International Journal of Advanced Information Technology (IJAIT) Vol. 1, No.6, December 2011

A QUERY LEARNING ROUTING APPROACH BASED ON SEMANTIC CLUSTERS

Taoufik Yeferny[1], Amel Bouzeghoub[2] and Khedija Arour[3]

[1]Department of Computer Science, Faculty of Sciences of Tunis, Tunisia
yeferny.taoufik@gmail.com
[2]Departement of Computer Science, Institute TELECOM, TELECOM & Management Sudparis, France
Amel.Bouzeghoub@it-sudparis.eu
[3]Departement Computer Science, National Institute of Applied Sciences and Technology of Tunis, Tunisia
Khedija.arour@issatm.rnu.tn

ABSTRACT

Peer-to-peer systems have recently a remarkable success in the social, academic, and commercial communities. A fundamental problem in Peer-to-Peer systems is how to efficiently locate appropriate peers to answer a specific query (Query Routing Problem). A lot of approaches have been carried out to enhance search result quality as well as to reduce network overhead. Recently, researches focus on methods based on query-oriented routing indices. These methods utilize the historical information of past queries and query hits to build a local knowledge base per peer, which represents the user's interests or profile. When a peer forwards a given query, it evaluates the query against its local knowledge base in order to select a set of relevant peers to whom the query will be routed. Usually, an insufficient number of relevant peers is selected from the current peer's local knowledge base thus a broadcast search is investigated which badly affects the approach efficiency. To tackle this problem, we introduce a novel method that clusters peers having similar interests. It exploits not only the current peer's knowledge base but also that of the others in the cluster to extract relevant peers. We implemented the proposed approach, and tested (i) its retrieval effectiveness in terms of recall and precision, (ii) its search cost in terms of messages traffic and visited peers number. Experimental results show that our approach improves the recall and precision metrics while reducing dramatically messages traffic.

KEYWORDS

P2P, Learning routing methods and Clustering

1. INTRODUCTION

Peer-to-peer systems have recently a remarkable success in social, academic, and commercial communities because they are more scalable, fault tolerant, autonomic and cost effective compared with centralized systems. In fact, peer-to-peer systems have become synonymous with file-sharing systems like Gnutella, Kazaa, etc. [1, 2] that have enjoyed explosive popularity over the last few years. These systems have been developed according to different distributed architectures which can be roughly classified as unstructured or structured. Within an unstructured P2P system, it is easier to construct the network and implement complex queries.

DOI : 10.5121/ijait.2011.1601

Nevertheless, it is very difficult to locate resources in this type of systems. In fact, the routing of messages or queries to locate desired resources still remains a thriving challenge. Query routing in current P2P systems is generally based on the following techniques: query flooding, random walk or heuristic [3]. All these methods generate a very large number of messages and cannot quickly locate the request resource. A lot of researches have been conducted to enhance search result quality and to reduce network overhead. Recently researches focus on query-oriented routing indices methods, which utilize the historical information of queries and query hits to route future queries. Indeed, the observation of the past information is used to create a knowledge base per peer that will be used to guide the process of peers' selection.

In these methods, when a peer p_i propagates a given query Q among computing peers, it evaluates Q against its local knowledge base B_i in order to select a set R_p of relevant peers to whom the query Q will be routed. If the number of relevant peers is below a certain threshold P_{max}, a randomly set of peers will be added from the neighbours table. However, these peers are chosen randomly, thus we are not sure if they are able to answer the query or to select from their knowledge bases relevant peers.

To improve the efficiency of the proposed methods, in this paper we introduce a novel approach that aims to minimize the number of messages passing through the network and maximize the ability of retrieving relevant data from the peer-to-peer network. Our approach organizes the P2P network into clusters of peers sharing similar knowledge. Indeed, each peer P_i in the network makes new connections to link peers sharing similar knowledge, named friend peers. When p_i selects from its local knowledge base an insufficient number of relevant peers it forwards the query according to its content to the best friend peers, which are able to select from their knowledge bases the relevant peers according to the query content.

The rest of the paper is organized as follows. In Section 2, we present a critical overview of query routing methods based on query historic. Section 3 introduces our approach. The experimental evaluation result of the proposed approach is described in section 4. Section 5 concludes and sketches avenues of future work.

