An Efficient Learning Query Routing Method for Unstructured P2P Systems

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

[1]*Dept. of Computer Science Faculty of Sciences of Tunis, Tunisia MOSIC Research Group*
[2]*Dept. of Computer Science National Institute of Applied Sciences and Technology of Tunis, Tunisia URPAH Research Group Khedija.arour@issatm.rnu.tn,*
[3]*Dept. of Computer Science Telecom SudParis, France SAMOVAR CNRS LAB Taoufik.yeferny@it-sudparis.eu, Khedija.arour@issatm.rnu.tn, Amel.Bouzeghoub@it-sudparis.eu*

Abstract

In unstructured P2P systems, peers organize themselves into a random overlay. A challenging problem in these systems is to efficiently locate appropriate peers to answer a specific query. This paper proposes a semantic method in , which a query can be routed for appropriate peers instead of broadcasting or using random selection. This semantic is generally built from the contents of the peers, but can also bring in the implicit behavior of the users. The main objective of our method is to achieve better results in non-supervised tasks through the incorporation of usage data obtained from past search queries. This type of method allows us to discover the motivations of users when visiting a certain documents and peers. The terms used in past queries can provide a better choice of features queries. Hence, for each peer, our method learns from past queries to represent correlation between sent queries terms and related peers. We implemented the proposed method, and compared its routing effectiveness in terms of both recall and messages traffic with a broadcasting scheme (without learning). Experimental results show that our method is efficient and performs better than other non-semantic query routing methods with respect to accuracy. In addition, our approach improves the recall rate nearly 90% while reducing message traffic dramatically compared with Gnutella protocol.

Keywords: *WP2P, Learning routing methods, User profile.*

1. Introduction

In recent years, P2P systems are becoming very popular since they offer users the possibility to share and access to various resources distributed in large scale systems. They are more scalable, fault tolerant, autonomic and cost effective compared with centralized systems [1]. These systems have been developed according to different distributed architectures which can be roughly classified as unstructured or structured. A structured system [2, 3, 4] maintains a well-defined structure among participating peers and places objects based on the logical identifiers calculated by a predefined function. The search mechanism for an object in the structured P2P system is very simple and efficient. However, it's very difficult to dynamically maintain the structure in such as environments [4]. On the other hand, within an unstructured P2P system [5], there is no fixed topology for peers. Thus, each peer typically stores its own data objects and self maintains a set of links to other peers (called neighbors peers) so that the backbone of the system is formed. In contrast of a

structured P2P system, it is easier to construct the network and implement complex applications in unstructured P2P systems. Nevertheless, it is very difficult to locate appropriate resources in this type of system. In fact, the routing of messages or queries to locate desired resources still remains an exciting challenge. Indeed, efficient query routing requires smart decisions: selecting the best peers to which a given query should be forwarded for retrieving the best search results.

Query routing in current unstructured P2P systems is generally based on the following techniques: query flooding, random walk or heuristic [6]. All these methods generate a very large number of messages and cannot quickly locate the request resources. Thus, performing such as a task is greedy in bandwidth, which badly affects the system scalability. The efficiency of the researches in an unstructured P2P can be improved by introducing some semantic aspects into the process of query routing. Several works have attempted to improve the classical query routing techniques in unstructured systems by introducing semantics in the process of query propagation [7, 8]. The proposed semantic methods can be classified into content-oriented routing indices methods [9, 10] or query-oriented ones [11].

Content-oriented routing indices methods use meta-data extracted from the shared content of each peer to build a global index. This global index gives for each peer, in the network, a partial and approximate view of the network content so peers will be able to route efficiently their queries. Such methods improve the retrieval effectiveness; however they need a very large number of messages to build the global index.

Query-oriented routing indices methods exploit the historical information of past 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 represents the user's interests or profile. When a peer propagates a given query among computing peers, it evaluates it against its local knowledge base in order to select a set of relevant peers to whom the query will be routed. Thus, these methods are more advantageous than content-oriented ones, since no excessive network overhead is necessary for building the routing indices.

