the result obtained in *Sim4* are comparable or better than those obtained in *Sim1*. Besides, the choice to perform evaluations using different domains confirmed that the performance of the system is not dataset dependent.

We conclude the evaluation of the traffic-related performance of the system by reporting in Table 7 the number of *duplicate messages* for each simulation. A message is considered to be a "duplicate" the second (or further) time that it passes from the same node. The "duplicate messages rate" is the percentage of duplicate messages of the total number of messages, and it is used as an indicator of the bandwidth waste generated by the algorithm. The duplication rate is low and, as expected, it increases with the size of the simulation setting. The maximum duplicate rate is 17.4% in the fourth scenario, where the network is composed by 5000 peers and 10000 services. Overall, it can be observed that ERGOT obtains significant values of recall in different network configurations. The cost paid in

Table 7: Duplicate message rates

Simulation	Duplicate messages rate (%)
Sim1	3.1
Sim2	5.2
Sim3	8.6
Sim4	17.4

terms of bandwidth waste, given by the duplicate message rate, remains acceptable. This underlines how the ERGOT approach based on semantic links, which connect peers with similar peers to form an overlay, is valuable.

6.3.2. ERGOT versus Chord

In this section, the semantic discovery capability of ERGOT is compared with the DHT exact-matching mechanism implemented by Chord. Figs. 22-25 report the service recall ($\mathcal{R}^s$) for both ERGOT and the Chord DHT, by considering the simulation scenarios described in Table 5. The evaluation was performed by using the standard *get(key)* primitive

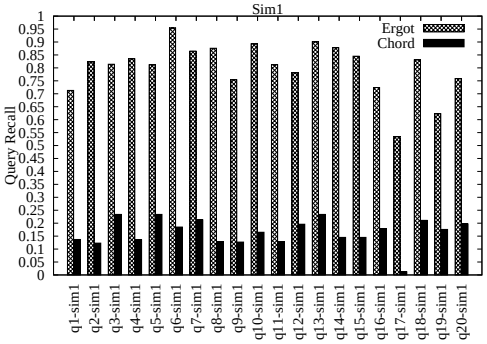

Figure 22: ERGOT vs. DHT: $\mathcal{R}^s$ in *Sim1*

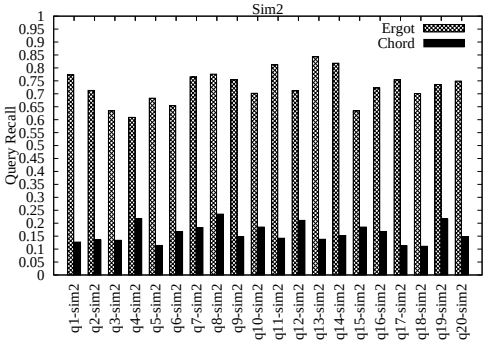

Figure 23: ERGOT vs. DHT: $\mathcal{R}^s$ in *Sim2*

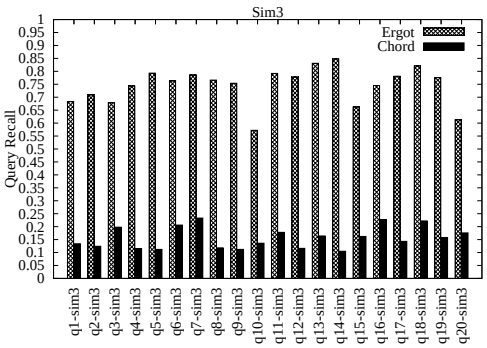

Figure 24: ERGOT vs. DHT: $\mathcal{R}^s$ in *Sim3*

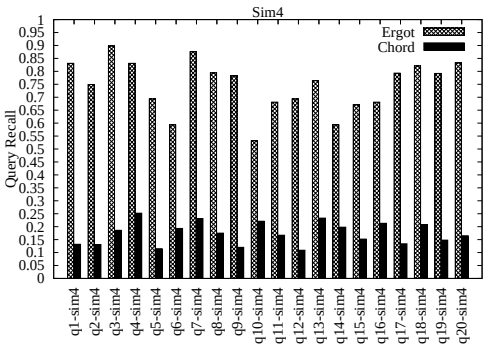

Figure 25: ERGOT vs. DHT: $\mathcal{R}^s$ in *Sim4*

for Chord, where *key* is the hash value of the *CO* concept in the request. In the case of multiple *CO* concepts, multiple *get(key)* operations are executed and, as final result, the union between all the retrieved service profiles is considered.

