PM4SWS: A P2P Model for Semantic Web Services Discovery and Composition

Mohamed Gharzouli
LIRE Laboratory, Mentouri University of Constantine
Constantine, 25000, Algeria
gharzouli@gmail.com

Mahmoud Boufaida
LIRE Laboratory, Mentouri University of Constantine
Constantine, 25000, Algeria
mboufaida@umc.edu.dz

***Abstract*— Since the beginning of the service oriented paradigm, the Web service discovery presents an important problem. Web Services should act autonomously with as minimal human intervention as possible, they should be able to discover other services which have particular capabilities and realize precise tasks. To improve the automatic discovery of Web services, other technologies are used to provide new flexible solutions for web services discovery and composition. In this context, the Web semantic and the Peer-to-Peer (P2P) computing present the most adapted technologies to the service-oriented approach. We present a distributed solution based on epidemic algorithms to discover semantic Web services in unstructured P2P networks. In this solution, we use a distributed table to preserve the description and the composition way of the P2P composed Web services. We improve this choice by adding other algorithms that ensure the data coherency of this table. Also, we present a distributed architecture for which implements this solution. A motivate example is presented in this paper to improve the proposed model.**

***Index Terms*— Semantic Web services; P2P computing; Web services composition; Web services discovery; Distributed applications;**

I. INTRODUCTION

Since its appearance, the internet has evolved toward many domains from business environment to scientific applications. Recent years have seen an evolution of data management systems from centralized systems to decentralized systems, because of the increase in the demand for resource sharing across different sites connected through networks. Decentralized systems enable large-scale distributed applications providing high scalability. Feasibility of these systems relies basically on P2P techniques.

P2P computing is considered as the next evolutionary step in computing and a new generation of the networking. This new direction in distributed computing focuses on networking and resource sharing with better reliability and scalability [7], [8]. Recently, P2P systems have opened many challenges in many fields: semantics, collaborative work and discovery of pertinent resources. Among these domains, Web services based on P2P computing require special attention from collaboration and interoperability in a distributed computing environment [8]. Web Services technology is considered as a revolution for the web in which a network of heterogeneous software components interoperate and exchange dynamic information [4], [5], [6]. In the last few years, other technologies have been used to improve the automatic discovery of Web services. An important improvement has been made for P2P computing and Web semantic technologies. P2P systems provide a scalable alternative to centralized systems 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. On the other hand, the Web semantic technologies can be used to facilitate the dynamic discovery of Web services. A semantic description of Web services is more understandable by the machine.

Therefore, in order to exploit the advantages of P2P networks and the Web semantic, we combine these technologies to propose a distributed model to manage, discover and compose Web services.

The contribution of this paper contains three main parts. The first one describes an unstructured P2P based strategy of Web services discovery and composition [3]. The principal idea of this strategy is to present research epidemic algorithms based on logic planning. These algorithms enable us to find basic services distributed among all the peers to compose a personalized service. Moreover, in this part, we use a table of composition to preserve the trace of the composition, for a possible future re-uses. This table is considered as a cache memory, which preserves the P2P composition ways. In addition, in this paper, we present some algorithms to ensure the coherence of data of this distributes table. The second part of this paper consists of describing a distributed framework, which implements the suggested algorithms presented in the last part [18]. We improve

Manuscript received September 14, 2010; revised November 5, 2010; accepted November 25, 2010

doi:10.4304/jait.2.1.15-26

this implementation through a motivation example. In the third part, we discuss the advantages and the disadvantages of this solution and we present some suggestions to evolve it.

The rest of this paper is structured as follows: in section 2, we briefly discuss the Web services discovery challenges. Section 3 presents a strategy to discover and compose distributed Web services in the unstructured P2P networks. Section 4 describes a framework that implements the precedent strategy. In section 5, we discuss future work. Section 6 presents related works and section 7 concludes the paper.

II. WEB SERVICES DISCOVERY CHALLENGES: OVERVIEW AND MOTIVATION

The first generation of works done on Web services architectures propose different solutions based on centralized discovery methods (such as UDDI), where Web services are described by service interface functions and they publish their capabilities and functionalities with a registry (Figure 1) [31].

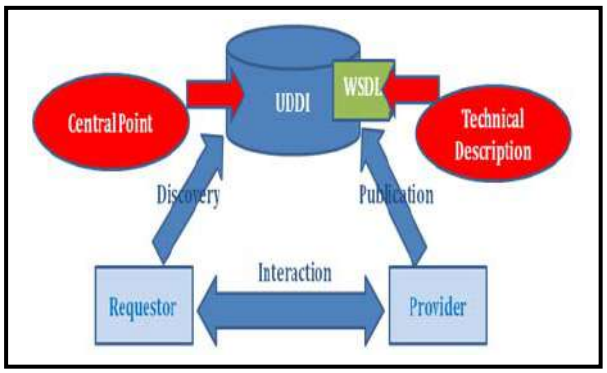

Figure 1. Web services Architecture

In this architecture, the discovery of Web services remains two main hard problems. The first one is the centralized point of publication and discovery which represent a repository like UDDI or a research engine like service finder [27]. These methods are not adapted to the dynamic interactions, they restrict the scalability of flexible and dynamic environment [1] [2], induces performance bottleneck and may result in single points of failure [13]. Moreover, the centralized control of published services suffers from problems such as high operational and maintenance cost. Furthermore, the universal UDDI repository suffers from the miss of moderation: many of published Web services are not available or they are not implemented by the providers.

The second problem of Web services discovery is the description realized generally by WSDL. This last provides only a technique description. However, the automatic Web services discovery requires a more intelligible description that more understandable by the machine.

However, 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 [14]. 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 [15].

To solve these problems, 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. Although, a considerable number of them discuss the dynamic discovery and the distributed composition of semantic Web services in the structured P2P networks like Chord [12], Pastry [16], and CAN [17]. This type of networks is proposed to solve a number of problems appeared in the first generation of the P2P architectures (centered and pure P2P systems). However, DHT-based protocols (used by a variety of these networks) are difficult to maintain because of their static topologies (for example, a ring for Chord). Moreover, the DHT (Data Hash Table) cannot allow complex research.

On the other hand, little of research works discuss the use of the unstructured P2P networks. These systems (such as Gnutella V0.4 [28]) 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 [11].

