Improving Resource Discovery and Query Routing in Peer-to-Peer Data Sharing Systems Using Gossip Style and ACO Algorithm

Hamdi Hanane, Benchikha Fouzia
LIRE laboratory, Department of Software Technology
and Information Systems, University of Constantine 2
Constantine, Algeria
emails: {Hamdihanane@hotmail.fr, f_benchikha@yahoo.fr}

***Abstract*—With the far-reaching significance of the Internet and the drastic advances in computer technology, more and more news and data are available on the Web. Peer-to-Peer (P2P) systems have become a popular way of sharing these data, and have drawn much attention of both academia and industry. The key and one of the most challenging design aspects in data sharing in P2P systems is how to find flexible, scalable and efficient mechanisms for searching and retrieving data. In the proposed approach, we use mobile agents to take advantage of a distributed system. To optimize the migration strategy of the mobile agents, we first resort to the biological behavior of ant colonies; the agents use trails leaved by other agents on peers they have visited. Then, to enhance the quality of the search results, reduce the randomness of peers and preserve their autonomy, we also introduce a social aspect, i.e., the friends list is used to gather peers having similar center of interest. To discover friends, the peers rely on the gossiping algorithm. We find that our contribution has three originalities distinguishing it from other approaches. The first one takes into account the two principal issues in data sharing in P2Pdatabase systems. The second one has the advantage to be totally independent of the centralized management. Finally, the third one is inherent to the resource discovery mechanism, which includes a social aspect using the friendship links.**

Keywords-P2P; Mobile agent; resource discovery; gossiping; ant colony optimization.

I. INTRODUCTION

With the far-reaching significance of the Internet and the drastic advances in computer technology, more and more news and data are available on the Web. Peer-to-peer systems have become a popular way of sharing these data, and have drawn much attention of both academia and industry. According to [1], P2P file transfer occupies 86.7% of the total file transfer traffic. The basic principle of P2P technology is that the peer acts as both a client and a server.

In the proposed research, we are interested in data sharing in P2P databases systems. Peer-to-Peer Data Management Systems (PDMS) have emerged recently; they combine P2P technology and distributed databases *Piazza* [2], *SomeWhere* [3] and *PeerDB* [4].

Although, coupling data integration techniques and P2P systems is efficient, it is essential to overcome some obstacles mainly due to the heterogeneity, decentralization and dynamic nature of P2P. In a P2P network, it is almost impossible to build or agree upon a mediation schema, with nodes joining and logging out continuously. Due to the absence of a centralized control, peers do not know the *a priori* location of the data they are looking for. The key and one of the most challenging design aspects here is to find flexible, scalable and efficient mechanisms for searching and retrieving data.

To take advantage of a distributed system, we must look for a distributed management solution. As the P2P networks, the paradigm of the mobile agent has been specifically developed for dynamic, distributed, open and heterogeneous environments. On behalf of network users, mobile agents execute software entities that are capable to migrate from one node to another in heterogeneous networks. Lange et al. [5] showed that mobile agents not only reduce the network load and overcome its latency, but also encapsulate protocols, execute asynchronously and autonomously, and dynamically adapt to changes.

Here, we rely on the mobile agents' intelligence and their capability of adapting their migration itinerary to the changing conditions. In that sense, in the proposed approach, we are inspired by the biological behavior of ant colonies. To optimize their migration strategy, mobile agents use trails leaved by other agents on peers they have visited. In addition, a mobile agent can make measurements anywhere on the network and take real-time decisions as well.

We also introduce a social aspect through the use of the friends list to gather peers having similar center of interest. This reduces the randomness of peers, which leads to better search results. It also enables robust self-monitoring and preserves the peer's autonomy. To discover their friends, the peers rely on the gossiping style; which proved to be very efficient for supporting dynamic and complex information exchange among distributed peers. They are useful for building and maintaining the network topology itself, as well as supporting a pervasive diffusion of the information injected into the network.

As the heterogeneity issue has been treated in a previous work [6]; in this paper, we mainly focus on the resource discovery and routing queries. The rest of paper is organized as follows. In Section 2, we first introduce a comparative study of the multiple research works then, to help understand the proposed approach, we present some of its

 ISBN: 978-1-61208-368-1

relevant concepts. Next, in Sections 3, 4 and 5, we lay out the details inherent to the development of the approach. A summary of the results of the contribution along with the conclusions are given in Section 6.