Recently researches focus on query-oriented routing indices methods [10, 12]. Few query-oriented routing indices methods have been proposed in the literature. They improve the search efficiency and effectiveness [13, 14] of traditional routing approach. However, in the existing methods the user profile is represented by some statistics about past queries (*e.g.*, query keywords, hits number per peer, *etc.*) but they do not exploit repetition rate for keywords seen in sent queries and relationships between them as well [15]. In this paper, we introduce a novel learning routing method that exploits repetition rate for keywords seen in sent queries and relationships between them as well in order to build a knowledge base per peer that represents the user's profile. In our method, the user's profile is a correlation between sent queries and positive peers or sent queries and query terms.

The rest of the paper is organized as follows. In Section 2, we present an overview of query routing methods in P2P systems. Section 3 provides a critical study of the overview. Section 4 defines our query routing method. In Section 5, we report the results of our experimental evaluation of the proposed strategy. Section 6 concludes, discusses and sketches avenues of future work.

2. Overview of query routing in P2P systems

The ideal solution for query routing in P2P systems is to send automatically the query to a set of peers that can provide an answer. Peer-to-Peer search requires intelligent decisions for query routing: selecting the best peers to forward a given query for retrieving the best query results. In literature, several research works have tried to improve the traditional query

routing method, which propagate a query to a random set of peers, by introducing semantics in the process of query propagation [8, 9]. To define this semantic, a first approach is based on content of peers. The content of each peer can be summarized in a *superdocument*. Hence, a router forwards the queries based on meta-information stored in this *superdocument*. Another approach is based on queries history. We can cite the CORI (Collection Retrieval Inference Network) and gGLOSS (generalized Glossary of Server's Server) as approaches which are based on peer's content. They represent the collection of each neighbor in a *superdocument*. The set of all *superdocuments* forms a special purpose collection that is used to identify the most promising collections for a given query.

The information retrieval system PlanetP [15] represents the content of each peer in the network in a compactly Bloom filter [15]. These Bloom filters are distributed across the network using a Gossiping algorithm [15]. The set of all Bloom filters forms a global index to give the peer a partial and approximate view of the network content.

Receiving a query, firstly a peer searches in its local index. If it is not possible to answer this query, it calculates a score of peers from the global index and propagates the query to the peers which have the greatest score.

Few methods use historical queries information. Nevertheless, REMINDIN [8] scheme exploits social metaphors to define a strategy of query routing. In this scheme, each peer maintains a set of RDF(S) statements in a local node repository (ontology). A statement may describe either data, *e.g.*, (TBL isAuthorOf WeavingTheWeb), or conceptual information, *e.g.*, (University subClassOf Organization). Furthermore, it stores meta-information about these statements in order to memorize where the statement came from and how much resource-specific confidence and overall confidence is put into these statements and peers, respectively. To select promising peers for a given query, a peer evaluates the query against the local node repository in order to select a set of statements matching the query. For each statement REMINDIN retrieves its meta-data, which comprise resource specific confidence values for each peer's knowledge about the particular statement and overall confidence values for each peer. Thereafter, promising peers are sorted according to their strength. Up to a specified threshold, best peers are returned as targets for the query. In directed BFS [14] 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. From these statistics, any peer can select the best neighbor to send the query. One disadvantage of this technique is that the statistics stored by a peer on its neighbors are too simple. These statistics do not contain the information related to the content of query. To alleviate this problem, Kalogeraki, *et al.*, [14] have presented a similar but more complex approach called intelligent search. In this approach, 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, peer performs an online ranking of its neighbors to choose the peers to forward the query.

In Route Learning [16], a peer tries to estimate the neighbors that will most likely reply to queries. Peers calculate 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 the source that is reachable by neighbor *i*.

Self-Learning Query Routing [16] learns the interests of nodes and constructs friend relations. Relations can be established automatically based on interest’s similarity between

two users. In this protocol, each peer views peers sharing same files with it, as friend candidates. To do this, when any peer issue search requests and gets results from some peers, it will send the search results to those who return successful results. By this process, the peers can get to know who is sharing the same files with them. Thereafter, queries will be routed to friend peers, if the searches in friend peers fail, broadcast search will be executed.