2. OVERVIEW OF QUERY ROUTING IN P2P SYSTEMS

Efficient query routing in unstructured P2P requires intelligent decisions: selecting the best peers to, which a given query should be forwarded for retrieving additional search results. Query routing in current P2P systems is generally based on query flooding used in the Gnutella system [1]. Peers organize themselves into a random overlay. In order to find content, a peer sends a query to all its neighbours on the overlay, which, in turn, forward the query to all of their neighbors and so on, until the query time-to-live (*TTL*) expires. While this solution is straightforward and robust, it generates a very large number of messages and cannot quickly locate the request resource. Thus, performing such a task is greedy in bandwidth, which badly affects the system scalability. Several works tried to tackle the scalability problems inherent to Gnutella networks. Indeed, recent researches focus on methods based on query-oriented routing indices, which utilize the historical information of queries and query hits to route future queries. In the following, we briefly summarize some of these works.

In directed BFS [4], each node maintains some statistics of its neighbors such as the number of times previous queries can be answered through a neighbor node, the number of results obtained for the queries and the latency in receiving the results. When a peer propagates a given query it uses these statistics, in order to choose the appropriate neighbors. The main critic that can be addressed to this technique is that statistics maintained by each peer about its neighborhood is not wealthy enough. These statistics do not contain the information related to the content of the

query. To palliate this problem, Kalogeraki and al. [4] have presented a similar but more complex approach called intelligent search. In this method, each peer ranks its neighbors based on their relevance to the query and only routes the query to those neighbors that have high relevance. To implement this technique, a peer builds a profile for each neighbor. The profile contains the most recent queries processed by its neighbors along with the number of query hits. Furthermore, a peer performs an online ranking of its neighbors to choose the peers the query should be forwarded to.

In Route Learning [5], a peer tries to assess the neighbors that will most likely reply to queries. Peers compute this estimation based on knowledge that accumulates gradually from query and query hit messages sent to and received from neighbors. Route Learning inherits its basic idea from the classification problem, where a peer having n neighbors has n classes to choose from to forward a query. Each class corresponding to a neighbor i can be used to find out the probability of having there source reachable by neighbor i.

Self-Learning Query Routing (SLQR) [6] learns the interests of nodes and constructs friend relations. Relations can be automatically established based on interests' similarity between two users. In [6] each peer considers peers sharing same files with it as friend candidates. To do this, when any peer issues search request and gets results from some peers, it will send the search results to those whom returned successful results. By this process, a peer can guess who shares the same files with him. Thereafter, queries will be routed to friend peers. If the searches in friend peers fail, broadcast search will be investigated.

Learning Peers Selection (LPS) [7] learns the implicit behaviour of users that is deducted from query history. LPS supports a new knowledge or user profiles. A profile is a correlation between sent queries and positive peers or sent queries and query terms. When a peer forwards a query, it selects from the knowledge base relevant peers to whom the query will be routed. The knowledge base will be periodically updated in order to consider the most up to date information about new queries.

The idea underlying all the proposed methods is to replace the classical routing method (spread by flooding) used in Gnutella [1], by a semantic routing method based on the historical information about past queries. These information are used to build a knowledge base per peer in order to guide the process of peers' selection. These methods improve the efficiency of the query flooding approach. However, they have not addressed the unsuccessful relevant peers search problem. Indeed, when a peer selects an insufficient number of relevant peers from its local knowledge, it floods the query through the network.

At a glance, table 1 compares these different methods with respect to five criteria:

- *Scale Transit*: The scalability of a system is important for it to be useful in large scale environments. One measure of scalability is the number of messages that need to be routed in order to locate information. For systems that require transmitting a huge amount of messages (e.g., broadcast-based systems), the bandwidth consumption will be high, hampering by the way the system scalability. Routing method can be scalable (i.e. yes) or not (i.e. no).
- *Storage Space*: Each peer may need to incur some storage space for maintaining metadata that are used in directing the search space. Clearly, storing more metadata also implies that it is more costly to keep these data up-to-date. Routing method may need large storage space (i.e. high) or not (i.e. low).