II. BACKGROUND AND RELATED WORKS

Several authors focus on resolving the issue of data sharing in a peer-to-peer network. Research works [2][3][7] are interested in the resource discovery and the query routing issues. In a topology independent of central process, band failure and bottlenecks, ones' challenge is to offer a resource discovery method with a maximum accuracy. In this context, various techniques are proposed in the literature. They depend essentially on the network topology, such as distributed hash tables, centralized repository, semantic overlay networks (super peer) and flooding.

King [8] uses the Distributed Hash Tables (DHT) keep a tight control on the structured network and often find the desired information by means of the query for unique identifiers. However, this technique requires knowledge of the identifiers before any query is processed, and is too sensitive to failure. Moreover, each peer depends on other peers; which limits its autonomy. Indeed, the algorithm is based on the concept of successor, which may turn out to be faulty. Finally, this technique handles poorly keyword searches and complex queries.

The central repository, such as *BitTorrent* [9], relies on a centralized topology. A central repository is used to index all the peers and their respective data. This technique does not satisfy the distributed control criteria for P2P systems. It also creates a bottleneck, and limits scalability by concentrating all the resource information at a single point. In fact, if the server crashes, the whole network stops working.

Owing to its numerous advantages, the overlay semantic networks or the commonly known Super-Peer topology, used in *Piazza* [2] and *SenPeer* [7], has become very popular. This topology has though a major drawback, namely, its mishandling of client/server at a group level. For instance, if for any raison, the Super-Peer fails, all peers turn out to be unavailable to the assignment of a new Super-Peer. Furthermore, the queries are sent only to related semantic groups. In this case, groups that do not have any semantic relationship with the query are simply ignored.

The flooding technique *Gnutella* [10] uses a recursive process to send a or query to all nodes in the network. Thus, it does an exhaustive search. To avoid messages travelling indefinitely in the network, the number of nodes a message can visit is set to a limit called Time To Live (TTL) [10]. PeerDB [4] proposes a different approach for the flooding technique. In this case, mobile agents are used for query routing and processing as well.

Other works in the literature use gossip style for information dissemination in the network. In *TRibler* [11], a P2P television recommender system is proposed. The authors suggest a social networking system, built on a P2P network which is *Bittorrent-based* file-sharing client [9]. To establish friendship links between the source and the target peers, this system relies on a gossip style. More P2P Recommendation Systems *are found in P2Prec* [12] and *P2PREcommender* [13].

Further research works, such as [3][9], are interested in schema mediation; omitting the resource discovery and the query routing aspects. Most of the peer-to-peer systems only deal with non structured or semi-structured data *Gnutella* [10]. However, some search approaches allow sharing structured data as well as doing searches based on its content. Most of these works are based on ontology to address the heterogeneity issue, such as global ontology [8][6] and local ontology approaches [3].

Before introducing the proposed approach in Section 3, let us first get familiar with some of its relevant concepts, i.e., P2P-database, Ant Colony Optimization algorithm (ACO) and gossiping protocol (also known as the epidemic protocol).

A. Peer-to-Peer databases systems

A P2P database system (PDBS) is perceived as a collection of autonomous local repositories which interact in a peer-to-peer style [2]. On the other hand, a Peer Data Management System (PDMS) is a triplet-set, i.e., S = <P; S; Mi> where, P, S and M are sets, respectively of autonomous peers, heterogeneous schemes and schema mappings; each of which enables the reformulation of queries between a given pairs of schemas [15]. A PDMS is a distributed data integration system providing transparent access to heterogeneous databases without resorting to a centralized logical schema. Instead of imposing a uniform query interface over a mediated schema, PDMSs let peers define their own mappings, supplied locally with regards to different schemas pairs or groups of pairs, to be used to reformulate queries.

B. Ant Colony Optimization (ACO)

The Ant Colony Optimization (ACO) [14] algorithm is inspired by the behavior of ants colonies. That is, the ants mark their trails through different intensities of pheromones, i.e., secreted chemical substances to communicate between them. In doing this, they create a system in which the best route, chosen by the others ants, would correspond to the highest level of pheromones. An important characteristic of this algorithm is to dynamically absorb changes in the graph. This makes this characteristic practical in dynamic network routing systems and also suited to P2P networks [14].