3. Synthesis on routing methods

Methods based on peer's content can provide important results but they have many problems related to the large scale of P2P systems. They incur either a higher storage cost since more indices need to be stored at a peer and a huge amount number of messages need to be exchanged to build and refresh these indices, which badly affect the system scalability. Furthermore, due to the network dynamicity, the indices may be obsolete or inconsistent. On the other hand, the proposed query routing methods based on queries history (for example Intelligent Search, Route Learning, etc) build a knowledge base per peer deduced from past queries and related query hits. However, in the existing methods the user profile is represented by some statistics about past queries (*e.g.*, query keywords, hits number per peer, *etc.*) but they do not exploit repetition rate for keywords seen in sent queries and relationships between them as well. In this paper, we propose a routing method that supports a new knowledge or user profiles. A profile is a correlation between sent queries and positive peers or sent queries and query terms. The originality of our method is that the computation is carried out in a completely asynchronous manner without any central knowledge or coordinating instance. In our method, a peer tries to estimate the peers that will most likely reply to queries. Peers computes this estimation based on knowledge accumulate from past queries and related query hits.

4. Proposed method

The idea underlying our proposal is to replace the classical routing method (spread by flooding) used in Gnutella [17], by a semantic routing method based on a set of profiles. The method is based upon the similarity between the past queries and the query to be routed. The objective of our approach is to have a low cost but effective routing approach in unstructured peer-to-peer networks.

4.1. Global Architecture

We proposed a method that analyzes queries collected on the P2P system. In fact, knowledge of user search patterns can be used to improve search performance. The key contribution is the observation of the past information, which can be used to create a sample knowledge database by peer to guide the process of peers' selection. In contrast, in Gnutella system, a query is first evaluated on the querying peer, then propagated recursively in a subset of neighbors chosen by a randomly manner (*i.e.*, flooding process). Research is terminated when cycles or a maximum number of hops, which a query is allowed to travel (called TTL or Time To Live), are checked. In order to improve the effectiveness of this method, we replaced queries propagation module by another module using knowledge about past queries in order to route the queries to promising peers. We have also added two other modules:

- A management profiles module that runs periodically. It builds the knowledge base from a log file, which store information about sent queries.

- A management log file module that runs when receiving responses.

Figure 1 illustrates the global architecture of our approach.

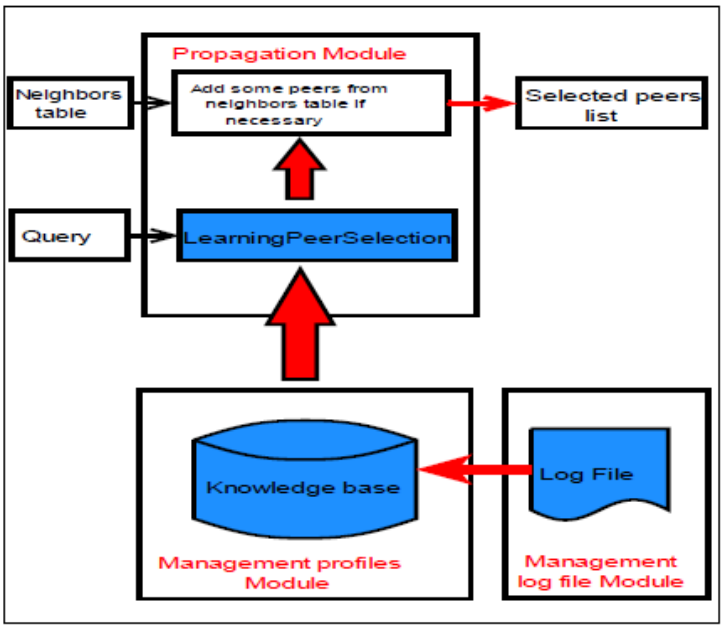

Figure 1. Global Architecture of Our Approach

4.2. Management log file module

Each peer stores information about past queries in a log file. There will be no global knowledge shared between all the peers but each peer will also have a list of data collected from the answered queries and store it in local file. When a peer receives responses for a query, this module updates the log file by adding information related to this query like the identifier of the query, these terms, the downloaded documents and associated peers

4.3. Management profiles Module

4.3.1. Construction of the knowledge base: The objective of this module is to generate a set of profiles that represent semantic relationships between sent queries and positive peers (*i.e.*, peers from, which there have been documents downloads). To build these profiles, we used a formal approach based on Formal Concepts Analysis [18].