As can be observed, in all the scenarios ERGOT obtained values of recall significantly higher than those obtained by Chord. This is mainly due to the fact that Chord is able to perform only exact lookup, which is a significant limitation when performing semantic-based service discovery where users can express more general requests. Table 8 reports the average values of service recall (i.e., $\mathcal{R}^s$) for each of the simulation scenarios for ERGOT and Chord. As can be noted, the difference between the two systems remains quite constant

Table 8: Average $\mathcal{R}^s$ in the four simulation scenarios

Simulation	**ERGOT** $\mathcal{R}^s$	**Chord** $\mathcal{R}^s$
Sim1	0.8	0.16
Sim2	0.72	0.16
Sim3	0.74	0.15
Sim4	0.74	0.17

in the various simulation. Chord obtained almost the same $\mathcal{R}^s$ value (around 0.16) in all the scenarios. ERGOT obtained $\mathcal{R}^s$ values ranging from 0.72 to 0.8. ERGOT enables us to rank results and, therefore, in case several services match the user request in terms of *CO* annotations, these can be further differentiated on the basis of the extent to which they match also in terms of operations with related input and outputs.

This is not the case for Chord, where services can only "exactly" be matched by exploiting *CO* annotations. The above discussion is meant to underline how Chord, and any DHT in general, is not appropriate for performing semantic-based service discovery where a service requester does not know exactly what s/he wants in the sense that s/he can also be interested in services semantically similar to the request. On the other hand, if we consider the number of messages used to perform service discovery, while Chord has an upper bound given by $O(\log N)$, where N is the number of nodes, ERGOT does not have such property since semantic links are obtained by a criterion of semantic similarity and not algorithmically as in Chord where each peer maintains $O(\log N)$ neighbors. In particular, the process needed to build a SON can be very costly in terms of number of messages since each peer has to discover in some way the "most similar" peers to establish semantic links with. However, in the case of ERGOT, the usage of Chord enables us to automatically build the SON by exploiting the normal service publishing activity in the DHT.

6.3.3. ERGOT versus Gnutella-like networks

This section compares ERGOT with an unstructured network (in the manner of Gnutella) in terms of service recall ($\mathcal{R}^s$). This has been done with the aim of further investigating how semantic links affect the performance as compared to links built without taking into account semantics. For each simulation scenario, we constructed an unstructured network with the same average degree per node as that of ERGOT expressed via semantic neighborhood relations. For each configuration scenario, the comparison between ERGOT and Gnutella was done as follows. First, we ran the search using ERGOT and measured its performance parameters (service recall, peer precision, etc.) as well as the number n of messages generated by the discovery process. Secondly, we ran the search using Gnutella, but we stopped the algorithm as soon as n messages were generated; thus, the performance parameters of Gnutella were measured at the time of this stop. This approach was necessary because using the Gnutella flooding technique we would have obtained, in most scenarios, 100% service recall but with a huge number of messages, which would have made the discussion of the comparison between the two algorithms very poor. Figs. 26-29 report the result of the evaluation for each configuration scenario.

As can be noted in all the simulations, ERGOT achieves higher values of $\mathcal{R}^s$ compared to a Gnutella-like network. In *Sim1*, the highest value of $\mathcal{R}^s$ reached by ERGOT is 0.954 in *q6-sim1*, where Gnutella reaches 0.599. Gnutella achieved the best $\mathcal{R}^s$, that is 0.801, in *q11-sim1* where ERGOT obtained 0.812. In *q6-sim1*, while using the same number of messages, ERGOT is better than Gnutella by about 59%. In *q11-sim1* the two approaches are quite close, which means that the unstructured topology, with respect to *q11-sim1*, is comparable to the semantic topology built by ERGOT.

In *Sim2*, ERGOT achieves the highest $\mathcal{R}^s$ (0.843) in *q13-sim2*. Gnutella obtained 0.434 for the same query. Also in this case, the semantic approach followed by ERGOT brings