However, the decision to employ the unstructured P2P networks in this work is justified by several reasons. Generally, in the context of semantic Web services discovery and composition, the requester (service customer) searches a capacity, a goal or a property of a resource (service). So, when it sends the request, the invoker requires no placement or identifier of the resource. This characteristic does not exist in the structured P2P systems, to find a resource in these networks, it is necessary that the requester must be known beforehand the identifier of the resource. So, the unstructured P2P systems are more adapted to discover distributed Web services. Furthermore, the unstructured P2P systems are used by very large communities of net surfer.

In addition, our approach explores various solutions to unstructured P2P discovery algorithms. We propose a distributed replication strategy that resolves the problem of flooding-based methods, while reducing the network traffic, and accelerates the discovery of semantic Web services. To achieve this objective, we propose use a replication table (the composition table) that preserves the composition way of precedent composed Web services. This table has many advantages that will be exposed in the following paragraph.

III. AN UNSTRUCTURED P2P BASED STRATEGY TO DISCOVER AND COMPOSE SWS

In our work already presented in [03], we proposed a whole of epidemic algorithms that supports the automatic discovery and composition of the semantic Web services distributed among an unstructured P2P network. The main objective of this solution is to compose a Web service which answers a particular request. This service can be composed from a set of Web services distributed through several peers of the network.

In order to discover the appropriate Web services, we implemented epidemic algorithms based on Input/output matching and oriented by the achieved goal. Furthermore, all participant peers store in a distributed table all traces of preceding composed Web services. This historic can be used to give a rapid response from future similar requests. In the following paragraphs we expose the composition table and the epidemic discovery algorithms.

A. Structure of the composition table

Each peer creates a table to store all compositions in which it has already participated to realize them. The different attributes of the composition table are defined as follows:

Initiator Peer: the peer that begins the composition.
compoID: the composition identifier (each composition is defined in the network by its initiator peer and its identifier).
Init-input: initial input of the composite Web service.
Init-output: initial output of the composite Web service.
Goal: the goal of the composite Web service
Executed services: Web services executed locally to compose the achieved Web service.
Reserve services: local web services which can be executed in the same composition.
Precedent peers: peers, which execute the precedent Web services for the composition.
Next peers: peers, which execute the next Web services for the composition.
Reserve peers: peers, which can replace the next peer.
State of the composite Web services: the composite Web service is active, if all the participants' peers are joining the network.

These concepts are presented in the following example (Figure 2):

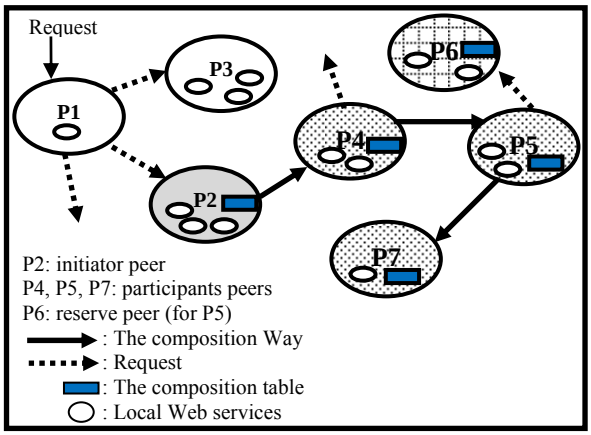

Figure 2. Example of an unstructured P2P composition

This example presents a P2P composition of a whole of web services distributed among five peers in the network: P2, P4, P5 and P7. P6 is a reserve peer for P5 in this composition i.e. it can replace P7. For P5, P2 and P4 are precedent peers and P7 is a next peer.

B. Epidemic discovery algorithms based on input/output matching

Initially, let us give the different definitions of the basic concepts used in the following algorithms.

Definition 1: A Web service is 3-uplet defined as: WS= (WS-input, WS-output, WS-goal).
Definition 2: A Goal is the conceptualization of a domain of services whose ultimate aims are identical or similar [1]. A goal is specified in the semantic description of the Web service.
Definition 3: A composition is a process constituted from a whole of basic services. In our context, there are two types of compositions: local composition and P2P-composition.
Definition 4: A Local Composite Web Service (LocCWS) is built from a whole of Web services belonging to the same peer.
Definition 5: A P2P-composition is a composition where the P2P composite Web service (P2P-CWS) is build from a whole of Web services belonging to different peers of the network. A P2P-composition (P2P-Comp) is tuplet (Initiator-Peer, CompID, Init-Input, Init-Output, Goal).
Definition 6: Participant Peers: all peers that participated (collaborated) in a P2P-composition (the initiator peer is a participant peer).

1) Main Algorithm: Each peer in the network executes the following main algorithm when receiving a request:

Begin
1. Receive the request (init-Input, init- Output, goal).
2. Search a local Web service where [(WS-input=Init-Input) and (WS-output=Iinit-Output) and (WS-goal=Goal)]
3. **If** (there is a local WS) **then** send the response; go to END;
4. **If** (there is not a local WS) **then**
   Compose a local Web service where [(LocCWS-input=Init-Input) and (LocCWS-Output=Init-Output) and (LocCWS-goal=Goal)];
5. **If** (there is a LocCWS) **then** send the response; go to END;
6. **If** (there is not a LocCWS) **then**
   Search-in-Composition-table (Init-Input, Init-Out-put, Goal);
7. If (there is not P2P-CWS in the composition table) **then** Launch-a-New-P2P-discovery( );
8. **End.**

Initially, each peer executes the main algorithm from step 1 to step 5 where it tries to response to the request locally (local basic Web service or a local composition).

In the step 6 of the algorithm, the peer searches in its composition table about a precedent composition realized in the network that responses the request. The procedure Search-in-composition-table is defined as follow:

Search-in-Composition-table ()
Begin
Select [Initiator-Peer, CompID] **from** the composition table **where** [(P2P-CWS-Input=Init-input) and (P2P-CWS-output=Init-Output) and (P2P-CWS-goal=goal) and (state-of -P2P-CWS=Active)]; /*create a selection-list*/
1. **If** (the selection-list ≠ empty) **then**
While (selection-list ≠ empty) and (P2P-composition is not possible) **do**
a) Select an Initiator-Peer from the selection-list;
b) Send a message: **search** (CompID, Init-Input, init-Output, Goal) from the initiator peer;
c) **If** (success P2P-discovery) **then** send the response; go to END; /* the end of the main program*/
End While; END.

