DOI 10.12694/scpe.v15i1.968
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 15, Number 1, pp. 89–99. http://www.scpe.org © 2014 SCPE

TTL-CHORD: A CHORD-BASED APPROACH FOR SEMANTIC WEB SERVICES DISCOVERY

MOHAMED GHARZOULI [*], YOUCEF MESSELEM [†], AND MOHAMED ELHADI BOUNAS [‡]

Abstract. Recently, P2P-based discovery methods for semantic Web services require special attention from collaboration and interoperability in distributed computing environment. In this paper, we present an approach for semantic Web services discovery through a structured P2P network based on Chord. This protocol is chosen because it is scalable and still achieves very good resilience and proximity performance. However, because the discovery of Web services is made semantically, the search time will be significant. For this reason, we integrate a proposed algorithm into original implementation of Chord, where we mark the sent requests by a TTL (Time To Live). After this proposition, we present some experimental results wherever we analyze and discuss the influence of the TTL on the discovery operation and on the network stability.

Key words: Web services discovery, P2P, Chord, distributed algorithm

99

1. Introduction. In recent years, Peer-to-Peer (P2P) computing has emerged as a popular model for distributed computing. Quickly, P2P infrastructures are increasing important attentions from both industry field and academic field. It enables large-scale aggregation of resources (e.g. files, applications, services, storage) geographically distributed and belonging to different domains. P2P networks have received great attention due to their inherent flexibility. In comparison to traditional client/server paradigm, P2P systems offer more advantages in term of scalability and robustness [1, 15].

Since their apparition, P2P networks have been used for files exchange between different participants. Until now, this application occupies the most important traffic rate of generated requests in P2P networks. However, the increasing popularity of P2P systems for file sharing indicates general interest in resources sharing [14]. In the last few years, research on P2P systems has been quite intensive, and has produced significant results in scalability, robustness, location, distributed storage, and system measurements. P2P systems are being increasingly used in many recent application domains [20]. Among these emergent research fields, Web services based on P2P computing require special attention from collaboration and interoperability in a distributed computing environment. In this field, a particular interest has been expressed on Web services discovery, in which important research works focus on new mechanisms for large-scale discovery of Web services [17, 24].

In the same context, to involve the automatic Web services discovery, which requires a more intelligible description, the Web semantic technologies have been used to improve the automatic discovery and composition of Web services. A semantic description is more comprehensible, which facilitates the dynamic discovery of Web services. In this field, many semantic languages are proposed like OWL-S [7], WSMO [21], METEOR-S [29] and other specifications.

Consequently, the P2P- based approaches for semantic Web services discovery are the result of the convergence of three IT disciplines: Web services, semantic Web and P2P computing. In this context, several research works improve the decentralized discovering strategies of semantic Web services in the P2P networks. An important number of them focus on new methods, algorithms and frameworks to efficiently and effectively select and compose semantic Web services over large-scale P2P overlay [4, 18, 27, 34].

In comparison with centralized discovery methods of Web services (such as those based on UDDI (Universal Discovery, Description and Integration) [35]), P2P-based discovery mechanisms provide a scalable alternative to centralized repositories by distributing the Web services among all peers. The P2P-based approaches offer a decentralized and self-organizing context, where Web services interact with each other dynamically [10].

However, although the results obtained by many researchers, P2P-based discovery methods pose many challenges in terms of protocols, infrastructures, fault tolerance, scalability and performance. An important issue arising in P2P applications is how to accurately and efficiently discover the required services in P2P

[*]Department IFA, Faculty of New Technologies of Information and Communication, University Constantine 2, Algeria. (gharzouli@gmail.com).
[†]Department IFA, Faculty of New Technologies of Information and Communication, University Constantine 2, Algeria.
[‡]Department IFA, Faculty of New Technologies of Information and Communication, University Constantine 2, Algeria.

networks. A number of criteria like: the type of the P2P overlay, the network size and the number of pertinent peers involve the quality of responses.

In contrast with our previous works ([9, 10, 11]), where we have employed the unstructured P2P networks for semantic Web services discovery and composition, in this paper, we present a new distributed algorithm based on an architecture that combines hybrid and structured topologies. We concentrate our efforts only on the discovery operation, where we try to implement a distributed algorithm for semantic Web services discovery by using an ameliorated version of Chord protocol [32], in which we have marked the research requests by a TTL (Time To Live).

The rest of this paper is structured as follows. In Section 2, we briefly introduce a background about discovery methods for Web services. Section 3 describes our solution for semantic Web services discovery. In this section, we present an architecture to implement this solution and we propose a distributed algorithm for Web services discovery in a hybrid/Chord P2P network. In Section 4, we discuss some experimental results. Section 5 reviews related works and Section 6 concludes the paper and discusses further research.