In our case, we use two contexts, which are extracted from the log file. The first context represents the relation between sent queries and its terms, called *C1*. The second one represents relation between sent queries and positive peers, called *C2*. In fact, the attributes of context *C1*, respectively *C2*, are the queries terms and peers, which answer to these queries. An algorithm of formal concepts generation is then applied to generate two sets of concepts, noted *E1* and *E2*. We have used Godin algorithm [19] implemented in Galicia *V3* platform. The concepts of E1, respectively *E2*, will be under the following form *({R1,...,Rm},{T1,...,Tn})*, respectively, *({R1,...,Rp},{P1,...,Pk})*, which *Ri* is a set of queries identifiers, Ti is a set of terms and Pi is a set of peers. These sets form a knowledge base *B(E1, E2)* used by selection peers algorithm in order to find the most relevant peers. Figure 2 represents the different steps for building formal concepts.

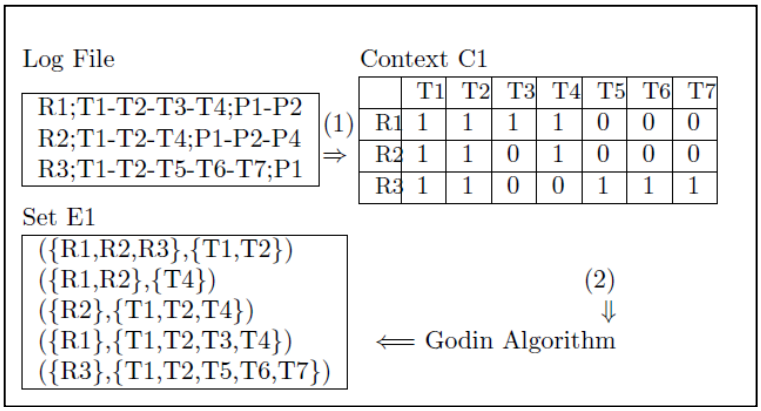

Figure 2. The different steps for building formal concepts

4.3.2. Updating the knowledge base: The knowledge bases are updated periodically for taking information about the new queries. To do this, we have defined two maintenance strategies (methods) of these bases:

- Static Strategy: consists to generate the knowledge base for each peer from the history of all sent queries.
- Incremental Strategy: consists to generate a temporary base B+ from the history of new queries: Queries issued after the last update operation. Thereafter we build the new knowledge base by increasing the old base with *B+* (New base=old base union B+).

4.4. Queries spreading module

When a peer receives a query, this module call LPS (LearningPeerSelection) algorithm for selecting a set of "relevant" peers to the query. If the number of selected peers is below a certain threshold we add a randomly set of peers from the neighbors table (procedure addRandom () in Algorithm 1).

Algorithm 1: PROPAGATION ALGORITHM

```
1  Algorithm: PROPAGATION (B, Q, Lv, Pmax)
2  Input :
3      B : knowledge base .
4      Q: Query to forward.
5      Lv : Neighbors list.
6      Pmax : The maximum number of peers to be selected for query.
7  begin
8      finalList := ∅//List of peers to forward the query to
9      selectedPeers :=LearningPeerSelection(B, Q, Pmax);
10     finalList := selectedPeers;
11     if (|selectedPeers| < Pmax) then
12         N := Pmax − |selectedPeers|;
13         addRandom(Lv, finalList, N);
14     Forward(finalList);
15 end
```

4.3.2. LearningPeersSelection algorithm LPS: The objective of this algorithm is to generate a set of similar concepts to a given query based on the following formula. Then it generates, from these concepts, the set of promising peers to whom the query will be routed.

Similarity computing

Let *k = (O, P, R)* a formal context. The similarity between two objects *a* and *b* of *O* is characterized by the common properties of this objects. More formally, this similarity defined as follows:

$$\mathrm{Sim}(P_a,P_b)=\frac{|P_a \cap P_b|}{|P_a \cup P_b|}$$

Where, P_a and P_b the properties sets of *a* and *b*.