C. Gossiping

Gossiping also known as *epidemic* protocols were first introduced in 1987 by Demers et al. [16], who employed them in propagating updates in loosely replicated databases. These protocols have the effect of maintaining mutual consistency among the replicas. Gossiping protocols have mostly been associated with dissemination of information [17]

By omitting specific details, gossiping protocols work through a simple model. Each node has a complete view of the network, and periodically picks a random node from the

 ISBN: 978-1-61208-368-1

whole network to exchange data with. This allows information, known by any node, to spread to the whole network with very high probability [17].

III. PROPOSED APPROACH

As a solution to the two principal issues for data sharing in P2P network, we introduce, in this section, the proposed approach.

To address the heterogeneity issue, the proposed approach relies on domain ontology. This solution is easy to implement. It takes into consideration the P2P characteristics, namely, autonomy, scalability and decentralization. In addition, to the advantages of semantics in data integration, the ontology provides users a common vocabulary to ensure unambiguous communication between heterogeneous sources. Each peer has a copy of the domain ontology. When joining the network, the peer must provide mapping between its local schema and the domain ontology. These mappings are stored locally and used for subsequent queries reformulation.

As stipulated in Section 1, here we are not concerned with providing solutions to the heterogeneity issue. We rather focus on the resource discovery and query routing issue. In doing this, we propose an agent-based architecture which obeys the P2P criteria. We, then, propose a mechanism of resource discovery and query routing, which takes advantage of using mobile agents and optimizes their migration strategy by avoiding an excess of message traffic through the use of the ACO. Finally, introduce a social aspect using the friends list to group together peers having similar center of interest. This reduces the randomness of peers, which leads to better search results. It also enables robust self-monitoring and preserves the peer's autonomy. To build and manage this friends list, we use the gossip style.

IV. AGENT-BASED ARCHITECTURE

Our solution is a distributed system evolved in a dynamic environment with peers joining and leaving the system.

When it comes to designing this type of system, agent technology is suitable, because multi agent systems not only allow the sharing or distribution of knowledge, but also the achievement of a common goal.

A. *Mobile agent technology*

In a broad sense, an *agent* is any program that acts on behalf of a (human) user. Then, a *mobile agent* is a program which represents a user in a computer network. To perform some computation on user's behalf, this program is capable of migrating *autonomously* from one node to another. In order to make possible previous difficult fault tolerance and distribution heuristics, agents are also able to perceive their environment and communicate with other agents [18].

Using agents in P2P networks offers many advantages.

- Reduction of the use of bandwidth and communication costs.
- Ability of adapting to a dynamical P2P environment.
- Asynchronous execution and fault tolerance.
- Autonomy.
- Cloning and dispatching of a mobile agent in different directions.

B. *Overview of the architecture of a peer*

As shown in Figure 1, a PDMS, as defined in Section 2.A, consists of a network of nodes referred to as peers.

Since PDMSs are a mechanism of a decentralized sharing of data, mappings are not controlled in any central manner. The only assumption we can make is that any peer that joins the system provides some mappings between its local schema and the domain ontology. Figure 1 shows an overview of the architecture of a peer. It is formed by different parts, which are succinctly described as follows.