2. Discovery methods for semantic Web services. Before presenting the different categories of semantic Web services discovery methods, it is important to clarify first what precisely means "Web services discovery". Although various proposals for discovering Web services are available, the understanding of discovery is different, and has often been confused with terms such as selection and matching [36]. Among little number of definitions of services discovery in the literature, the most official definition is that presented in [38]:

"The act of locating a machine-processable description of a Web service-related resource that may have been previously unknown and that meets certain functional criteria. It involves matching a set of functional and other criteria with a set of resource descriptions. The goal is to find an appropriate Web service-related resource".

Originally, the main goal of Web service technology is to define Web services in a machine-understandable way, in order to be automatically discoverable by other actors (agents or services). Consequently, to find the appropriate Web services is considered as an important key to service composition, invocation and execution. Basically, Web services are discovered by measuring the similarity between service requirements given by a service requester and service advertisements from service providers [36]. However, independently to the category of technique used to measure the similarity measurements, we can classify the Web services discovery methods into two main types: centralized and decentralized.

2.1. Centralized Methods. Centralized discovery methods (such as UDDI) present the first generation of works done on Web services architectures, where Web services are described by service interface functions and they publish their capabilities and functionalities with a registry [37]. Consequently, the centralized discovery methods are not adapted to the dynamic interactions, they restrict the scalability of flexible and dynamic environment [12, 18], induce performance bottleneck and may result in single points of failure [26]. Moreover, the centralized control of published services suffers from many problems such as high operational and maintenance cost.

In addition, even if the Web services are described semantically, one of the major problems with existing structure is that UDDI does not capture the relationships between entities in its directory and therefore is not capable of making use of the semantic information to infer relationships during search [3]. Secondly, UDDI supports search based on the high-level information specified about businesses and services only. It does not get to the specifics of the capabilities of services during matching [28].

2.2. Distributed Discovery Methods. To solve the problems of the centralized methods, other architectures are proposed to facilitate the discovery and composition of Web services. Recently, many solutions are proposed to proceed to the distributed discovery of Web services. The majority of research works illustrate P2P-based discovery methods. Many of them are related to automated discovery and composition of semantic Web services in the P2P networks. These research works are categorized according to the different types of P2P networks: unstructured (Gnutella V0.4 [2]), hybrid (Gnutella 0.6 [25]) or structured (like Chord [32]).

In this research field, several research works improve the discovering methods of Web services in the structured P2P networks. These systems provide an effective routing of a point to another of the system, and are based on a concept of local knowledge: a node does not have total knowledge, but it can approaches to the required data.

Structured P2P systems are proposed to solve a number of problems appeared in first generation of these systems (hybrid and pure P2P systems). They are based on a Distributed Hashing Table (DHT). An example of a structured P2P system (the most used in the recent research works) is the protocol Chord.

3. A Chord-based Architecture for Semantic Web Services Discovery. This paper is interested to decentralized P2P networks. As we have already mentioned, in these systems, there are two methods to send queries: DHT and flooding-based algorithms. In comparison with unstructured P2P topologies, structured P2P systems have several advantages like: scalability, availability, and management of the fault tolerance.

On the other hand, unstructured P2P networks (such as Gnutella V0.4) require no centralized directories and no precise control over network topology or data placement. Though, the flooding-based query algorithm used in these systems does not scale; a query generates a large amount of traffic hence large systems become quickly overwhelmed by the query-induced load [6].

Our proposed architecture is based on Chord, which is chosen because it is simple and still achieves very good resilience and proximity performance. In the following subsection, we will briefly present the main characteristics of this protocol.

3.1. Chord Overview. Chord implements a DHT using an m-bit identifier ring, $[0, 2^m - 1]$, for routing and object location. In an $N-nodes$ system, a query can be routed via $O(\log N)$ hops (see Fig.3.1). The destination point of a query is determined by hashing its key. At each hop during routing, the query is forwarded to a peer whose identifier most immediately precedes the destination point in its finger table (contains the list of peers that a peer uses to send messages to) [4].

Chord is a dynamic system where the peers can enter or leave the system at anytime at will. Then, each node maintains information only about $O(\log N)$ other nodes, and resolves all lookups via $O(\log N)$ messages to other nodes. Chord maintains its routing information as nodes join and leave the system; with high probability each such event results in no more than $O(\log_2 N)$ messages [32].

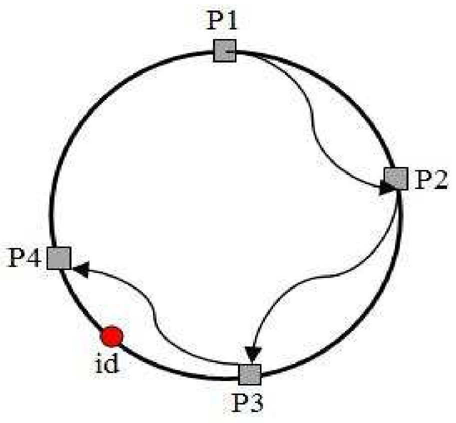

Fig. 3.1. *Example of Chord system*

Chord protocol has several advantages and important characteristics like: Scalability, availability, and management of the fault tolerance. However, in the context of semantic Web services discovery, the requester can search for a capacity, a goal or a property of a resource. In this case, one of the disadvantages of this protocol is the convergence of the request, especially, when the number of nodes is certainly large. For this reason, we have integrated the time aspect in our discovery algorithm, in which we have marked the requests by a TTL (Time To Live).