If there is not a possibility to answer the request locally or by using the composition table, the peer starts a new P2P discovery (step 7 of the main algorithm). For this raison, we proposed two epidemic discovery algorithms based on logic planning methods. We present in what follows two types of algorithms: Before-chaining and Back-chaining.

2) Before-chaining algorithm: In this algorithm, each peer searches an input of a service, which can progress the composition process. To start a P2P discovery based on the Before-chaining algorithm, the peer prepares the list of the services which have an input equal to the init-input (List-WS-Init-Input). These local services can be basic or partially composed locally. The Before-chainning algorithm is defined as follow:

Begin
1. Create List-WS-init-Input (the list of Web services with the same input).
2. **While** (List-WS-Init-Input not empty) **do**
a) Select the Head of List-WS-Init-Input;
b) Init-Input :=Output_of_Heading-List-WS-Init-Input; calculate the TTL;
c) Send message search (initiator- Peer, Iinit-Input, Init-Output, goal, TTL);
d) **If** (time ≤ TTL and response success research) **then**
Stop research; send a success response; go to End;
Else
If (time ≤ TTL and response not success research) **then**
Select a reserve service from the list-WS-Init-Input;
End While; **END.**

3) Back-chaining algorithm:. In this discovery method, each peer searches the output of the service which can grow the composition process. This algorithm uses the List-WS-Init-Output (the list of the services which have an output equal to the init-output).

Begin
1. Create List-WS-init-Output (the list of Web services with the same Output).
2. **While** (List-WS-Init-Output not empty) **do**
a) Select the Head of List-WS-Init-Output;
b) Init-Output :=Input_of_Heading-List-WS-Init-Output; calculate the TTL;
c) Send message search (initiator- Peer, Iinit-Input, Init-Output, goal, TTL);
d) **If** (time ≤ TTL and response success research) **then**
Stop research; send a success response; go to End;
Else
If (time ≤ TTL and response not success research) **then**
Select a reserve service from the list-WS-Init-Output;
End While; **END.**

In addition, it is important to note if the composition is successful starting from a Back-chaining discovery, this composition will be saved in the composition table as follows: the initiator peer is the last participant peer and each participant peer permutes its precedent and its next went.

C. Data coherency of the composition table

The composition table presents a distributed way that stores and preserves data about precedent composite Web services discovered in an unstructured P2P network.

Instead of using a centralized repository of goals like in [1], the table of composition offers a distributed solution which has several advantages:

- Each Peer stores a historic of compositions, already realized in the network. In addition, each peer can exploit the experiment of other peers.
- The peer that participates in several compositions has a very rich table, which gives it more probability to discover composite Web services for future requests. This property encourages the peers to collaborate with other peers, in order to achieve common goals (this property minimizes the number of the egoistic peers in the network).
- The composition table permits to create a collaborative workspace. Peers have similar tables that can be grouped in collaborate communities. Automatically, peers that have the similar tables, they have the similar interests (this idea will be discussed in section 6).

However, the volatile nature of nodes in a P2P networks creates the main problem of such distributed architecture. This solution suffers from the problem of the data coherency of the composition table. If a participant peer leaves the network, all the compositions where it already participated become inactive, so each participant peer must be modified its composition table in this situation.

To ensure the data coherency of the composition table, each peer -before it leaves the network- must inform the other participant peers about any modified composition. Else, if a peer suddenly quits the network, one of their neighbors can inform the other participant peers. A peer can detect the absence of its neighbors in a new discovery operation. The other peers are informed in the tow ways from the initiator and the last peer in a composition. This operation is called the notification procedure. The following figure shows a scenario of a peer that participated in two compositions and launches a notification procedure before it leaves the network.

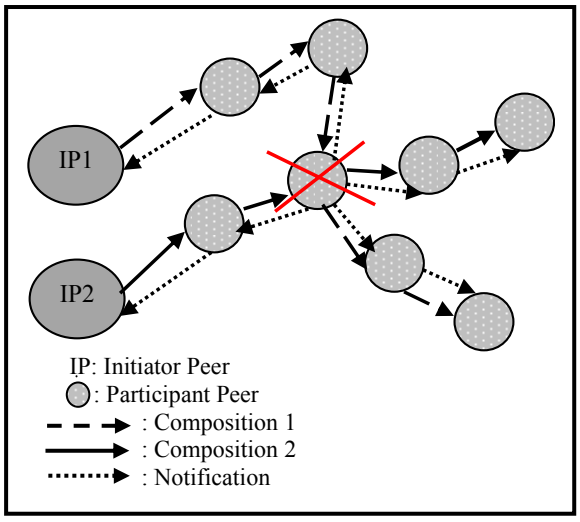

Figure 3. The notification procedure

For this, we define two types of notification algorithms: precedent-notification and next-notification.

1) Notification Algorithms

Each peer executes this algorithm when it receives a notification message form a peer (Next-Peer-I) that appears as a next peer in a composition stored in the composition table.

Notification-precedent

1. List-of-precedents:= Select « Initiator Peer, CompID, Precedent Peer » From the composition table Where Next peer= Next-Peer-I)
2. Send message to precedents peers of List_of-Precedents : «Initiator Peer, CompID, state of composite WS:= inactive»

The list of precedents contains the first's peers of each composition where the peer Next-Peer-I appears as a participant peer.

In the other side, each peer accomplishes the next algorithm if it receives a notification message from a peer (Precedent-Peer-I) that appears as a precedent peer in a composition stored in the composition table.

Notification-next

1. List-of-next:= Select « Initiator Peer, CompID, Next Peer » From the composition table Where Precedent peer= Precednt-Peer-I)
2. Send message to next peers of List_of-next : «Initiator Peer, CompID, state of composite WS:= inactive»

2) Reparation of a composition

In addition to the notification procedure, we can ameliorate this solution by giving the opportunity to repair the interrupted compositions. This is possible by replacing the absent peer by one of their reserve peers in each composition. The following algorithm presents the reparation of a composition by using the before chaining philosophy. In this case, the first precedent peer of the absent peer executes the following algorithm:

Repair-Composition

1. Select a reserve peer
2. Send a message: *Repair* (Initiator Peer, CompID, Init-Input, Init-Output, Goal, Next-Peer-J);
   If (reparation possible) then launch an notification message to the other participant peer;
   Else
   Select the next reserve peer; go to 2
3. End

The Next-Peer-J is the first next peer of the absent peer. The following figure present the message "*Repair*" by using a before chaining algorithm.