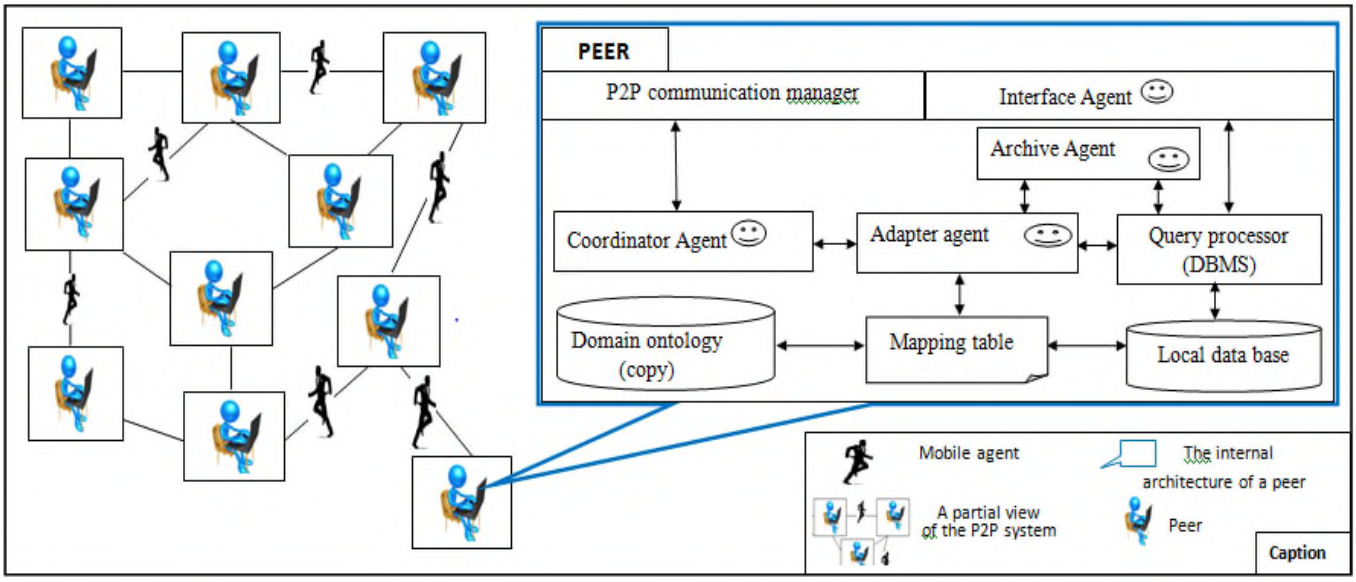

Figure 1. Overview of the proposed architecture.

- P2P network: Our approach relies on a pure unstructured P2P topology. The network is composed of a set of peers interconnected in an unstructured manner.
- Peers or users of the network: A set of peers participating in data sharing. Each peer has a partial view of the network, and plays the role of client, server and mediator.
- Mobile agents: The mobile agent travels over the network to meet the requirements of its initiator peer.

Here, the general architecture of a peer, shown in Figure 1, satisfies not only the P2P characteristic, but also the requirements of the proposed approach.

A. The internal structure of the Peer

In a P2P network, each peer must be able to play the role of-data provider, data requester and mediator.

To fulfill these roles, each peer must contain at least a:

- Data source,
- Copy of the domain,
- Set of mappings between the elements of the domain ontology and the local schema.
- Set of agents responsible of information retrieval and data mediation.

According to the proposed internal architecture of a peer, Figure 1, there are five types of agents.

1) Interface Agent (IA): It is a stationary agent that receives all user queries and process them locally. If necessary, the *IA* sends the processed queries to the coordinator agent. When the *IA* receives results, it presents them to users.

2) Coordinator Agent (CA): This agent coordinates and manages the other agent's work. It also creates mobile agents and dispatches them over the network. The social behavior of the peer is also managed by this agent; it is responsible of the creation and the updating of the friends list. Figure 2 shows the internal structure of the *CA*.

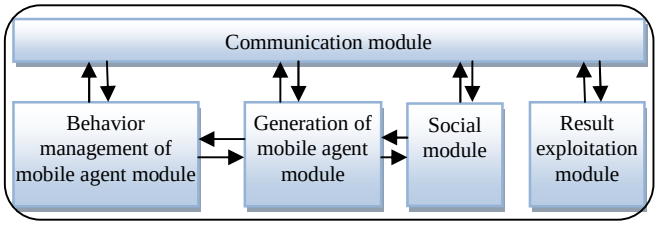

Figure 2. Coordinator agent.

3) Social Mobile Agent (*SMA*): This agent is created by the *CA*. To collect data, the *SMA* travels over the network. Figure 3 shows the proposed *SMA* architecture. It contains

- A communication module to communicate with its *CA*, any peer it visits, and any other agents it may meet.
- A mobility manager and a cloning manager module. The social module is responsible of the social behavior of the mobile agent; it holds the interest list of its Creator Peer (CP) and measures its matching inherent to the interest list of other agents it may meet.
- A result exploitation module which is responsible of the query execution results.

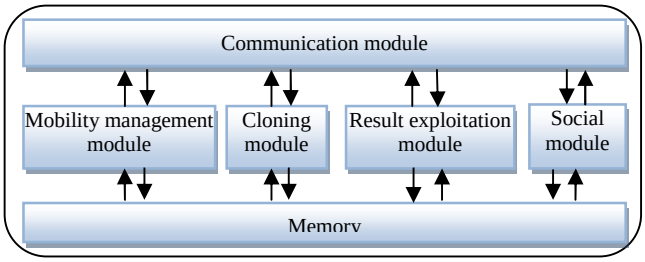

Figure 3. Social mobile agent.