3.2. An overview of the proposed architecture . In this paper, we describe a P2P based architecture that combines the hybrid and the structured typology of P2P network, in which we employ Chord. This architecture supports the centralized and decentralized discovery methods of Web service. It uses a central common ontology, which used by the different providers to annotate semantically their Web services descriptions.

The P2P network is used to organize the service providers into a hybrid/Chord overlay and allows them to advertise and lookup services in a centralized/decentralized dynamic manner. Figure 3.2 shows the main components of this architecture.

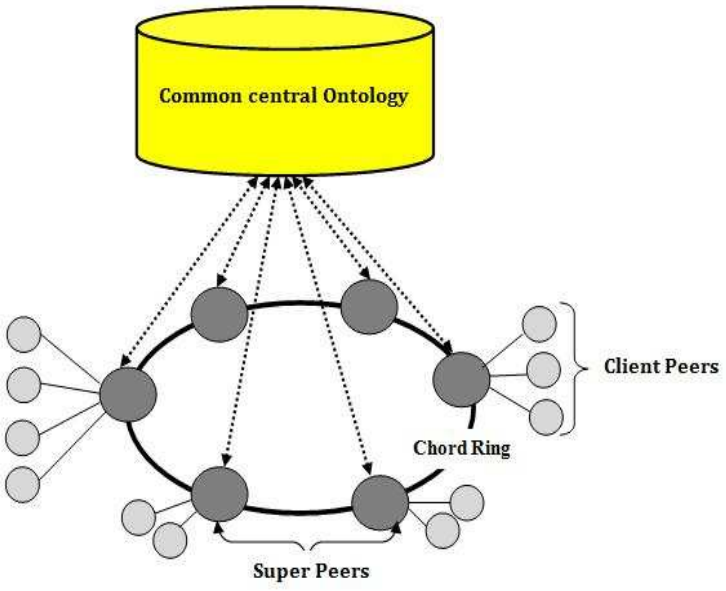

Fig. 3.2. *The main Architecture*

The main P2P network is presented by the whole of super peers organized as a Chord ring. The main system is decomposed on clusters presented by the Client/Server relationship between each super peer and their connected client peers. A cluster is a mini network, in which every super peer provides to the client peers an ensemble of services in term of discovery and publication of Web services. In the following subsections, we detailed the three main components of this architecture: Central common ontology, clusters and the distributed discovery over the Chord overlay.

3.3. Central Common Ontology. In a large distributed system, it is evident that the provided resources are strongly heterogeneous. In the same context, if the resources are Web services, this problem is vital. Even though the technical interoperability of Web services was assured essentially by the three main standards XML, SOAP [30] and WSDL [31], the semantic interoperability is already a real problem that influences the discovery of Web services, especially, in the large distributed networks.

Consequently, to guarantee the semantic interoperability between the different provider peers of the network, it's important to provide a mechanism that assures the dynamic discovery of Web services those distributed among the different participant peers.

In this framework, we propose to use a common central ontology that provides a large variety of concepts of great selection of Web services domains. In the first case, each provider peer can reuse one or more several descriptions that provided by this central base. An example of such ontology is that created by [33] that

provide about 240 OWL-S services descriptions based on a main ontology named *"concepts.owl"*. In this case, if the provider want to supply a Web service that corresponding to one of the provided OWL-S descriptions, he can reuse directly this one. Thus, in our proposed architecture each provider peer can use the common ontology to extend semantically the descriptions of their Web services.

Furthermore, in our work, we suppose that every Web service is defined by the tipple (input, output, goal). The following example presents a Web service that accepts in its input book information, in the output the price of this book and the goal specifies the used Money (EUR in this example):

Listing 1

example of a semantic search request

```
Input : Title||Author||Edition||year−of−edition Output : Price
Goal:∗input:=#book:Title||Author||Edition||year−of−edition
∗output:=#book:Price//Money:EUR
```

3.4. Client Peers/Super Peers. A client peer joins the system by connecting to one and only one super peer. Then, it leaves the network by disconnecting from it's super node. Each super peer processes the different requests forwarded by their client peers. However, this description does not mean that client peers are only customers. They can provide some Web services; which are indexed and published within the Chord ring by super peers.

Furthermore, to the high availability and computing capacity, super peers are characterized by the number and the quality of the provided services and resources. Thus, after the first peer joins the system as a super peer, it designs one of its client peers with good resources as a new super peer. Furthermore, the super peer can observes the participation of each client peer, if anyone was a good participant; it will be supported by the super peer to be a new one.