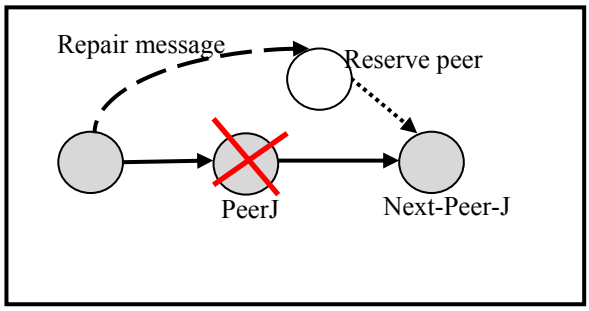

Figure 4. Reparation of a composition

If an initiator peer quits the network, only the second peer of the interrupted composition can repair the composition by launching a discovery procedure through using back chaining algorithm. This scenario will be more complicate if there are two or more second peers in a composition (parallel execution of Web services in a composition).

Finally, it's important to mention that the reparation procedure is not always possible in such architecture. Furthermore, the notification messages are very expensive when the number of compositions stored in the table is important. In addition this solution gives less of scalability because of the nature of the unstructured P2P networks (a proposition is presented in section 6).

IV. A DISTRIBUTED ARCHITECTURE TO IMPLEMENT THE DECOVERY STRATEGY

To implement the strategy presented in the precdent section, we proposed in [18] a distributed architecture composed from a whole of components allow to accomplish the different tascks defined in the epidimic algorithms defined priviously.

A. *A reference architecture*

The proposed model is composed from two main components: a framework installed on the various peers of the network and a central base of OWL ontologies [24] (and OWL-S descriptions [23]). This last is used as a reference to develop the various local ontologies and semantic descriptions of the different Web services of the network (in figure 5). The objective of the use of the same language of semantic descriptions (OWL-S) and a central base of ontologies is to ensure the homogeneity.

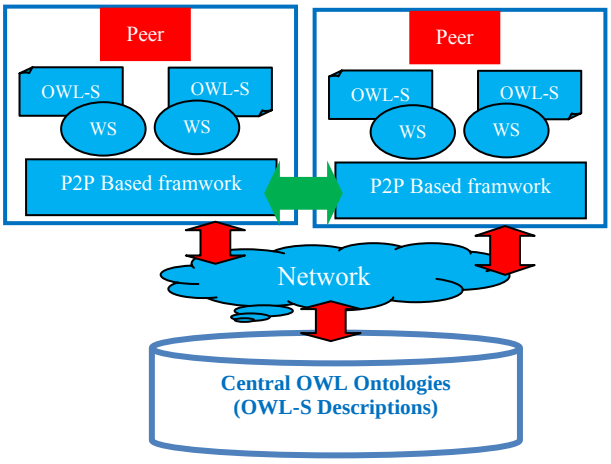

Figure 5. A reference Architecture.

The most important rules of this architecture are defined as follow:

- Each peer in the networks implements a number of Web services described semantically with OWL-S.
- The central ontologies base will be consulted periodically by the different peers to develop or enrich local ontologies of every peer. This operation is made in the passive time. So, it does not affect the discovery research time.
- The semantic descriptions of Web services remain local in each peer.
- The central OWL ontologies base contains the different concepts used in diverse fields of proposed Web services.

Also, the central base offers some OWL-S descriptions for various Web services. If a peer has a similar Web service, it can reuse these OWL-S descriptions. In this context, an example of a universal OWL ontologies and a collection of OWL-S descriptions for a variety of Web services is generated manually by Ganjisaffar and Saboohi which contains more than 240 semantic services descriptions [9].

Furthermore, P2P framework (like JXTA [30] or Gnutella [28]) is used for P2P communication. Thus, with such architecture, we can ensure that the discovery and the composition of Web services are purely decentralized.

B. *The P2P based Framwork*

The main idea of this framework is to make a purely distributed discovery of Web services. This distributed framework contains three layers: the semantic manager, the local composition engine and the P2P composition module (figure 6).

1) *The Semantic Manager module*

It manages the semantic descriptions of Web services. The user uses an OWL-S generator and a base of local OWL ontologies (figure 6 -arrow 2-). These last must be developed according to the central ontologies (figure 6 -arrow1-).

The OWL-S generator uses the WSDL descriptions and the ontologies of the fields to generate the OWL-S descriptions of Web services. In our case we used to wsdl2owl-s generator [10] as a core to implement this component.

2) *The Local Composition Module*

This module has as an objective to discover a local Web service to answer the external requests for the other peers. To achieve this goal, we propose two components: the local search engine and the local composition engine.

The **Local search engine** has two possible tasks: searching a basic Web service or an eventual local composite Web service.

When the peer receives a request from the network (figure 6 -arrow3-), the P2P composition engine passes the request to the local search engine (figure 6 -arrow4-). This last searches a basic Web service to answer this request (look step 2 in the main algorithm in section 3).

In the same time, the local search engine uses an OWL-S matchmaker to discover a possible composition from a whole of local Web services which responds to the request (figure 6 -arrow5-). As a result, we have three possible scenarios:

- If there is any basic web service or a probable composition, the local search engine returns a negative response to the P2P composition engine.
- If there is a basic Web service or an eventual composition. In the invocation step, the local search engine generates a BPEL file and sends it to the **local composition engine** (figure 6 -arrow6-), which uses the service invoker to invokes the basic Web services or a whole of Web services according the process defined in the BPEL file.
- If there is a probable semi-composition formed from a single Web service or a whole of Web services, the local search engine generates a request and proposes it to the P2P composition engine. This last can send this request to other peers in the network that can continue the discovery operation.