4) Adapter Agent (AdA): This agent translates the user's request from the local language (local schema) to the global language (domain ontology), and vice versa. To do so, it uses the mapping tables created by the peer when it joins the network for the first time.

5) Archive Agent (ArA): This agent stores the resolved queries at time T, and checks its memory for each new peers request, to know whether or not this request has been already resolved earlier. It also memorizes the trails leaved by the mobile agents which have visited this peer.

B. Functioning Principle

1) Basic steps

To better understand the proposed approach, we consider a motivating example of an application for data sharing in a P2P network. We are interested in data sharing in the field of medical research; therein, it is mandatory to share data between different research institutes, doctors and scientists. To this end, let us consider a network of scientists interested in sharing the locally stored data. The scientists typically store different kinds of data, including papers and reports they have written, articles and theses they have downloaded, information about conferences, seminars and other scientific events, reports about patients they have treated, etc. The goal of such a network is that each user (peer) can discover data, provided by other peers (users) and sends queries to them, accordingly. For instance, to exchange reliable information with one another, one may want to know which peer has already treated a patient who has lived with Alzheimer's disease for 10 years. Likewise, another peer may also want to look for datasets used by others for some kinds of experiments. In this last scenario, each scientist should have its own database. In order to overcome heterogeneous schemas, databases must undergo an integration to allow scientists access data from other researchers in a transparent manner, irrespective of problems of heterogeneity or distribution.

To this effect, let us assume that a new researcher noted Pi joins the network, and would like to make a research. Here are the basic global steps of the proposed mechanism; from Pi joining the network until the acknowledgement of a query. The details of these steps are left for Section 5.

 ISBN: 978-1-61208-368-1

- Pi, first builds the mappings table between its local schema and the domain ontology. These mappings are stored locally to be used later for query reformulation.
- Next, he builds its friends list from its neighbor list which will be detailed in Section 5.1
- Then, he makes, on his local schema, a query about the search he would like to do. Only then, the query is translated into the vocabulary of the domain ontology.
- Pi creates mobile agents and sends them along with the query over the network.
- When a mobile agent arrives at a peer Pj, it transmits it the query, which prior to running it, translates it into the vocabulary of its local schema.
- The mobile agents come back to their initiator peer, Pi along with the data and the information they have collected.

2) Mobile agent life cycle

The life cycle of a mobile agent, Figure 4, designates all progression steps of the agent; from its creation until its death. The mobile agent creation can be initiated either by a user agent or by another mobile agent. Figure 4 illustrates the life cycle of our mobile agent.

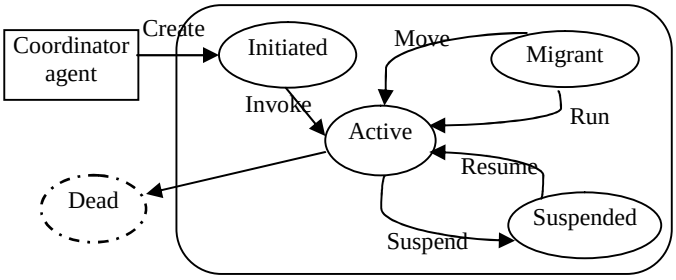

Figure 4. Mobile agent life cycle.

Initiated: The agent has been created but has not registered yet. As it has neither a name nor an address, it cannot communicate with other agents.
Active: The agent has been registered and has a name. In this state, it can communicate with other agents.
Suspended: The agent is stopped because its thread is suspended.
Migrant: The agent is moving to a new location.

Dead: The agent has finished and his thread has ended his execution and it is not any more in the AMS.

V. GOSSIPING AND ACO BASED RESOURCE DISCOVERY

As explained earlier, resource discovery and query routing are crucial issues for data sharing inP2P networks. Using mobile agents, in this section, we propose a mechanism of resource discovery and routing queries that couples the gossiping with the ant colony optimization.

Our mechanism is divided into two parts.

i. The first one deals with building and managing the friends list. That is, we exploit the gossiping to discover and group together users having similar interests "Friends".

ii. The second one deals with guiding the agents in their migration phase. Using the links established so far with the ACO, we optimize the routing strategy of the mobile agents.

This mechanism reduces the randomness of peers, which leads to better search results. It also enables robust self-monitoring and preserves the peer's autonomy. In such a way, it eases information dissemination, as peers discover new content and new peers. Finally, it is robust, resilient to failure and easy to implement.

a) Building and managing the friend list

Our work differs from the conventional gossiping model through the following points.