In addition, the clustering organization of the peers (client peers/Super peers) offers many benefits:

- Whenever client peers just send queries and reply some service requests, super peers (peers provide a lots of services) benefit directly from the advantages of the Chord overlay network by doing all the message routing and indexing which minimizes the discovery time of the requested Web services when the requestor is a super peer.
- This gain for super peers, in term of resources and the search time by exploiting the distributed system (Chord ring), encourages the other peers (client peers) to provide more services and participate actively in the system. Thus, the number of the selfish peers in the system will be reduced, whenever, this one of the most important problems of the P2P systems (selfish peers are just consumers, they don't provide any resource for the other peers).
- For the super peers, the system is more flexible and scalable because of the simplicity of Chord in term of routing messages and locating resources. Whereas, the client peers can join (leave) the system by simply connecting to (disconnecting from) their super peers.

3.5. Distributed discovery over the Chord ring. To ensure the cooperation between the different super peers, we implement a distributed discovery algorithm over the Chord ring. As a first step of this work, we favorite to propose the Algorithm 13, that researches only for basic Web services. However, we have optimized the original implementation of Chord protocol through the insertion of a TTL (Time To Live) into sent requests. Because the discovery of Web services is made semantically, the search time, in each super peer, will be significant. For this reason, the marking of the request by a TTL becomes the discovery operation more realistic and reflects the exigencies of users in term of research time.

In this context, it's important to mention that the original implementation of Chord didn't design for "semantic discovery". It is more developed for "exact search" where the requester knows previously the precise identifier of the service. So, the "semantic discovery" is the most motivation to use the TTL in our solution. In this approach, the requester doesn't know the exact identifier of the requested service. The semantic discovery is based on keywords extracting from the common ontology. Consequently, the integration of the TTL becomes a necessity because the search time is important in such approach. Thus, in contrast with the original implementation of Chord, the requestor searches "semantically" for the adequate service until he finds it or the discovery operation stops when the TTL is finished.

The following Algorithm 13 is implemented in each super peer. It is executed when a super peer receives a request from a client peer or another super peer in the Chord ring.

From the Algorithm 13, we can deduce that the execution stops if we obtained the researched Web service or if the TTL is finished. For this reason, it's evident that the insertion of the TTL influences directly the obtained result. To study this effect, in the following section, we present some experimental results after the simulation of the Algorithm 13.

4. Simulation and Experimental Results. After the insertion of the Algorithm 13 in the Chord implementation for *PeerSim* simulator [19], we obtained various experimental results presented in the next sections.

Algorithm 13 Distributed Discovery Algorithm

```
Require: $Webservice(input, output, goal, TTL)$
Ensure: $service\_contract$
  $Receive[Search\_Service(input, output, goal, TTL)]$;
  if (there is a Web service published in the cluster) then
      $Get(contract\_service)$;
  else
      $Decrease(TTL)$;
      $Send[Search\_Service(input, output, goal, TTL)]$;
  end if
```

In the following, we present different case studies by using some characteristic of the network: size (number of super peers), number of nodes that join/leave the network, stabilizations..etc. The observations of the different input/output results are summarized in Table 4.1.

Table 4.1
Observations of the hops number in the Chord ring.

Number of super peers	Super peers Join/leave the Network	Number of hops	Min/Max of hops	Hops Average
500	200/250	2049	0/13	3
1000	250/500	3094	0/13	3
2000	500/750	3169	0/15	4
5000	750/1000	4187	0/15	6
10 000	1000/1250	5322	0/14	6
50 000	2000/1000	6900	0/15	8

4.1. Number of Failures. The main difference between the original implementation of Chord protocol and our implementation is that we mark the discovery request by a TTL. For these reason, we made a simulation to detect the number of failures where the request doesn't achieve its goal because of the end of TTL. Furthermore, the number of super peers that can be leave the network or when they can break down. To calculate the initial TTL for each experience, we have added a method named `durationTime` to the classes `ChordProtocol` and `CreateNw`. The java code of this method is presented in the following listing:

Listing 2
java code of the durationTime method

```
public long durationTime(long duration) {
 startTime = System.nanoTime();
..............................
 endTime = System.nanoTime();
duration= endTime−startTime;
return duration;
}
```

The main objective of this method is to evaluate the maximum time to cover the entire network. It's calculated as follow:

MaxTTL = $\sum_{i=1}^{n}(durationTime(super-peer(i))) + \sum_{j=1}^{m} Time\_create\_new\_peer(j)$. Where:

- n : number of super peers
- m : number of new super peers that joined the network

The "MaxTTL" (or TTL) is evaluated before the integration of the Algorithm 13 in the original implementation of Chord. Table 4.2 summarizes the evaluation of the TTL in correspondence with the network size.

In addition, before the integration of the Algorithm 13 in the source code of Chord protocol, we have obtained the observations presented in Table 4.3.

Table 4.2
Evaluation of the TTL.