Presentation of the LPS algorithm

To choose the relevant peers LPS is based on the knowledge base *B (E1, E2)* generated by the management profiles module. In fact, for a given query *Q*, the algorithm extract from *E1* the very similar concept *C* to *Q* (Concept, which have the higher similarity value with *Q: getConcept(B.E1,Q)* function in algorithm 2). The similarity between a concept *C* in *E1* and a query *Q* is calculated as follows:

$$Sim(Q, Int(C)) = \frac{|Q \cap Int(C)|}{|Q \cup Int(C)|}$$

Where:
Int(C) represents the intent of *C* and *Q* the list of query terms.
Thereafter, we determine from *E2*, a set of similar concepts to *C* (*SQPC: SimilarQueriesPeersConcepts*) containing the best promising peers for *Q*. In this case, the similarity between a concept *C* and a concept *Cj* in *E2* is calculated as follows:

$$Sim\big(Ext(C_i), Ext(C_j)\big) = \frac{|Ext(C_i) \cap Ext(C_j)|}{|Ext(C_i) \cup Ext(C_j)|}$$

Where:

Ext(C) represents the extent of the concept *C*.

Finally, selected peers is determinate form the intent of each concept in *SQPC* (*getSelectedPeers(SQPC)* function in algorithm 2). If the number of selected peers is lower than a threshold *Pmax* we extract another concept *C* similar to *Q* from *E1* and repeat the same steps.

Algorithm 2: PEER SELECTION ALGORITHM

Algorithm: LEARNINGPEERSELECTION(B, Q, P_{max})
Input :
    B : Knowledge base;
    Q : Query;
    P_{max} : The maximum number of peers to be selected for query.;
Output:
    $selectedPeers$: List of selected peers
begin
    $selectedPeers = \emptyset$
    concept=getConcept(B.E1,Q)
    while $concept$ *and* $|selectedPeers| < P_{max}$ **do**
        $SQPC := getSimilarConcept(B.E2, Extent(concept))$
        $selectedPeers := selectedPeers \cup getSelectedPeers(SQPC)$
        $B.E1 := B.E1 \backslash \{concept\}$
        $concept := getConcept(B.E1, Q)$
    Return (selectedPeers)
end

5. Experiments

We evaluate the performance of the *LPS* algorithm by extending a peer-to-peer simulator PeerSim [20]. We present below also the environment, the integration process and the performance evaluation.

5.1. Environment

To test our approach, we have chosen the PeerSim simulator, 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.

5.2. Integration

PeerSim simulator is a simulation environment of general P2P networks, which is not specialized for P2P information retrieval system. So to test our routing algorithm, we must implement a P2P information retrieval system using the routing method proposed in this paper. To tackle this problem, we decided to use the generic simulator for P2P information retrieval system [21], which can be seen as a specialization of PeerSim for information retrieval. In this context, we implemented a new protocol, a new Initializer to initialize the knowledge base and an observer to view simulation results and update log files.

5.3. Data source characteristics

The data set used in our experiments is the "Big Dataset", developed under the RARE project [22]. This data set was obtained from a statistical analysis on Gnutella system data [35] and from the TREC collection [23], which allows us to simulate our algorithm in real conditions. Big Dataset is composed of *25000* documents and *5000* queries, distributed over 499 peers. Some queries are replicated upon different peers. It provides XML files describing the nodes, the associated documents and the queries to be launched on the network.

5.4. Evaluation measures

To test the quality of our approach, we used the Recall (*R*) metric and the messages traffic, defined as follows for a given query *Q*:

$$R(Q) = \frac{RD}{RRD}$$

With, *RRD* is the number of relevant retrieved documents and *RD* is the number of relevant documents.

Messages traffic = number of messages exchanged to answer a given query *Q*.

5.5. Initial parameters of simulation

The simulation of our algorithm and that used by Gnutella was based on the following parameters:

- *TTL*: The maximum number of hops that a query is allowed to travel (the horizon of the query). We varied this parameter from *2* to *5* to see its impact on the quality of routing.
- *Pmax*: The maximum number of peers to be selected for query, initialized to *3*.
- *Overlay size*: Number of peers in the network, initialized to *500* (number of peers in the Dataset).