First, we leave out the assumption of complete knowledge of the network. Each node has only a partial view of the network.

Then, the type of data is such that nodes exchange their list of interest.

After a gossip exchange, nodes update their friends list to incorporate a part of the received links. In doing this and continuously refreshing their friends list, nodes can self-organize into better and suitable topologies.

Basically, our gossip protocol proceeds as follows. A peer Pi knows a group of other peers or contacts, which are maintained in a list called Pi's view. Periodically, with a gossip period noted Tgossip, Pi picks a random neighbor Pj from its view to gossip. Then, peers Pi and Pj exchange information. The gossip algorithm we propose here is inspired by gossip-based approaches for P2P membership management [17].

Each peer has a partial view of the network. It knows a small but continuously changing set of other peers. This view is divided into two lists. While the first one is the list of neighbors assigned by the system using bootstrapping, i.e., neighbors list; the second is the list of peers having in common the same centers of interest, i.e., friends list.

Note that each peer also maintains a list of interest. This list contains keywords that represent the issues and research topics for which the pair is most interested in. These keywords are annotated to the domain ontology.

Our friend discovery protocol is guided by the Buddycast algorithm described in *Tribler* [11]. To discover a friendship link, each peer Pi, in our algorithm, periodically sends a Gossip message to its neighbors, and updates its current friends list, accordingly. When a peer Pj receives a Gossip message, i.e., a message containing the IP (Internet Protocol) address, port of the Pi and its list of interest, it retrieves the list of interest and compares it to its interest list. If the lists are similar, it adds it to its friends list, and replies with a friendship invitation. Otherwise, it replies with a message saying, 'we are not interested in the same topics'.

Peers are conceived such that no other contact with the same peer can be made for the next three months. That is, we suppose that a researcher does not frequently change its research topics.

Two threads are defined in Figure 5 and Figure 6. They describe the gossip behavior of each peer Pi. The first, called 'active behavior', is executed every Tgossip time

 ISBN: 978-1-61208-368-1

unit. It describes how Pi initiates a periodic gossip exchange. The second, called 'passive behavior', shows how it reacts to a gossip exchange initiated by some other peer.

Algorithm 1 *Gossip behavior of user u*

```
// active behavior, runs periodically every T time units
Let N(P) be the set of actual neighbors
Let F(P) be the set of actual friends.
Loop
(wait Tgossip)
  For all Pi  N(P) do
   NewPeers← neighbors.SelectRandom ()
      For all P NewPeers do
 if  P  F(P) then
          Connect with P
          GossipMsg = (myAddress, myInterestList)
          Send GossipMsg to P
           Receive GossipMsg from p
         endif
       end for
  endfor
end loop
```

Figure 5. Active behavior of a peer.

Algorithm 2 *Gossip behavior of pr*

```
//passive behavior Runs when contacted by some peer
Forever do
waitGossipMessage()
Receive GossipMsg from U
GossipResp = (myAdress, reponse)
Send GossipResp to U
```

Figure 6. Passive behavior of a peer.

b) Adaptive migration strategy

Adaptive migration strategy is a way of guiding an agent during its migration step. Our migration model is based on direct and indirect cooperation between agents. Direct cooperation takes place when the agents communicate with each other, whereas, indirect cooperation exploits the notion of mobile agent's trails. The idea of the indirect cooperation mechanism is based on the stigmergy theory. This ant colony-based theory was first developed by Grassé [19]. Explicitly, it uses the fact that ants do not communicate directly with each other.

Once a query is formulated in the proposed model, the created *SMA* dos not receive any predefined path; it builds its itinerary by itself as and when it travels over the network. To achieve this, we resort to the Ant Colony Optimization algorithm, in which an agent uses trails left by other agents on peers they have visited. Using this method, our approach enhanced the agent's autonomy.

- ***Migration algorithm***

The migration algorithm is used to show how the mobile agents would behave while searching for information in the network. It is described as follows.