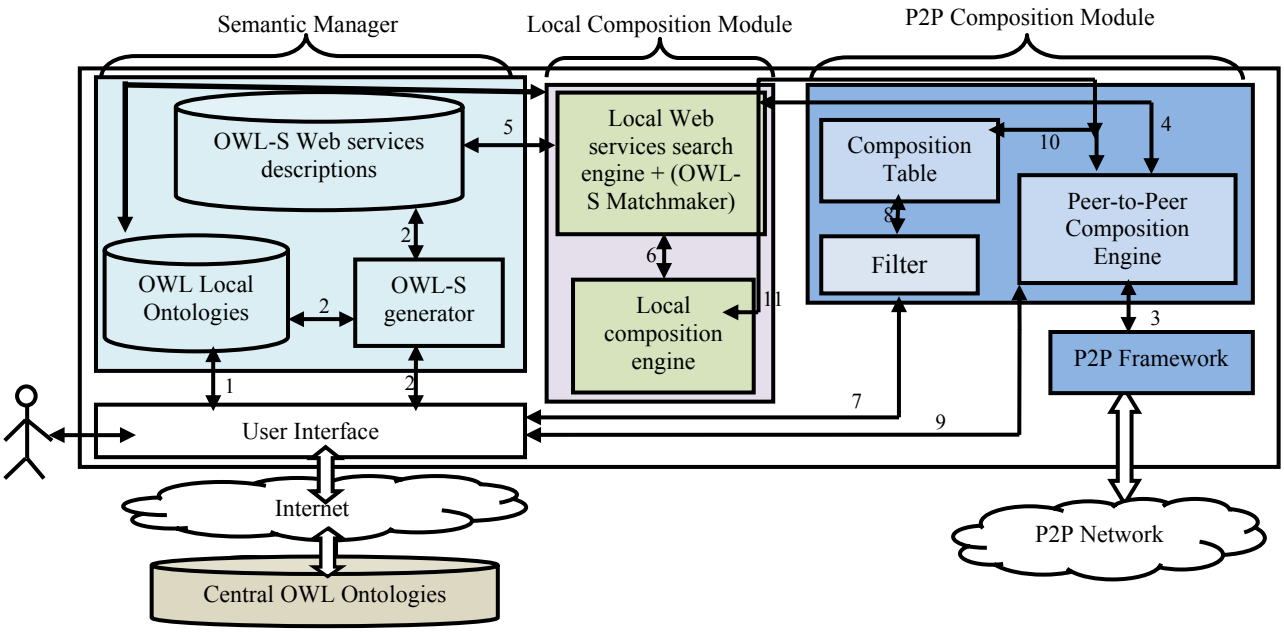

Figure 6. A Distributed Framework to discover and compose Semantic Web Services.

3) The P2P Composition Module

This module presents the interface between each peer and the P2P network. It's the main component used to realize a collaborative composition among a whole of participant peers. This layer contains three components: the composition table already presented in section 3, a filter and the P2P composition engine.

The filter is a program used directly by the user to clean the composition table from the inactive composition (figure 6 -arrow7-). This operation is very important in a dynamic environment like the P2P networks where many peers join and quit the network frequently. For this reason, there are many P2P compositions that become inaccessible when one or many participant peers are absent. The filter makes statistics about the composition table entrees to detect the compositions that became inactive since a long time (figure 6 -arrow8-). These statistics offer a clear vision to the user about the composition states in the table.

The P2P composition engine contains a request/response component to receive or to send requests (figure 6 -arrow9-). This component is used in many possible scenarios that are a relationship with the scenarios already explaining the local composition layer. Furthermore, the user can use the P2P composition engine to start a search operation in the network.

When it receives a request from another peer of the network (using the P2P platform) (figure 6 -arrow3-), the P2P composition engine sends the request to local composition engine. This last can return three possible responses:

- *A positive response*: in this case the P2P composition engine sends the response to the request peer.
- *A negative response*: the P2P composition engine evaluates a new TTL and transfers the request to other peers (the direct neighbors in the network).
- *A semi-composition proposition*: the P2P composition engine searches before in the composition table from a composition that responds to the main request or can continue the composition with the request of the semi-composition proposition (figure 6 -arrow10-). If there is a composition in the table, the P2P composition engine sends the request to the initiator peer of this composition. Else the P2P composition engine evaluates the TTL and continues the discovery using the request proposed by the semi-composition of the local composition engine.

In the end, if the discovery has been finished successfully, the initiator peer generates the BPEL file to launch the P2P composition. In this case, each participant used the P2P composition and the local composition engine to receive, execute and send the response (figure 6 –arrow 11-).

C. A Motivate example

In this example we want to improve the implementation of some components of the framework. This example presents a distributed application named "Constantine books" which gives the possibility to the user to search about a price of a book in US dollar, Euro or Algerian Dinar. The user can enter one or more input arguments like: authors, title of the book, edition home and the year of edition.

The following scenario shows how a new Web service is composed from three basic Web services:

- Search-ISBN: gives the ISBN of a book

- Search-Price: the input of this Web service is the ISBN of a book and the output is the price in US dollar.
- Convertor-Price: is a money convertor from the US dollar to EUR.

TABLE I.
THE INPUTS AND THE OUTPUTS OF WEB SERVICES

Web service	Inputs	Outputs
Search-ISBN	Title Author Edition Year-of-edition	ISBN
Search-Price	ISBN	Price
Convertor-money	Money-USD	Money-EUR

The composition of these Web services gives a composite web service with the following arguments:

Input: Title||Author||Edition|| year-of-edition

Output: Money-EUR

Goal: *input: =#book:Title||Author||Edition|| year-of-edition *ouput:=#book:Price//Money:EUR

In order to simplify the test, we choice to implement these Web services on two peers in the network (figure 7). The first peer which is the initiator peer deploys the Web service "Search-ISBN" and "Search-Price" and the second peer deploys the web service "Convertor-money".

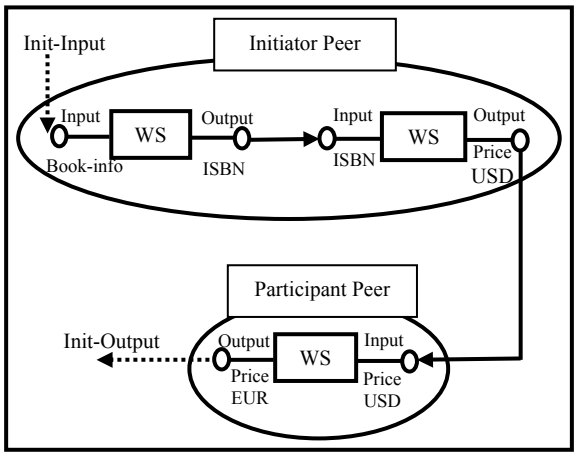

Figure 7. Example of a P2P composite Web service

When it receives the discovery request, the P2P composition engine of the initiator peer generates an OWL file and sends it to the search engine. The appendix A presents the generated request of this example (named request4.owl). In this example, we use an ontology called "Book.owl" (Appendix B). An execution example of the application is presented in appendix D.