- *Knowledge Source*: A peer needs to collect information in order to build its local knowledge base. In the surveyed approaches peers may collect information about only their neighbors or about any peer in the network that replies to theses queries.
- *Sharing Knowledge*: Peer shares information about its local knowledge base content with other peers in the P2P network (i.e. yes) or not (i.e. no).
- *Peer Selection*: Peer selection process can be totally semantic, partially semantic (both random and semantic).

Table 1. Comparative study of routing methods.

Criteria\Methods	**Directed BFS**	**Intelligent Search**	**Self-Learning Query Routing**	**Route Learning**	**LPS**
Scale Transit	Yes	Yes	Yes	Yes	Yes
Storage Space	Low	Low	Low	High	Low
Knowledge Source	Neighbors	Neighbors	Any peer	Neighbors	Any peer
Sharing Knowledge	No	No	No	No	No
Peer Selection	Partially semantic	Partially semantic	Partially semantic	Partially semantic	Partially semantic

To summarize, we can say that all the surveyed approaches are effective in term of scale transit. Furthermore, we note that Route Learning stores more meta-data which implies an important cost to update. In addition, knowledge bases in Directed BFS, Intelligent Search and Route Learning include only information about neighbors that reply to past queries. However, an important number of positive peers (peers whose reply to past queries) is not considered. On the other hand, these methods build the knowledge gradually from query and query hit messages sent to and received from neighbors. Therefore, they are not able to route future queries directly to relevant peers but they try to estimate the neighbors that will most likely reply to queries.

Nevertheless, it is worth of mention that all proposed approach are based on both semantic and random peer selection algorithm, which badly affects their performance. In addition, they don't share their knowledge, thus each peer in the network has only information about its local knowledge base content. We propose a new approach to improve the two last criteria. The selection of relevant peers in our approach is based only on semantic algorithms. Indeed, when a peer selects from its local knowledge base an insufficient number of relevant peers it forwards the query according to its content to the best friend peers.

3. PROPOSED APPROACH

The common idea of routing methods based on queries historic is to exploit knowledge from past queries in order to select peers which are most likely to provide an answer for a forthcoming query. Thus, when a peer p_i propagates a given query Q it evaluates it against its local knowledge base B_i in order to select a set R_p of relevant peers to whom the query Q will be routed. If the number of the relevant peers is below a certain threshold P_{max}, a randomly set of peers will be added from the neighbors table. These methods can be represented by a generic algorithm named *QueryRouting()* (see Algorithm 1).

The differences between all the proposed approaches in the literature are the knowledge representation (i.e. ontology, simple matrix, etc.), and the *EvaluateQuery()* function, which are specific for each approach. We propose to improve these approaches by replacing the

addRandom() procedure by another one named *addSemantic()*, which adds a list of peers having knowledge closed to that of the current peer p_i, named friend peers. Indeed, each peer in the network needs to make two types of connections, namely neighbors and friends. The role of friends connections is to link peers sharing similar knowledge. Therefore, we must perform peer clustering at the level of overlaying network topology. In the following, we present the two phases of our approach namely peer clustering strategy (3.1) and friend peers' selection process (3.2).

Algorithm 3: QUERY ROUTING ALGORITHM

```
1  Algorithm: QueryRouting (B, Q, F_r, P_max)
2  Input :
3      B : knowledge base.
4      Q: Query to forward.
5      F_r : Friend list.
6      P_max : The maximum number of peers to be selected.
7  begin
       finalList := ∅                          ▷ List of peers to forward the query to.
       Rp := LearningPeerSelection(B, Q, P_max) ▷ Evaluates the query Q against the know-
                                                  ledge base B to extract a set Rp of relevant peers.
8      finalList := Rp
9      if (|Rp| < P_max) then
10         N := P_max − |Rp|                   ▷ Determines the number of
11                                               peers to be added.
12         addSemantic(finalList, N, F_r, Q)   ▷ Adds N best friend peers
13                                               from friend list F_r to finalList.
14     Forward(Q, finalList);                  ▷ Forwords the query to peers
15                                               in finalList.
16 end
```

3.1 Peer Clustering

Before describing our peer clustering method, we present the following notations (see Table 2) and definitions.

Table 2. Definition of Terms.