- If Pi desires to make a search in the network, it first formulates a query, and then runs it locally to find out whether or not the sought-after data exist in the local database. In the affirmative, the search stops. Otherwise, the query is reformulated by the *AdA* in the domain ontology vocabulary, and transmitted to the *ArA* to check if the query has already been resolved. In the affirmative, the query is transmitted to the *CA* along with the address of peers that have answered. Only then, the *CA* creates and sends mobile agents to the peers concerned.
- Otherwise, a social mobile agent is created by the Pi. This peer is known as the *CP* of this particular mobile agent as well as all of its clones. The mobile agent is created by the *CA* of the *CP*. The *SMA* is given in the form of parameters; i.e., information about the *CP* (name, address and port), energy, which corresponds to the maximum number of peers that may be visited before it must return to its CP, and a cloning factor, used to determine how many times the mobile agent may be cloned at any peer. Note that the energy and the cloning factor parameters control the depth and breadth of the search. Therefore, they handle the maximum number of peers that may be visited and the friends list as well.
- The SMA clones itself as many times as the number of friends present in the list it received. This allows a mobile agent to be sent to each of these friends. Note that if a destination cannot be located, the SMA tries the next one in the list.
- Upon arrival at each peer, the *SMA* checks whether or not it or its replica has already visited this peer. The details of how agents exploit the trails are given in Section 5.2.c In the affirmative, the *SMA* stops its migration and comes back to the *CP*. Otherwise, it decrements its energy; i.e., E←E-1, updates its migration history and that of the Host Peer (*HP*), with information about the *CP*, and leaves a trail of its visit. It also updates itself with information about the current peer.
- Then, the *SMA* gives the query to the *CA* of the *HP*, which in turn transmits it to the *AdA* of the *HP*, to be reformulated in the local schema vocabulary. Only then, the *CA* runs it and search for the result in the *HP* database, to be transmitted to the *SMA*, which stores it and then moves to the next destination.
- When the mobile agents' energy has dissipated, and there are no more destination to visit, they then go back home.
- Once at home, the agent and its clones update the *CP* with the information they have so far collected throughout their journey and then dispose of themselves.
- During the course of its journey, the peer's machine may shut down while the *SMA* is still collecting

information. The agent has then the capacity to wait for the peer's return. In such a case, it goes to sleep and does not wake up until the machine is powered on again.

- ***Ant Colony Optimization***

Without trails, the agent chooses a random schema migration. In the proposed approach, a trail designates the on sight visit tracking of an agent, or its next destination. This may lead to less information being acquired about the network. However, it keeps the information up to date, and yet prevents peers that form cycles from having to deal repeatedly with mobile agents from the same *CP*.
Social Mobile Agents use trails as follows.

- If two SMA, from the same *CP*, arrive at the *HP* within a preset time period, the last agent arriving verifies its migration history.
  - If the latter is empty, therefore E=E-1. It means that the current *HP* is the agent's first destination. Then, it sends a message to its CP, and suicides.
  - Otherwise, it stops its migration and returns to the *CP* along with the information it has already collected.
- Upon arrival at a node, the *SMA* asks the archive agent of the actual *HP* whether or not a replica of itself has visited this place.
- It also asks the archive agent about the trails of agents that have visited that peer in the last T time, and verifies if there are trails of an agent that have been created by a friend peer.

If there is one then, the *SMA* clones itself and sends the clone to the destination of that agent (we suppose that a friend of friend is potentially a friend). Upon arrival, the clone compares its interest list to the host peer's. If there is a correspondence, it runs, and comes back to its *CP* with the results of the query and the information about a new friend it has made. Otherwise, it destroys itself.

- If an agent meets another agent on sight, they exchange their interest lists. If there is a correspondence, they exchange their *CP* addresses and their future destinations.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have tackled the problems of resource discovery and query routing in P2P data sharing systems. In doing this, we have first studied and analyzed the existing approaches and mechanisms found in the literature. Then, we have proposed a mechanism for resource discovery and query routing. For this purpose, we have used an agent-based architecture, a resource discovery and a query routing mechanism. This latter is two-fold; one relies upon a gossip protocol to build and manage the friends list, and the other on the ACO to optimize the mobile agent migration.

As we have shown throughout this paper, our contribution exhibits three originalities. The first one takes into account the two principal issues in data sharing in P2Pdatabase systems. That is, while other PDBMS treats just one issue, the proposed one handles both hiding the heterogeneity of data from different sources through domain ontology, and developing an efficient resource discovery mechanism. The second one has the advantage to be totally independent of the centralized management; hence, the peers are completely autonomous for both schema mediation and resource discovery. Finally, the third one is inherent to the resource discovery mechanism, which includes a social aspect using the friendship links. This reduces the randomness of the peers, leading to better search results. It also takes advantage of making use of mobile agents in P2P networks, as seen in Section 4.1, and optimizes their migration through ACO features. In a future work, we will focus on conducting more experiments on the proposed mechanism by means of the PeerSim simulator [20].