D. Example of the implemenation of the search engine

To realize the search engine, we used a part of the implementation presented in [19]. The search engine has an input an OWL request and in output the whole of basic web services that response the request. Else, this list will used by the OWL-S Matchmaker to find a probably local composition or a semi-composition.

The search engine use a data base contains four tables that represent four classes presented as follow:

TABLE II.
STRUCTURE OF TABLE SERVICE DESCRIPTION

Attribute	Signification
OWL_URL	URL of an OWL-S description of a Web service. This attribute is the primary key of this table.
ONT_URL	URL of the local ontology used to generate the OWL-S description.
NAME	Name of the Web service.
Contact_Info	Information about the provider of the Web service
WSDL_URL	URL of the WSDL description of a web service

TABLE III.
STRUCTURE OF THE TABLE HAS-INPUT

Attribute	Signification
OWL_URL	URL of an OWL-S description of a Web service.
INPUT_CLASS_URL	URL of the class which represents an input of the Web service.

TABLE IV.
STRUCTURE OF THE TABLE HAS-OUTPUT

Attribute	Signification
OWL_URL	URL of an OWL-S description of a Web service.
OUTPUT_CLASS_URL	URL of the class which represents an output of the Web service.

TABLE V.
STRUCTURE OF THE TABLE HAS-GOAL

Attribute	Signification
OWL_URL	URL of an OWL-S description of a Web service.
GOAL_CLASS_URL	URL of the class which represents a goal of the Web service.

The main algorithm of the search engine is defined as follow:

1) Extract the input class, output class and goal class from the request;
2) Extract the descriptions of Web services from the data base;
3) For each description do:
   IF
   Number of input classes of the description ≠ number of input classes of the request **OR** Number of output classes of the description ≠ number of output classes of the request **OR** Number of goal classes of the description ≠ number of goal classes of the request
   Then
   Go to the next description;
   Else if
   If
   (Each input class of the request has an equivalent class **OR** super class **OR** under class in the description) **AND** (Each output class of the request has an equivalent class **OR** super class **OR** under class in the description) **AND** (Each goal class of the request has an equivalent

class **OR** super class **OR** under class in the description)
Then
Add the web service to the list ;
Return the list;

This algorithm is implemented by the method called "*findcalifiedcondidates*" (Appendix C). The other methods are defined as follow:

- **mySearchEngine :** the class constrictor
- **readServiceRequest :** read the request extract the input, output and goal classes.

V. DISCUSSION ABOUT FUTURE WORK

To evaluate the performances of our solution, we must test this framework in a large scale network. For this reason, now we are using NS2 [21] to simulate an unstructured P2P network and testing our solution by using a simple flooding protocol in the first time. After that, we plan to use GnutellaSim [22] to evaluate the performance of the proposed algorithm by using the Gnutella V0.4. Specially, we need to estimate the effect of the semantic matching on the computational complexity. Moreover, we want to deduct how the composition table can accelerate the discovery operation through the re-use of the compositions already carried out in the network.

The simulation step consists to define the basic observations to show the performances of the localization process of semantic Web services by flooding the network. The main properties that we want to observe in the case of a pure unstructured network are:

- The number of the relevant requested peers and the total number of requested peers.
- The number of propagated requests, the number of transmitted results and the number of messages necessary for network operating.
- The adequate TTL to return a result.
- The adequate size of the composition table.

As a future work, we plan to ameliorate this solution by regrouping peers that have similar Web services. This proposition can resolve some problems posed by the unstructured architecture. In this situation, each group is managed by one Super-Peer. The composition table is stored in the super-peer. Each table saved all the compositions where the members of the group are already participated.

For example, for a composition, if a participant peer quits the network; his Super-Peer tries to repair the composition by replacing the absent peer with another from the group. In this case, the reserve peers are generally belongs to same group of the absent peer. If the reparation is not possible locally, the Super-peer can send a *"repair message"* or a "*notification message"* to the other Super-Peers (figure 8).

To simulate this architecture, we plan to use the Gnutella V.0.6 [29]. After that, we can compare the two architectures (unstructured and Super-Peer).

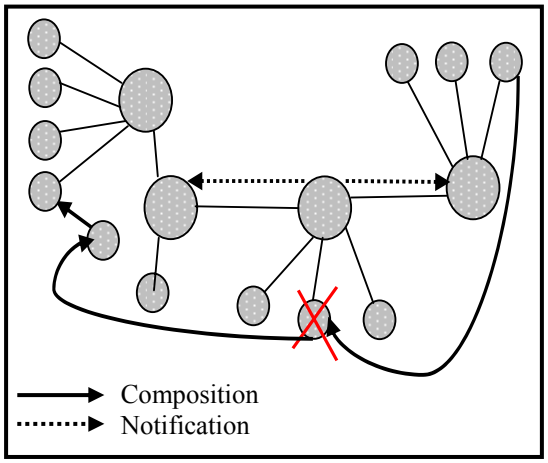

Figure 8. The notification procedure in the Super-peer model

VI. RELATED WORKS

Recently, several research works improve the decentralized discovering methods of Web services in the P2P networks. A number of centralized and P2P Web service discovery methods have been proposed in the context of the Web services composition and Web services based business process management. Among these, [1], [4], [5] and [6] have similar concepts to those which are used in our method.

F. Mandreoli et al [1] present the architecture of FLO^{2}WER, 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 FLO^{2}WER 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 addition, a central repository creates single point of failure and a centralized control of published services. Then, this solution suffers from problems such as high operational and maintenance cost. In contrast, our discovery method is a pure decentralized solution with any central repository. The composite Web services already realized in the network are published with a distributed description among all participant peers.