Network size	MaxTTL (nano second)	TTL (second)
500	5 521 468 554	$\simeq 6$
1000	10 756 845 215	$\simeq 11$
2000	21 032 225 142	$\simeq 21$
5000	45 689 998 569	$\simeq 46$
10 000	112 231 543 268	$\simeq 112$
50 000	564 523 669 884	$\simeq 565$

Table 4.3
Number of failures without discovery algorithm.

Number of super peers	Super peers Join/leave the Network	observations	Number of failures	Stabilization
500	200/250	828	2	126
1000	250/500	1200	5	220
2000	500/750	3169	7	608
5000	750/1000	4187	8	828
10 000	1000/1250	5322	10	1750
50 000	2000/1000	6900	13	5458

After the integration of the discovery Algorithm in the Chord implementation, we have obtained the results presented in Table 4.4.

Table 4.4
Number of failures after the integration of Algorithm13

Number of super peers	Super peers Join/leave the Network	observations	Number of failures	Stabilization
500	200/250	828	9	544
1000	250/500	1200	13	627
2000	500/750	3169	15	700
5000	750/1000	4187	18	965
10 000	1000/1250	5322	25	2078
50 000	2000/1000	6900	32	7865

According to the experimental results summarized in the Table 4.4, we can conclude that the TTL increases the number of failures, because the request cannot continue its path in the Chord ring when the TTL is finished. However, in Table 4.3, the request stops, only, if a super peer (receiver or intermediary) is break down or leaves the network. Figure 4.1 schematizes a comparison between the number of failures before and after the integration of Algorithm 13 (where we mark the request by a TTL).

From the results presented in Figure 4.1, we can demonstrate that, after the implementation of the discovery algorithm of Web services (Algorithm 13), the TTL has increase the number of failures. In comparison with the original implementation of Chord protocol, the integration of the TTL becomes the Web services discovery operation more realistic. Furthermore, we have observed that the most practical number of super peers is 2000, where the percentage of the success request is approximately 50%. In this context, it's important to clarify that as a first step, we studied only the influence of the TTL on the success rate of the discovery operation. For this reason, the described simulation and experimental results offer better values in the non-TTL set of experiments. An additional work is necessary to decrease the number of failures in the TTL results.

4.2. Stabilizations. Another important characteristic that we have experimented is the stabilization of the network after the implementation Algorithm 13. It is presented by the number of operations realized per a super peer to update the DHT table. Figure 4.2 demonstrates the number of stabilizations before and after the addition of the discovery algorithm to the Chord implementation.

In Figure 4.2, we observe that the call of the stabilization method, by a super peer, increases when the size of the network augments. Once the number of super peers exceeds 10000, the network becomes non-stable.

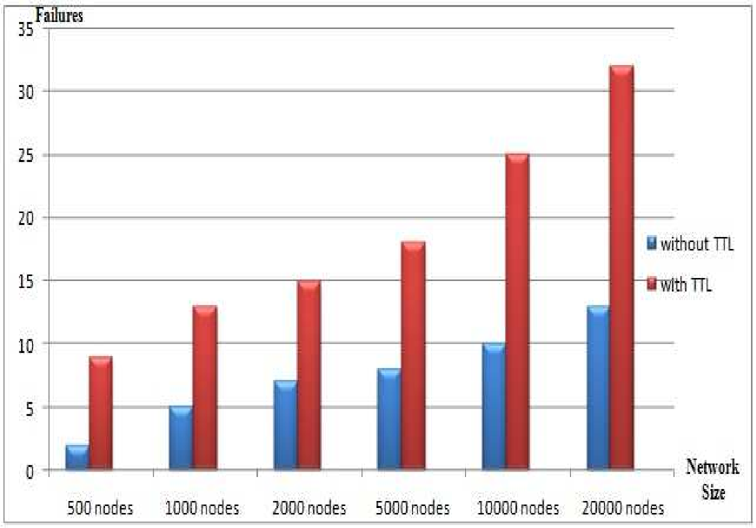

Fig. 4.1. *Number of failures with and without TTL*

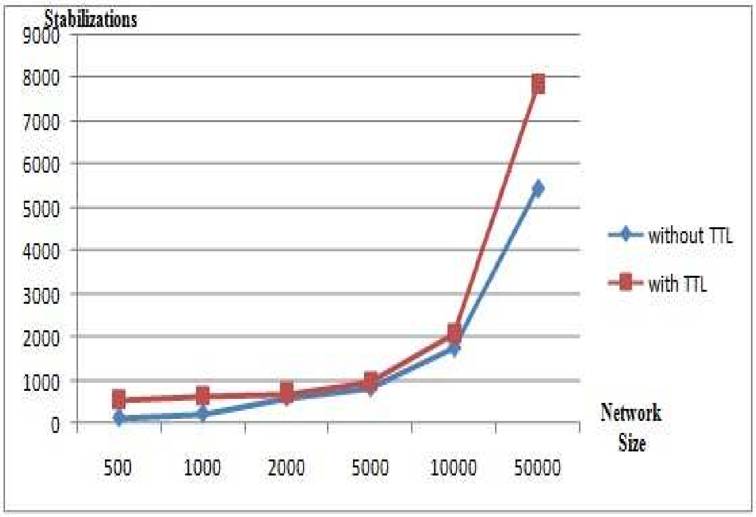

Fig. 4.2. *Number of stabilizations*

For this reason, the super peers call the stabilization method frequently. In addition, we note that, when the number of super peers is between 2000 and 5000, the number of stabilizations is very close in the tow cases. Most precisely, the number of calls of stabilization method is nearly the same when the number of nodes is 2000. This result confirms the conclusion that we had already deduced in the previous subsection.