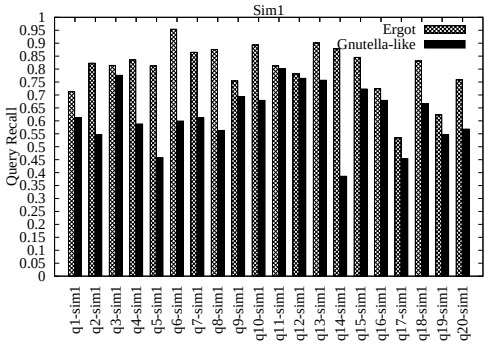

Figure 26: ERGOT vs. Gnutella-like: $\mathcal{R}^s$ in *Sim1*

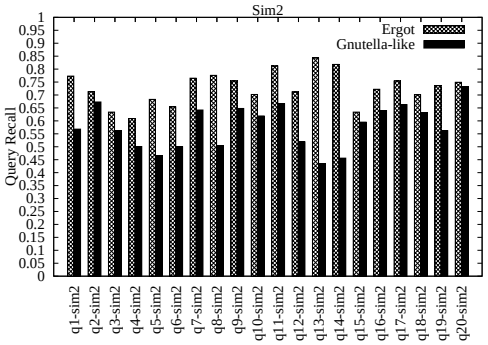

Figure 27: ERGOT vs. Gnutella-like: $\mathcal{R}^s$ in *Sim2*

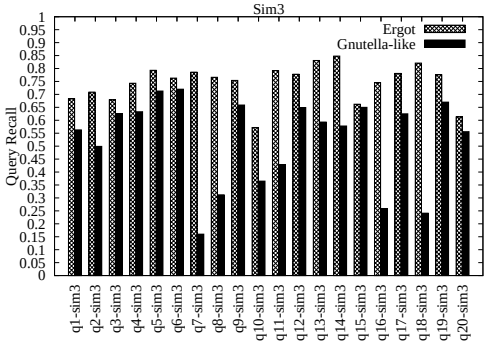

Figure 28: ERGOT vs. Gnutella-like: $\mathcal{R}^s$ in *Sim3*

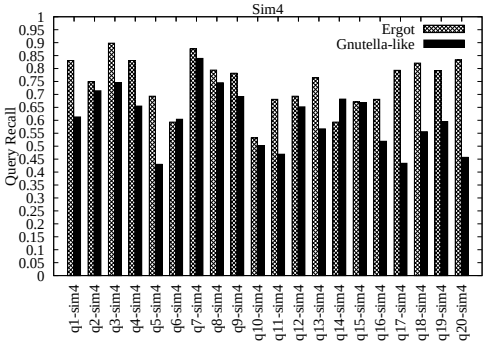

Figure 29: ERGOT vs. Gnutella-like: $\mathcal{R}^s$ in *Sim4*

better results. The lowest $\mathcal{R}^s$, namely 0.609, is obtained in *q4-sim2* where Gnutella obtained 0.501. Notable is the difference between the two systems in *Sim3*, where the average value of $\mathcal{R}^s$ is 0.74 and 0.52 for ERGOT and Gnutella, respectively. In this case the improvement obtained using ERGOT is about 42%. Finally, in *Sim4* the average $\mathcal{R}^s$ is 0.74 for ERGOT and 0.60 for Gnutella, respectively. In this latter scenario, the most notable difference can be noted for *q20-sim4* where ERGOT obtained a value of $\mathcal{R}^s$ equal to 0.83 while Gnutella obtained 0.45.

The results above underline how the semantic-driven routing of the queries adopted in ERGOT brings better results. The *Sim3* and *Sim4* configurations, which are characterized by a larger number of peers and services than *Sim1* and *Sim2*, demonstrate that ERGOT performs well even in large network scenarios. This is particularly clear from the *q20-sim4* results, where ERGOT almost doubles the service recall.

As a final evaluation, we assessed the traffic-related parameters for Gnutella in *Sim4*, which is the largest configuration on which ERGOT has also been evaluated (see Figs. 17 and 21). In particular, Fig. 30, shows the number of relevant and not-relevant peers contacted over all the existing relevant peers, while Fig. 31 shows the peer precision, recall and F-measure.

By comparing the results shown in Fig. 30 (Gnutella-like) with those presented in Fig. 17 (ERGOT), we observe that a Gnutella-like network contacts a significantly higher number of not-relevant peers as compared to ERGOT, for all the queries considered. Since the total number of messages used by Gnutella and ERGOT was fixed in our simulations, Gnutella contacts a lower number of relevant peers as compared to ERGOT. This leads to poor performance of Gnutella as compared to ERGOT in terms of peer precision ($\mathcal{P}^p$), recall ($\mathcal{R}^p$), and F-measure ($\mathcal{F}^p$), as can be observed by comparing the results in Fig. 31 (Gnutella-like) with those presented in Fig. 21 (ERGOT) and discussed earlier. In particular, we observe that the average value of ($\mathcal{F}^p$) over all the 20 queries is 0.80 for ERGOT, while it is only 0.57