P	*A set of peers that composed the Peer-to-Peer network*
$p_i \in P$	*A peer in the Peer-to-Peer network*
Fr_i	*The set of p_i friends*
B_i	*A local knowledge base of p_i. For the sake of generality, the knowledge base can be modelled formally as a triplet B (E_i, F_i, I) where E_i is a set of past queries, F_i a set of peers which answered to queries in E_i and I is a semantic relation between E_i and F_i.*
R_i	*A Set of representative vectors of the peer p_i characterizing its local knowledge base B_i.*

Definition 1 Representative Vectors R_i: Each peer $p_i \in P$ selects a representative vectors set R_i to describe its knowledge base content. We define, the cluster centroid of a specific past queries set belonging to the p_i' knowledge base as a representative vector $r_{i_j} \in R_i$.

Definition 2 $Distance(r_{i_k}, r_{j_v})$ is the distance measure between $r_{i_k} \in R_i$ and $r_{j_v} \in R_j$, in other sense, the similarity between particular cluster belonging to two different knowledge bases B_i and B_j. We used the Euclidean distance between centroid of two clusters represented by r_{i_k} and r_{j_v}. Let's, $r_{i_k} \in R_i$ and $r_{j_v} \in R_j$ tow representative vectors, the Euclidean distance is defined as follows:

$$Distance(r_{i_k}, r_{j_v}) = \sqrt{\sum (x_i - y_i)^2} \qquad (1)$$

Where x_i, y_i are the i^{th} components of r_{i_k} and r_{j_v}. Remember that each i^{th} components is the weight of the i^{th} term in the vector.

Definition 3 $Similarity(Q, r_{i_k})$ is the similarity measure between a representative vector $r_{i_k} \in R_i$ and a given query Q. The similarity between r_{i_k} and Q is characterized by the common terms. More formally, this similarity is defined as follows:

$$Similarity(Q, r_{i_k}) = \frac{|Q \cap r_{i_k}|}{|Q \cup r_{i_k}|} \qquad (2)$$

Based on the above definitions, we introduce our peer clustering algorithm. It consists of three steps:

1. ***Computing representative vector***: Every peer p_i in the network processes its knowledge base B_i in order to extract a set of representative vectors R_i. Each vector $r_{i_k} \in R_i$ is a cluster centroid of past queries E_i belonging to the base B_i. We have used k-means clustering algorithm implemented in Weka [8] platform to cluster the past queries E_i. This task is executed when a peer builds or updates its knowledge base in order to update the representative vector accordingly. In the following, for simplicity of the presentation, we assume that each knowledge base B_i is represented by one vector r_{i_1}, thus $R_i = \{r_{i_1}\}$.
2. ***Searching for friend peers***: After computing its representative vector r_{i_1}, the peer p_i floods r_{i_1} within a certain Time To Live (*TTL*). It sends a query, named *searchFriends*(r_{i_1}, *TTL)*, similar to that of ping-pong messages in Gnutella, to search for peers having similar interests. When a peer p_j receives this query it computes the distance *Distance* (r_{i_1}, r_{j_1}) and answers to the query by sending its representative vector r_{j_1} and the distance value.
3. ***Selecting friend peers***: When p_i receives the representative vectors of other peers it selects the best k peers having minimum distance. These peers form the set Fr_i of p_i' friends.

Each peer in the network runs the peer clustering algorithm when it updates its knowledge base. Hence, peers are organized dynamically in new semantic clusters. This task is periodically executed offline.

3.2 Query Routing Algorithm Based on a Clustered Network

Assuming that the P2P network is clustered, thus each peer p_i makes Friends connections to link peers having similar knowledge. To improve the routing efficiency of existing approaches that use the *addRandom()* procedure in Algorithm 1, we propose to replace this procedure by another one *addSemantic()* that adds a list of friend peers having similar knowledge. In this way, if the number of relevant peers selected from the local knowledge base is insufficient, the query is routed selectively according to its content to the best friend peers, where we are sure that it is able to select promising peers.

We introduce the *addSemantic()* procedure in algorithm 2. It involves three steps:

1. Compute the similarity between the representative vector of each friends peers and the query *Q*.
2. Sort the list *Fr* of friend peers according to the similarity value.
3. Add to *finalList* the best *N* friends having the highest similarity values.