4.3. Traffic generation. The last propriety that we have analyzed, in this experimental study, is the traffic generation. This last is defined by the number of sent requests generated by the nodes that research a particular Web service. Figure 4.3 presents the number of observations with and without integration of Algorithm 13.

In this case, we detect that the difference between the two results is almost negligible. This conclusion is resulted because the discovery algorithm traits the live time of a request but not the number of sent requests. This is generated arbitrary by the class *"trafficGnerated"* implemented originally by the Chord protocol. However, from a more precise observation, we can view that the exactitude between the two results is inspected

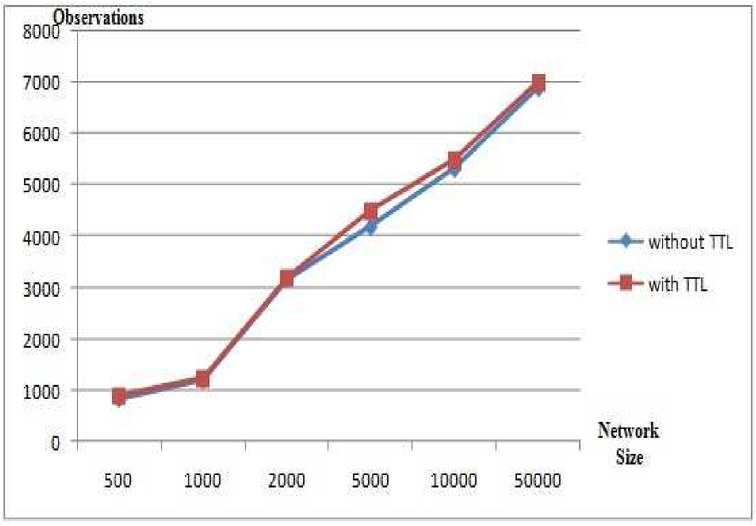

Fig. 4.3. *Number of observations in function of the size of the network*

when the number of super peers is less than 2000.

5. Related works. Firstly, it's important to clarify that this paper doesn't focus on a new family of P2P routing schemes or an extension version of Chord, like the works presented in [5, 8, 16]. Our work presents a new distributed approach for semantic Web services discovery. In this context, several research works improved the decentralized discovering methods of Web services in the P2P networks. For example, a number of centralized and P2P Web service discovery methods have been proposed in the area of the Web services composition, Web services based business process management and Web services discovery in P2P Clouds [13]. In comparison with our previous works [10, 11], where we have employed the unstructured P2P paradigm, in this work, we have used Chord, which is a structured P2P protocol. Another difference is that in our previous work [10], we have proposed a number of algorithms that realized the discovery and the composition of Web services in the same time. The proposed solution has posed some problems in term of convergence and the complexity of the algorithms. In contrast, this article presents an algorithm, which intended the discovery operation only. However, the proposed algorithm has proved its scalability and doesn't influence the stabilization and the traffic generation of Chord protocol.

Another related work is that of Z. Zhengdong et al [39], that designs a localized mechanism of semantic Web services in CAN-based P2P networks to ensure that each service is registered in a specific node of the public Web. The registration services of public Web nodes are divided by the area and shared by all nodes. This work processes the CAN-based P2P networks; in contrast, in our work, even we presented a simple discovery algorithm; the proposed solution is generic and can be extended to a multi-layer Chord model like in [5].

In the same area, F. Mandreoli et al. [18] present the architecture of FLO2WER, which is a framework that supports large scale interoperation of semantic Web services in a dynamic and heterogeneous P2P context. They have adopted a hybrid approach that exploits the advantages of centralized registries for service discovery and composition, as well as the dynamism and the scalability of non-structured P2P networks. The main idea of FLO2WER framework is that, while decentralizing the knowledge of what specific services are available in the system, they keep centralized knowledge of what objectives may be satisfied within the network namely Goals. Each Goal specifies therefore a sub network of specific services, and it is stored in an appropriate repository, called Goal Repository. However, it is not described and detailed how the goals have been discovered. Moreover, the use of a central repository of goals is similar to central discovery methods based on Web services functionalities. In contrast, our discovery method combines decentralized and centralized solution in the same time.