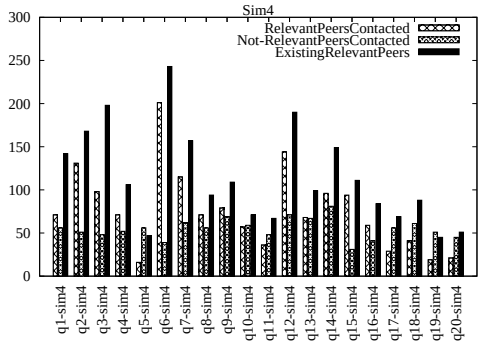

Figure 30: Gnutella-like: R_c^p, N_c^p and R_e^p in *Sim4*

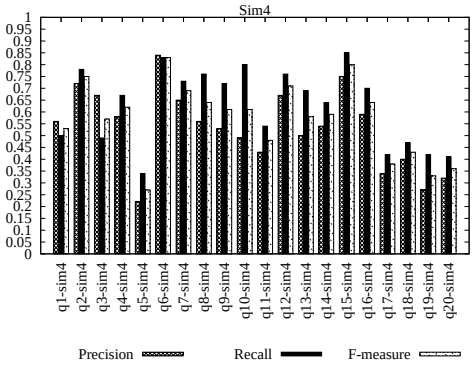

Figure 31: Gnutella-like: $\mathcal{P}^p$, $\mathcal{R}^p$ and $\mathcal{F}^p$ in *Sim4*

for Gnutella. From the results discussed above, we conclude that by exploiting semantic links, as is done in ERGOT, the search performance improves significantly as compared to the use of semantic-free links as in Gnutella-like overlays, in terms of both accuracy and network traffic.

7. Concluding Remarks and Future Research Directions

DHT-based systems allow for performing exact-match searches with logarithmic performance bounds, but do not fit well with semantic similarity search models, where services are described using multiple annotations. The ERGOT system proposed in this paper enables semantic-driven query answering in DHT-based systems by building a SON over a DHT. The SON in ERGOT is built incrementally, as a result of peer interactions that occur naturally in DHTs during advertising of a new service description, as well as during search. To enable semantic service matchmaking over the resulting DHT/SON network, a measure of semantic similarity between service descriptions has been also defined. The strategy implemented in the ERGOT system has been extensively evaluated in different network scenarios and it has been compared to Chord and Gnutella strategies. Experimental results demonstrate the efficiency of the combined DHT and SON strategy in terms of both accuracy of search and network traffic.

There are some aspects that provide room for future work. First, it would be worth investigating and comparing different strategies to perform service matchmaking. Even if service matchmaking is not the main aim of the present paper, it can be worth investigating how to combine similarity-based matchmaking with other indicators such as quality of service. Another interesting topic for future research is the ability to work with multiple ontologies, where peers have their own ontologies to annotate service profiles. In this case, a strategy to perform ontology matching has to be adopted. In this respect, the combination of DHTs and SONs could be useful since it can be exploited to create a distributed registry of matchings to which peers in the network can refer to link their own ontologies with similar ones.

Acknowledgements

We would like to acknowledge Paolo Missier and Carole Goble for their contribution to the early stage of this work.

References

[1] S. Banerjee, S. Basu, S. Garg, Sukesh Garg, S. Lee, P. Mullan, and P. Sharma. Scalable Grid Service Discovery Based on UDDI. In *MGC*, pages 1–6, 2005.