Algorithm 2: ADD SEMANTIC PROCEDURE

```
1  addSemantic(finalList, N, Fr, Q)
2  Input:
3      finalList : list of peers to forward the query to.
4      N : Number of peers to be added.
5      Fr : Set of friends peers.
6      Q: Query.
7  begin
8      addedPeers = 0;
9      for freind ∈ Fr do
10         r := freind.getRepresentativeVecteur();
11         freind.similarity := similarity(Q, r);
12     sort(Fr);
13     while freind ∈ Fr and addedPeers < N do
14         finalList := finalList ⋃ freind;
15         addedPeers++
16 end
```

4. EXPERIMENTS

Our approach presents a solution for routing effectiveness of any learning routing approach. To test the efficiency of our approach, we used the Learning Peer Selection algorithm proposed in [7]. We defined two versions of this algorithm:

- *LPS*: Learning Peer Selection without clustered network (peers are still randomly connected).
- *LPSCN*: Learning Peer Selection over Clustered Network (The network is organized into clusters of peers with similar interests).

To evaluate the performance of our approach we extend a peer-to-peer simulator PeerSim [9]. In the following, we present the environment and the performance evaluation.

4.1 Environment

We have chosen the PeerSim simulator [9], which is an open source java tools. The structure of this simulator is based on components and makes it easy to quickly prototype a P2P protocol, combining different plugin building blocks that are in fact Java objects. It supports two simulation models: the cycle-based model and the event-based model. In our case, we used the cycle-based model.

4.2 Data source characteristics

The data set used in our experiments is the "Big Dataset", developed under the RARE project [10]. This data set was obtained from a statistical analysis on Gnutella system data [11] and from the TREC collection [12], which allow us to simulate our algorithm in real conditions. Big Dataset is composed of *25000* documents and *5000* queries.

To distribute documents and queries among the set of peers we used the Benchmarking Framework for P2PIR [13]. This Framework is configurable, which allows user to choose certain parameters (i.e. number of peers, replication rate of queries, etc.) and provides XML files describing the nodes, the associated documents and the queries to be launched on the network. In our case, we have chosen a number of peers equal to *810* and a query replication rate equal to *6*.

4.3 Evaluation measures

To test the quality of our approach, we used the *Recall (R), Precision (P)* metrics [14] performance measures, the number of visited peers (*VP*) and the messages traffic (*MT*), defined as follows for a given query *q*:

$$R(q) = \frac{RRD}{RLD} \qquad (3)$$

$$P(q)@k = \frac{RRD@k}{RTD} \qquad (4)$$

Where, *RRD* denotes the number of relevant retrieved documents, *RLD* is the number of relevant documents, *RRD@K* is the number of relevant retrieved documents in the first *k* rank positions, in our case we fixed k to *3* and *RTD* denotes the number of retrieved documents.

4.4 Simulation setup

The simulation is based on the following parameters:

- *TTL*: The maximum number of hops that a query is allowed to travel (the horizon of the query), initialized to *5*.
- *Pmax*: The maximum number of peers to be selected for a query, initialized to *3*.
- *Overlay size*: Number of peers in the network, initialized to *810* (number of peers in the Dataset).