In another work, T. Essafi et al. [34] present a scalable P2P approach to service discovery using ontology. This work incorporates input/output matching algorithm proposed in paper [23] and extends the solution described in paper [22] by adding an encoding that locates servers in a P2P network to simplify rerouting of query messages. Idem to the precedent work [24], this project adopts only network centralization and hierarchy. However, the main objective of our work is to ensure dynamic Web service discovery. Also, in this work, they still relay on the old DAML-S, which proved to be less flexible than the new OWL-S.

One of the adjacent researches works is that of O. D. Sahin et al [27] where they proposed a similar architecture of us. Though, this paper presents the different semantic techniques for Web services discovery in P2P Chord network. It was very poor in term of experimental results and the improvement of the original Chord implementation. In contrast, our work focuses on Chord proprieties before and after the implementation of the algorithm (more precisely, the influence of the TTL).

6. Conclusion. In this paper, we proposed a P2P based discovery approach for semantic Web services in a large distributed environment. In this approach we employed Chord, which is a structured P2P protocol that improved a high degree of scalability. At this purpose, we adopted a hybrid architecture that exploits the advantages of the decentralized design of the Chord protocol and the clustering organization of the different peers.

This architecture contains two main components: a common central ontology and a P2P network. To guarantee the semantic interoperability in such large environment, which characterized by high degree of heterogeneity, we supposed that the participant peers offer semantic descriptions of their proposed Web service by using the same ontology. This proposition ensures the interoperability and facilities the discovery operation in the P2P network.

The main contribution of our work is the proposition of distributed algorithm for semantic Web services discovery. To achieve this goal, we have updated the original Chord implementation (for Peersim simulator) by injecting the proposed algorithm. The main change that we have realized is the addition of the TTL to the sent requests.

After the simulation, we have obtained experimental results about the effect of the TTL in term of number of failures, stabilization and the traffic generation in the network. According to the achieved results, we have concluded that the TTL increases the number of failures, because the request cannot continue its path in the Chord ring when the TTL is finished. However, for the network stabilization and the traffic generation, the implementation of the discovery algorithm doesn't strongly influence the network. For the network stabilization, we have observed that the call of the stabilization method increases when the number of super peers exceeds 10000 nodes. Though, for the last propriety that we have analyzed (traffic generation), we have detected that the difference between TTL-Chord and original Chord is almost negligible.

However, this task presents the first step of this work. In the future, we plan to optimize the proposed algorithms by the following characteristics:

- Evaluation of the quality of the discovered services (QoS),
- Improvement of the proposed algorithm to discover composite Web services,
- Proposition of a probabilistic algorithm that ameliorates the proposed one.

As a short objective, we are, now, trying to apply the proposed solution for semantic Web services discovery in a P2P Cloud environment.

REFERENCES

[1] Tien Tuan AnhDinh, GeorgiosTheodoropoulos, and Rob Minson. Evaluating large scale distributed simulation of p2p networks. In *12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, pages 51–58, Vancouver, Canada, October 2008.

[2] The annotated Gnutella protocol specification v0.4. http://rfcgnutella.sourceforge.net/developer/stable/, 2001.

[3] Bawa S. Batra, S. Review of machine learning approaches to semantic web service discovery. *Advances in Information Technology*, 1(3):146–151, August 2010.

[4] Fatih Emekçi, Ozgur D. Sahin, Divyakant Agrawal, and Amr El Abbadi. A peer-to-peer framework for web service discovery with ranking. In *ICWS*, pages 192–199, 2004.

[5] Yung-FaHuang Eric Jui-LinLu and Shu-ChiuLu. Ml-chord: A multi-layered p2p resource sharing model. *Journal of Network and Computer Applications*, 32:578–588, 2009.

[6] Ronaldo A. Ferreira, Murali Krishna Ramanathan, Asad Awan, Ananth Grama, and Suresh Jagannathan. Search with probabilistic guarantees in unstructured peer-to-peer networks. In *Peer-to-Peer Computing*, pages 165–172, 2005.

[7] OWL-S: Semantic Markup for Web Services. http://www.w3.org/submission/owl-s/, 2004.