[2] L. Bentivogli, P. Forner, B. Magnini, and E. Pianta. Revising WordNet Domains Hierarchy: Semantics, Coverage, and Balancing. In *COLING 2004 Workshop on Multilingual Linguistic Resources*, pages 101–108, 2004.
[3] D. Bianchini, V. De Antonellis, and M. Melchiori. Flexible Semantic-Based Service Matchmaking and Discovery. *World Wide Web*, 11(2):227–251, 2008.
[4] D. Bianchini, V. De Antonellis, and M. Melchiori. P2P-SDSD: On-the-Fly Service-Based Collaboration in Distributed Systems. *IJMSO*, 5(3):222–237, 2010.
[5] D. Bianchini, V. De Antonellis, M. Melchiori, and D. Salvi. On-the-Fly Collaboration in Distributed Systems Through Service Semantic Overlay. In *iiWAS*, pages 407–410, 2008.
[6] O. Corcho, P. Alper, I. Kotsiopoulos, P. Missier, S. Bechhofer, and C. A. Goble. An Overview of S-OGSA: A Reference Semantic Grid Architecture. *J. Web Sem.*, 4(2):102–115, 2006.
[7] A. Crespo and H. Garcia-Molina. Semantic Overlay Networks for P2P Systems. In *AP2PC*, pages 1–13, 2004.
[8] E. Deelman, D. Gannon, M. S. Shields, and I. Taylor. Workflows and e-Science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009.
[9] X. Dong, A. Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Simlarity Search for Web Services. In *VLDB*, pages 372–383, 2004.
[10] F. Emekçi, O. D. Sahin, D. Agrawal, and A. El Abbadi. A Peer-to-Peer Framework for Web Service Discovery with Ranking. In *ICWS*, pages 192–199, 2004.
[11] L. Gong. JXTA: A Network Programming Environment. *IEEE Internet Computing*, 5(3):88–, 2001.
[12] Z. Kaoudi, M. Koubarakis, K. Kyzirakos, M. Magiridou, I. Miliaraki, and A. Papadakis-Pesaresi. Publishing, Discovering and Updating Semantic Grid Resources using DHTs. In *CoreGRID Workshop on Grid Programming Model, Grid and P2P Systems Architecture, Grid Systems, Tools and Environments*, 2007.
[13] F.B. Kashani, C. Chen, and C. Shahabi. WSPDS: Web Services Peer-to-Peer Discovery Service. In *ICOMP*, pages 733–743, 2004.
[14] T. Kawamura, J. De Blasio, T. Hasegawa, M. Paolucci, and K. Sycara. Public Deployment of Semantic Service Matchmaker with UDDI Business Registry. In *ISWC*, pages 752–766, 2004.
[15] M. Klusch, B. Fries, and K. P. Sycara. OWLS-MX: A Hybrid Semantic Web Service Matchmaker for OWL-S services. *J. Web Sem.*, 7(2):121–133, 2009.
[16] M. Klusch and F. Kaufer. WSMO-MX: A Hybrid Semantic Web Service Matchmaker. *Web Intelligence and Agent Systems*, 7(1):23–42, 2009.
[17] V. Kunal, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller. METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services. *Inf. Technol. and Manag.*, 6(1):17–39, 2005.
[18] L. Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. In *WWW*, pages 331–339, 2003.
[19] M. Li, B. Yu, O. Rana, and Z. Wang. Grid Service Discovery with Rough Sets. *IEEE TKDE*, 20(6):851–862, 2008.
[20] Y. Li, F. Yang, K. Shuang, and S. Su. A Semantic Peer-to-Peer Overlay for Web Services Discovery. In *SOFSEM (1)*, pages 750–760, 2007.
[21] G. Miller. WordNet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4):235–312, 1990.
[22] M. Paolucci, K. P. Sycara, T. Nishimura, and N. Srinivasan. Using DAML-S for P2P Discovery. In *ICWS*, pages 203–207, 2003.
[23] M. P. Papazoglou, B. J. Krämer, and J. Yang. Leveraging Web-services and Peer-to-Peer networks. In *CAiSE*, pages 485–501, 2003.
[24] G. Pirrò and N. Seco. Design, Implementation and Evaluation of a new Semantic Similarity Metric Combining Features and Intrinsic Information Content. In *ODBASE*, pages 1270–1287, 2008.
[25] G. Pirrò, M. Ruffolo, and D. Talia. SECCO: On Building Semantic Links in Peer-to-Peer Networks. *J. Data Semantics*, 12:1–36, 2009.
[26] G. Pirrò, P. Trunfio, D. Talia, P. Missier, and C. A. Goble. ERGOT: A Semantic-Based System for Service Discovery in Distributed Infrastructures. In *CCGRID*, pages 263–272, 2010.
[27] P. Plebani and B. Pernici. URBE: Web Service Retrieval Based on Similarity Evaluation. *IEEE TKDE*, 21(11):1629–1642, 2009.
[28] O. Sahin, C. Gerede, D. Agrawal, A. El Abbadi, O. I. Ibarra, and J. Su. SPiDeR: P2P-Based

Web Service Discovery. In *ICSOC*, pages 157–169, 2005.
[29] B. Sapkota, L. Vasiliu, I. Toma, D. Roman, and C. Bussler. Peer-to-Peer Technology Usage in Web Service Discovery and Matchmaking. In *WISE*, pages 418–425, 2005.
[30] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services. In *P2P*, pages 104–111, 2002.
[31] C. Schmidt and M. Parashar. A Peer-to-Peer Approach to Web Service Discovery. *World Wide Web*, 7(2):211–229, 2004.
[32] N. Seco, T. Veale, and J. Hayes. An Intrinsic Information Content Metric For Semantic Similarity in WordNet. In *ECAI*, pages 1089–1090, 2004.
[33] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 149–160. ACM Press, 2001.
[34] L. Vu, M. Hauswirth, and K. Aberer. Towards P2P-based Semantic Web Service Discovery with QoS Support. In *BPS*, 2005.
[35] Y. Zhu and Y. Hu. Efficient Semantic Search on DHT Overlays. *J. Parallel Distrib. Comput.*, 67:604–616, May 2007.