T. Essafi and al [6] presents a scalable P2P approach to service discovery using ontology. This work incorporates input/output matching algorithm proposed in paper [26] and extends the solution described in paper [25] by adding an encoding that locates servers in a P2P network to simplify rerouting of query messages. Idem to

the precedent work [1], this project adopts network centralization and hierarchy, which suffers from the phenomenon of the "single-point" failure. However, the main objective of our work is to ensure dynamic Web service discovery without central point. Also, in this work, they still relay on the old DAML-S, which proved to be less flexible than the new OWL-S.

[2], [4] and [5] define a composite web service as finite automata. J. Hu et al [2] and F. Emekci et al [4] propose a structured P2P framework for Web service discovery in which Web services are located based on both service functionality and process behavior. They represent the process behavior of the Web services with finite automata and use these automata for publishing and querying the Web services within the system. This framework is scalable due to the underlying P2P architecture because the Web services can join and leave the system dynamically. In our work, we represent the process behavior of the Web services by finite automata (a finite sequence of basic Web services belong to several peers). In addition, in our context we proposed an unstructured P2P solution where the participant peers and the Web services (which compose the resulting service) are not beforehand known. In addition, we publish the P2P composed Web services with a distributed table (composition table). If a peer wants to re-use a Web service which has been already composed, it transfers the request from the initiator peer of this composite Web service.

VII. CONCLUSION AND PERSPECTIVES

In this paper, we proposed an unstructured P2P Model for semantic Web service discovery and composition in which Web services are distributed among all peers of the network. In this model, we defined firstly a strategy based on an epidemic algorithm to discover the adequate basic Web service or to discover a P2P composite Web service, which answers a received request. Furthermore, we defined to philosophies (before and back chaining) to progress the discovery process in the network. The second pertinent idea in this strategy is the use of composition table provides a purely distributed method to discover the previous P2P composed Web services. The distribution of this table creates a collaborative workspace where each peer can exploit the experiment of the other peers. Furthermore, this table permits to preserve the trace of the various successes compositions for a possible future re-use. This characteristic can ameliorate the research time in the network. Also, we proposed a whole of algorithms to ensure the data coherency of the composition table by notifying the other peers about the absent of a participant peer or by repairing an interrupted composition.

The second part of this work presents a distributed framework, which implement the previously presented strategy. The main goal of this framework is to offer a collaborative workspace between a whole of peers where each peer can offer their Web services and can exploit the resources of other peers. Composing a whole of Web services belong to different peers of the network is an important task to achieve a goal that not realized by one or many Web services of a single peer. Furthermore, we presented a motivate example to improve the implementation of this framework (some program codes are presented in the appendixes). Finally, we suggest some future works to ameliorate the proposed solution.

However, this work needs to improve some important points. Especially we hope to improve the QoS of the returned results by proposing some selection criteria of the Web services. Also, wish to develop the implemented algorithms using a probabilistic approach to filter the pertinent peers in the network. Furthermore, we plan to improve our solution proposed in [20] to give a formal verification for the composition of semantic Web services in a P2P network.

APPENDIX A SEARCH BOOK PRICE (REQUEST4.OWL)

```
<?xml version='1.0' encoding='ISO-8859-1'?>
……..
  <!ENTITY domainOnt "..\ontologies\\book.owl">
  <!ENTITY DEFAULT   "..\request4.owl">
<rdf:RDF
…….
  xmlns:service=   "&service;#"
  xmlns:process=   "&process;#"
  xmlns:profile=   "&profile;#"
……..
  <owl:Ontology about="">
    <owl:imports rdf:resource="&rdf;" />
    <owl:imports rdf:resource="&rdfs;" />
    <owl:imports rdf:resource="&owl;" />
    <owl:imports rdf:resource="&service;" />
    <owl:imports rdf:resource="&profile;" />
    <owl:imports rdf:resource="&process;" />

    <owl:imports rdf:resource="&domainOnt;" />
  </owl:Ontology>
  <profile:Profile rdf:ID="getBookPrice">
   <profile:hasInput rdf:resource="#input1-Edition"/>
  <profile:hasOutput rdf:resource="#output1-bookPrice"/>
  </profile:Profile>
 <process:Input rdf:ID="input1-Edition">
  <process:parameterType rdf:resource="&domainOnt;#Price"/>
  </process:Input>
 <process:UnConditionalOutput rdf:ID="output1-bookPrice">
    <process:parameterType rdf:resource="&domainOnt;#Price"/>
  </process:UnConditionalOutput>

</rdf:RDF>
```

APPENDIX B BOOK.OWL

```
<?xml version="1.0"?>
<rdf:RDF
……
<owl:Class rdf:ID="Book">
</owl:Class>
<owl:Class rdf:ID="ISBN">
</owl:Class>
<owl:Class rdf:ID="Price">
</owl:Class>
<owl:Class rdf:ID="Money">
</owl:Class>
<owl:Class rdf:ID="Title">
</owl:Class>
<owl:Class rdf:ID="year">
</owl:Class>
<owl:Class rdf:ID="author">
</owl:Class>
<owl:Class rdf:ID="Edition">
</owl:Class>
```

Classes

```
<owl:ObjectProperty rdf:ID="has_ISBN">
<rdfs:domain rdf:resource="#book"/>
<rdfs:range rdf:resource="#ISBN"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_Price">
<rdfs:doamin rdf:resource="#book"/>
<rdfs:range rdf:resource="#Price"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="has_Title">
<rdfs:domain rdf:resource="#book"/>
<rdfs:range rdf:resource="#Title"/>
</owl:ObjectProperty>
……
……
```

} Proprieties

```
</rdf:RDF>
```

APPENDIX C SEARCH ENGINE IMPLEMENTATION

REFERENCES

[1] F. Mandreoli, A. M. Perdichizzi, and W. Penzo, "A P2P-based Architecture for Semantic Web Service Automatic Composition", *IEEE computer Society DOI 10.1109/DEXA*2007, Regensburg Germany, 2007, pp. 429-433.

[2] J. Hu, C. Guo, H. Wang, and P. Zou, "Web Services Peer-to-Peer Discovery Service for Automated Web Service Composition", *Springer-Verlag Berlin Heidelberg (LNCS 3619)*, *ICCNMC 2005*, Zhangjiajie China, 2005, pp. 509-518.