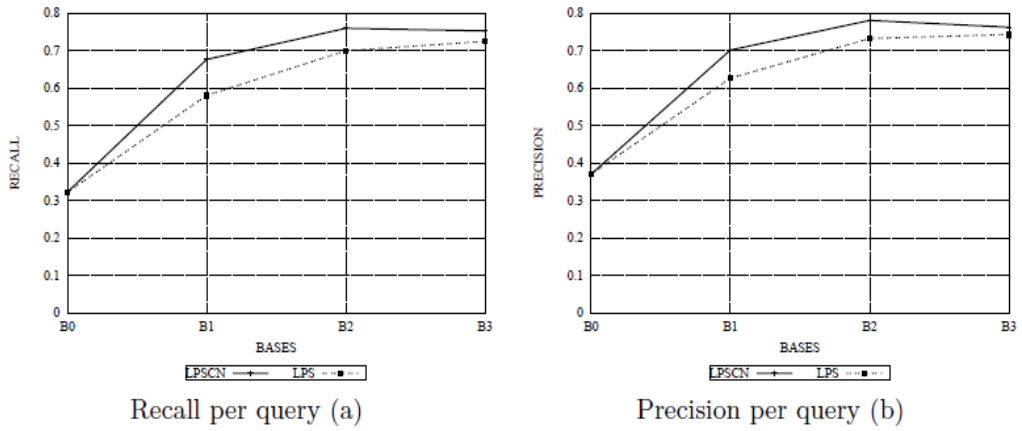
LPS and *LPSCN* start with an empty Knowledge base B_0, so they use flooding routing techniques like the Gnutella System [1]. Thereafter, the knowledge bases are periodically updated in order to take new information about new queries. During the simulation task, the knowledge base for each peer has been updated three times respectively after *9000*, *1800* and *27000* queries, for building B_1, B_2 and B_3 bases.

4.5 Results

4.5.1 Comparison of retrieval effectiveness for *LPS* and *LPSCN*

To test the retrieval effectiveness of our approach, we compute the average recall and precision by interval of *9000* queries (number of queries used to update knowledge bases) sent from different peers.

Figure 1 illustrates the retrieval effectiveness for *LPS* and *LPSCN*. At the beginning, the two algorithms start with flooding routing method like the Gnutella system [1], thus the average recall and precision values are low (around 0.32). As expected, the recall and precision for *LPS* and *LPSCN* increase after each update of knowledge bases. In addition, *LPSCN* is more effective than *LPS* by using different knowledge bases. Figure 1 (a), shows that *LPSCN* gives recall between *0.67* and *0.75*, while the recall for *LPS* varies between *0.57* and *0.72*. Furthermore, Figure 1 (b) shows that precision for *LPSCN* varies between *0.70* and *0.76,* however it varies for *LPS* between *0.62* and *0.74*.



Recall per query (a)

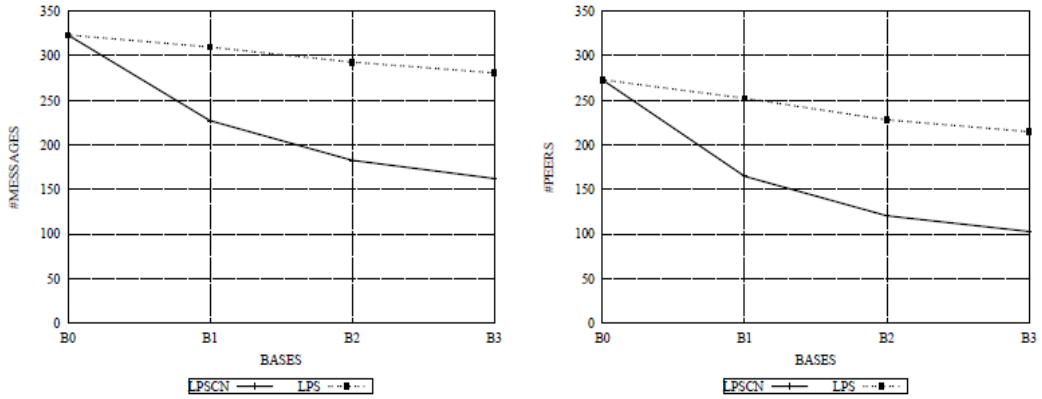
Precision per query (b)

Figure 1. Retrieval effectiveness for *LPS* and *LPSCN*.

4.5.2 Comparison of search cost for *LPS* and *LPSCN*

To test the search cost effectiveness of our approach, we compute the average message number and average visited peers number by interval of *9000* queries sent from different peers for *LPS* and *LPSCN*. Figure 2 (a) shows that our approach reduces message traffic. Indeed, the average number of messages for *LPCN* has decreased from *323* by using the initial base B_0 to *227* by using *B1* then *182* by using *B2* and *162* by using *B3*. However, the average number of messages for *LPS* has decreased from *323* by using an empty initial base *B0* to 309 by using *B1* then *292* by using *B2* and *280* by using *B3*. The average message number value for *LPSCN* and LPS are respectively *190* and *294*. Our approach demonstrated the best search cost performance overall achieving up to *35%* less messages traffic than *LPS*.