[8] Mikael Hammar Alberto Negro Gennaro Cordasco, Luisa Gargano and Vittorio Scarano. Fchord:improved uniform routing on chord. In *11th Colloquium on Structural Information and Communication Complexity (SIROCCO 2004),Lecture Notes in Computer Science, Vol. 3104*, 2004.
[9] Mohamed Gharzouli and Mahmoud Boufaïda. A generic p2p collaborative strategy for discovering and composing semantic web services. In *ICIW*, pages 449–454, 2009.
[10] Mohamed Gharzouli and Mahmoud Boufaïda. A distributed p2p-based architecture for semantic web services discovery and composition. In *NOTERE*, pages 315–320, 2010.
[11] Mohamed Gharzouli and Mahmoud Boufaïda. Pm4sws: A p2p model for semantic web services discovery and composition. *Journal of Advances in Information Technology, Special Issue on Advances in P2P Technology*, 2(1), 2011.
[12] Jianqiang Hu, Changguo Guo, Huaimin Wang, and Peng Zou. Web Services Peer-to-Peer Discovery Service for Automated Web Service Composition. In *ICCNMC*, pages 509–518, 2005.
[13] Zhongzhi Shi Jing Zhou, Nor Aniza Abdullah. A hybrid p2p approach to service discovery in the cloud. *I.J. Information Technology and Computer Science*, 3(1):1–9, 2011.
[14] Peep Küngas and Mihhail Matskin. Semantic web service composition through a p2p-based multi-agent environment. In *AP2PC*, pages 106–119, 2005.
[15] Dongsheng Li, Xicheng Lu, Yijie Wang, and Nong Xiao. Efficient search in gnutella-like ”small-world” peer-to-peer systems. In *GCC (1)*, pages 324–331, 2003.
[16] Shiming Zheng Wenjie Sun Li Chen, Zilin Song and Zhanfeng Wang. A model of web service discovery based on balancechord. *Journal of Computational Information Systems*, 7(7):2241–2247, 2011.
[17] Jie Liu and Hai Zhuge. A semantic-link-based infrastructure for web service discovery in p2p networks. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, WWW '05, pages 940–941, New York, NY, USA, 2005. ACM.
[18] Federica Mandreoli, Antonio Massimiliano Perdichizzi, and Wilma Penzo. A p2p-based architecture for semantic web service automatic composition. In *DEXA Workshops*, pages 429–433, 2007.
[19] Alberto Montresor and Mark Jelasity. Peersim: A scalable p2p simulator. In *Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09)*, 2009.
[20] Hajar Mousannif, Ismail Khalil, and Gabriele Kotsis. The cloud is not 'there', we are the cloud! *IJWGS*, 9(1):1–17, 2013.
[21] WSMO: Web Service Modeling Ontology. http://www.w3.org/submission/wsmo/, 2004.
[22] Kawamura T. Payne T.R. Sycara K. Paolucci, M. Semantic matching of web services capabilities. In *First International Semantic Web Conference (ISWC)*, pages 333–347, Sardinia, Italy, 2002.
[23] Sycara K. Nishimura T. Srinivasan N. Paolucci, M. Using daml-s for p2p discovery. In *International Conference on Web Services (ICWS)*, pages 203–207, Las Vegas, Nevada, USA, 2003.
[24] Mike P. Papazoglou, Bernd J. Krämer, and Jian Yang. Leveraging web-services and peer-to-peer networks. In *CAiSE*, pages 485–501, 2003.
[25] Gnutella protocol Development v0.6. http://rfc-gnutella.sourceforge.net/src/rfc-06-draft.html, 2002.
[26] K. Rageb. An Autonomic ¡K, D¿-Interleaving Registry Overlay Network for Efficient Ubiquities Web Services Discovery Service. *Journal Information Processing Systems (JIPS)*, 4(2), 2008.
[27] Ozgur D. Sahin, Cagdas Evren Gerede, Divyakant Agrawal, Amr El Abbadi, Oscar H. Ibarra, and Jianwen Su. SPiDeR: P2P-Based Web Service Discovery. In *ICSOC*, pages 157–169, 2005.
[28] Winterhalter C. Schmidt, A. User context aware delivery of e-learning material: Approach and architecture. *Universal Computer Science (JUCS)*, 10, 2004.
[29] METEOR-S: Semantic Web services and Process. http://lsdis.cs.uga.edu/projects/meteor-s/, 2004.
[30] SOAP specification. http://www.w3.org/tr/soap/, 2007.
[31] WSDL specification. http://www.w3.org/tr/wsdl20, 2007.
[32] Morris R. Karger D. Kaashoek M.F Balakrishnan H. Stoica, I. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOM*, pages 149–160, California, USA, 2001.
[33] Semantic Web Central: Project sws tc. http://projects.semwebcentral.org/projects/sws-tc/, 2006.
[34] N. DORTA T. ESSAFI and D. SERET. A scalable peer-to-peer approach to service discovery using ontology. In *Proc of 9th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, 2005.
[35] UDDI:. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec, 2005.
[36] Xia Wang and Wolfgang A. Halang. *Discovery and Selection of Semantic Web Services*, volume 453 2013. Springer-Verlag Berlin Heidelberg, studies in computational intelligence edition, 2013.
[37] W3C Working group Web services Architecture. http://www.w3.org/tr/ws-arch/, 2004.
[38] WS-Gloss:. http://www.w3.org/tr/ws-gloss/, 2004.
[39] Yahong H. Ronggui L. Weiguo W. Zengzhi L. Zhengdong, Z. A p2p-based semantic web services composition architecture. In *IEEE International Conference on e-Business Engineering*, pages 403–408, Macau, China, 2009.

Edited by: Dana Petcu
Received: Sep 1, 2013
Accepted: Mar 4, 2014