[3] M. Gharzouli and M. Boufaida, "A Generic P2P Collaborative Strategie for Discovring and Composing Semantic Web Services", *In the Proc of Fourth International Conference on Internet and Web applications and Services(ICIW'09)*, Venice/ Mestre, Italy 2009, pp. 449-454.

APPENDIX D AN EXECUTION EXAMPLE OF THE APPLICATION "CONSTANTINE BOOKS"

[4] F. Emekci, O.D. Sahin, D. Agrawal, and A. El Abbadi, "A Peer-to-Peer Framework for Web Service Discovery with Ranking", *Proc of the IEEE International Conference on Web Services (ICWS'04)*, California USA, 2004, pp. 192-199.

[5] O. D. Sahin, C. E. Gerede, D. Agrawal, A. El Abbadi, O. Ibarra, and J. Su, "SPiDeR: P2P-Based Web Service Discovery", *In the Proc of ICSOC'05*, Amsterdam, The Netherlands, 2005, pp. 157-169.

[6] T. ESSAFI, N. DORTA and D. SERET, "A Scalable Peer-to-Peer Approach To Service Discovery Using Ontology", *In the Proc of 9th World Multiconference on Systemics, Cybernetics and Informatics.* Orlando, 2005.

[7] R. Schollmeier, "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications", *In the Proc of the First International Conference on Peer-to- Peer Computing*, Linkoping, Sweden, 2001, pp. 101-102.

[8] K. Verma, 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", *Journal of Information Technology and Management, Special Issue on Universal Global Integration, Vol. 6, No. 1*, 2005, pp. 17-39.

[9] Y. Ganjisaffar and H. Saboohi, "Semantic Web Central: Project sws-tc", 2006, http://projects.semwebcentral.org/projects/sws-tc/

[10] J. Giampapa, M. Paolucci, N. Srinivasan and R. Vaculin"Semantic Web Central: Project wsdl2owl-s, translator from wsdl to owl-s", 2004, http://projects.semwebcentral.org/projects/wsdl2owl-s/

[11] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker "Search and Replication in Unstructured Peer-to-Peer Networks", *Proc of the 16th international conference on Supercomputing*, New York, USA, 2002, pp. 84-95.

[12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications", *In Proceedings of the ACM SIGCOM,* California, USA, 2001, pp.149-160.

[13] K. Rageb "An Autonomic <K, D>-Interleaving Registry Overlay Network for Efficient Ubiquities Web Services Discovery Service", Journal Information Processing Systems (JIPS) Vol. 4, no.2, June 2008.

[14] S. Batra, S. Bawa "Review of Machine Learning Approaches to Semantic Web Service Discovery", Journal of Advances in Information Technology (JAIT) vol. 1, no. 3, August 2010.

[15] Schmidt, A., Winterhalter, C. (2004), "User Context Aware Delivery of E-Learning Material: Approach and Architecture", Journal of Universal Computer Science (JUCS) vol.10, no.1, January 2004

[16] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", *In the Proc of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware* 2001), Heidelberg, Germany, 2001.

[17] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network", In *Proc of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications,* ACM SIGCOMM, California, USA, 2001, pp. 161-172.

[18] M. Gharzouli and M. Boufaida, "A Distributed P2P-based architecture for semantic Web services discovery and composition", *In the Proc of 10th Annual International Conference on New Technologies of Distributed Systems (NOTERE)*, Tozeur, Tunisia 2010, pp. 315-320.

[19] L. Yu "Introduction to semantic web and semantic web services ", Chapman & Hall/CRC, 2007.

[20] D. Benmerzoug, M. Gharzouli and M. Boufaida "Formalisation and Verification of Web Services Composition based on BPEL4WS", *Proc of First Workshop of Web services in Information Systems (WWS'09)*. Algies, Algeria, 2009, pp. 37-47.

[21] The network simulator -ns2-, available in http://www.isi.edu/nsnam/ns/

[22] A scalable packet-level Gnutella simulator, available in http://www.cc.gatech.edu/computing/compass/gnutella/

[23] OWL-S: Semantic Markup for Web Services, available in http://www.w3.org/Submission/OWL-S/

[24] OWL: Web Ontology Language, available in http://www.w3.org/TR/owl-features

[25] M. Paolucci, K. Sycara, T. Nishimura and N. Srinivasan**,** "Using DAML-S for P2P Discovery", *In the Proc of International Conference on Web Services (ICWS).* Las Vegas, Nevada, USA, 2003, pp. 203-207.

[26] M. Paolucci, T. Kawamura, T.R. Payne and K. Sycara, **"**Semantic Matching of Web Services Capabilities", *in the Proc of the First International Semantic Web Conference (ISWC)*, Sardinia, Italy, , 2002, pp. 333-347.

[27] Service finder: a search engine for web services discovery, available in www.service-finder.eu

[28] The annotated Gnutella protocol specification v0.4, available in: http://rfcgnutella.sourceforge.net/developer/stable/index.html

[29] Gnutella protocol Development v0.6, available in: http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

[30] JXTA a P2P platform, available in: https://jxta.dev.java.net/

[31] Web services Architecture, W3C Working group, 2004, available in http://www.w3.org/TR/ws-arch/

Mohamed Gharzouli was born in Constantine (Algeria). He received his BS degree in Computer Science from Mentouri University of Constantine (Algeria) in 2002, and MS degree in Computer Science from Larbi Tebessi University of Tebessa (Algeria) in 2004. Currently, He is working as Assistant professor in Department of Computer Science in Mentouri University of Constantine (Algeria) since 2006. He is a member of the research group 'Information Systems and Knowledge Bases' in LIRE Laboratory (Constantine, Algeria). He supervised many Master and License students. Since October 2007 until now, he is preparing his Ph.D in Computer Science. He has published a number of articles in International Conferences. He is a program committee member of ICIW conference. His research interests include Web services, Web Semantic, P2P Networks, Web Applications and Distributed Applications.

Mahmoud Boufaïda is a full professor in the Computer Science department of the University of Constantine, Algeria. He heads the research group 'Information Systems and Knowledge Bases'. He has published several papers in international conferences and journals. He has managed and initiated multiple national and international level projects including interoperability of information systems and integration of applications in organizations. He has been program committee member of several conferences. His research interests include cooperative information systems, web databases and software engineering.