Figure 2 (b) shows that the average visited peers number for *LPSCN* has decreased from *234* by using the initial base *B0* to *273* by using *B1* to *164* by using *B2* and *120* by using B3. However, the average number of visited peers for LPS has decreased from 273 by using an empty initial base *B0* to *252* by using *B1* to *228* by using *B2* and *214* by using *B3*. It is worth of mention that our approach overall achieving up to *45%* less visited peers than *LPS*.



\

Figure 2. Search cost for LPS and LPSCN.

These results show that learning routing methods can achieve better results when the network is organized into clusters of peers with similar interests, which prove the effectiveness of our approach.

5. CONCLUSION AND FUTURE WORKS

Many learning routing methods have been proposed in the literature. The idea underlying these methods is to accumulate information about past queries and queries results in order to build a knowledge base per peer, which will be used to guide the peer selection process for the future queries. In these methods, each peer in the network uses only its local knowledge base to select relevant peers. Usually, an insufficient number of relevant peers are selected from the current peer's local knowledge base which badly affects the approach efficiency. To palliate such drawback, we introduced a novel method that clusters peers sharing similar interests. It exploits not only the current peer's knowledge base but also that of the others in the cluster to extract relevant peers. Indeed, we defined a peer clustering strategy and a semantic query routing algorithm. The experimental results prove the retrieval effectiveness and the search cost of our approach. As future work, we plan to evaluate the proposed method with other semantic algorithms and benchmarks.

REFERENCES

[1] Gnutella, "Gnutella Web site," http://www.gnutella.com/

[2] A. Krekelberg and N. S. Good, "Usability and privacy:a study of kazaa p2p file-sharing," in Proceedings of the SIGCHI conference on Human factors in computing systems. ACM Press, pp. 137–144, 2003.

[3] H. Jin, X. Ning, H. Chen, and Z. Yin, "Efficient query routing for information retrieval in semantic overlays," in Proceedings of the 21st Annual ACM Symposium on Applied Computing. ACM Press, pp. 23–27, 2006.

[4] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti, "A local search mechanism for peer-to-peer networks," 11th International Conference on Information and Knowledge Management (CIKM'2002), 2002.

[5] S. Ciraci, I. Krpeoglu, and zgr Ulusoy, "Reducing query overhead through route learning in unstructured peer-to-peer network," Journal of Network and Computer Applications, vol. 32, no. 3, pp. 550 – 567, 2009.

[6] Chen H., Gong Z. y Huang Z., "Self-learning Routing in Unstructured P2P Network", International Journal of Information Technology, vol. 11, No. 12, pp. 59~67, 2005.

[7] T. Yeferny and K. Arour, "Learningpeerselection: A query routing approach for information retrieval in p2p systems," International Conference on Internet and Web Applications and Services, vol. 0, pp. 235–241, 2010.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, "The weka data mining software: an update". SIGKDD Explor. Newsl.

[9] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris, "The Peersim simulator," http://peersim.sf.net, 2011.

[10] RARE: Le projet RARE (Routage optimise par Apprentissage de REquêtes). http://www-inf.it-sudparis.eu (20/Mars, 2010)

[11] S. Goh, P. Kalnis, S. Bakirs, K.L. Tan "Real datasets for _le-sharing peer-to-peer systems," Database Systems for Advanced Applications, 10th International Conference.

[12] TREC: Text REtrival Conference. http://trec.nist.gov (20/Mars, 2010)

[13] S. Zammali, K. Arour.: P2PIRB: Benchmarking framework for p2pir. In: Third International Conference on Data Management in Grid and Peer-to-Peer Systmes,Bilbao, Spain. (2010) 100{111[14] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel, "Performance measures for information extraction", in Proceedings of DARPA Broadcast News Workshop, pp. 249–252, 1999.

Authors

Taoufik Yeferny received the master degree from Dept. of Computer Science, Faculty of Sciences of Tunis, Tunisia in 2009. Currently he is Phd student in Faculty of Sciences of Tunis, Tunisia. His research interest includes routing in P2P systems. He is a member of **MOSIC** and **URPAH** Tunis.