REFERENCES

[1] "Cisco Systems, Inc, Cisco Visual Networking Index: Forecast and Methodology, 2012-2016,"http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ ns537/ns705/ns827/white paper c11-481360.pdf.

[2] A. Halevy, Z. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. "The piazza peer data management system". IEEE Transactions on Knowledge and Data Engineering, 16(7):787–798, 2004.

[3] P. Adjiman, P. Chatalic, F. Goasdoué, M.C. Rousset, and L. Simon."Somewhere in the semantic web". In PPSWR 2005: International Workshop on Principles and Practice of Semantic Web Reasoning, 2005, pp. 1–16.

[4] W. S. Ng, "Adaptive p2p platform for data sharing", PhD Dissertation, National University of Singapore republic of Singapore, March, 2004.

[5] G. Karjoth, D. Lange, and M. Oshima, "a security model for aglets," IEEE Internet Computing, 1997, vol. 1, no. 4, pp. 68-77.

[6] H. Hamdi, F. Benchikha "An agent and ontology based approach for data mediation in P2P ", International Conference on Artificial Intelligence and Information Technology, Ouargla, Alegria, 2014.

[7] D. C. Faye "Semantic data mediation in SenPeer, A peer data management system", PhD Dissertation, Nantes University, 2007.

[8] M. R. A. King, " Locating data sources and query optimization in distributed peer-to-peer environment", PhD Dissertation, Toulouse III – Paul Sabatier University 2010.

[9] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. "The Bittorrent P2P_le-sharing system : Measurements and analysis". In 4th Int'l Workshop on Peer- to-Peer Systems IPTPS, vol. 3640, 2005, pp. 205-216.

[10] Gnutella http://rfc-gnutella.sourceforge.net/index.html, retrieved : June, 2014.

[11] J. A. Pouwelse, J. Yang, M. Meulpolder, D. H. J. Epema, and H. J. Sips "BuddyCast: an operational peer-to-peer epidemic protocol stack" in Delft University of Technology Parallel and Distributed Systems Report Series 2008.

[12] R. Akbarinia, E. Castanier, R. Coletta, F. Draidi, E. Pacitti,and D. Parigot "P2Prec: a social-based p2p recommendation system" in Proc: the 20th ACM Conference on Information and Knowledge Management, (CIKM2011), United Kingdom, 2011, pp. 2593-2596.

[13] R. Baraglia, M. Mordacchini, P. Dazzi, and L. Ricci "A P2P REcommender system based on Gossip Overlays (PREGO)" 2010 10th IEEE International Conference on Computer and

Information Technology (CIT 2010), West Yorkshire, UK, 2010, pp. 83 - 90.

[14] M. Dorigo and C. Blum. “Ant colony optimization theory: a survey”. Theoretical Computer Science, Elsevier Science Publishers Ltd., Essex, UK, 2005, vol. 344, n. 2-3, pp. 243-278.

[15] P. Cudre-Mauroux “Peer data management system” MIT Computer Science and Artificial Intelligence Laboratory Cambridge, MA, USA, 2007.

[16] A. Demers, D.Greene, C.Hauser, W.Irish, J.Larson, S.Shenker, and al. “Epidemic algorithms for replicated database maintenance”.In Sixth Symp. on Principles of Distributed Computing, 1987, pp. 1–12, New York, NY, USA. ACM Press.

[17] S. Voulgaris “Epidemic-based self-organization in peer-to-peer systems” PHD Dissertation, University of Amsterdam, 2006.

[18] J. Nair “An application for resource discovery in a peer to peer network using mobile agents” in International Journal of Emerging Technology and Advanced Engineering, March, 2012, vol. 2, Issue 3.

[19] P. P. Grassé. “The reconstruction of the nest and inter-individual coordination among Belicositermes natalensis and Cubitermes sp. the theory of stigmergy: Test interpretation of the behavior of termites manufacturers. Social insects”, 6 :41–80, 1959. 106.

[20] Official website of the PeerSim simulator, http://peersim.sourceforge.net/, retrieved: June, 2014.

 ISBN: 978-1-61208-368-1