In addition, our algorithm requires a knowledge base for each peer. To do this, we launched the first *20.600* queries, using Gnutella algorithm, in order to build initial log file for each peer. Thereafter, we execute the management module to build a knowledge base for each peer from its log file, denoted *B0*. Knowledge base will be periodically updated in order to take new information about new queries. On simulation task, knowledge base for each peer has been updated twice times respectively after *8000* and *6000* queries, for building *B1* and *B2* bases.

5.6. Results

To compare our algorithm LPS to that of Gnutella, we calculate the average recall and average number of messages by interval of *2000* queries sent from different peers by varying the *TTL*. From Figure 3, we can deduce that the average recall for our algorithm LPS is more than that of Gnutella for all variations of *TTL*. It varies between:

- 0.28 and 0.25 if the TTL is equal to 2
- 0.45 and 0.49 if the TTL is equal to 3
- 0.63 and 0.69 if the TTL is equal to 4
- 0.82 and 0.86 if the TTL is equal to 5

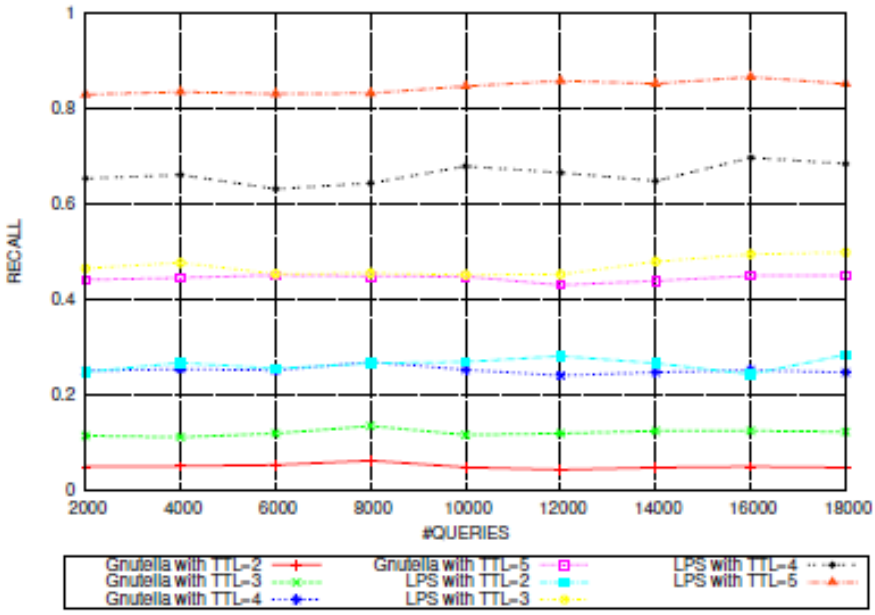

Figure 3. Relation between Recall and Nbr of Queries

However the recall for Gnutella varies between 0.43 and 0.44 in the best cases for a TTL equal to 5. In addition, recall for our algorithm LPS increases after each update of the knowledge base. Figure 4, indicates that the number of messages for our algorithm *LPS* is lower than that of Gnutella for all variations of *TTL*. We note that the difference is very low for a TTL less than 3, the two algorithms produce almost the same number of messages, but this difference becomes important for a *TTL* greater than or equal to 4. In fact, the average number of messages for our algorithm has decreased from 283 by using the initial base B1 to 262 by using B2 and 259 by using B3 for a value of *TTL* equal to 5. Also it increases from 109 by using the initial base *B1* to 106 by using *B2* and 102 by using *B3* for a value of *TTL* equal to 4. By against, the number of messages for Gnutella routing algorithm is stable at 313 if the *TTL* is equal to 5 and at 115 if the *TTL* is equal to 4.

We can deduce that the size of the knowledge base has an impact on the quality of our routing algorithm. Furthermore, the quality of our routing algorithm becomes acceptable from a value of *TTL* greater than or equal to 3. We note that from *TTL* equal to 3, recall of our algorithm *LPS* is higher than Gnutella routing algorithm with *TTL* equal to 5. Therefore, we can say that our algorithm produces best results with an average number of messages equal to 37 than with a Gnutella average number of messages equal to 313. Hence, our approach can be seen as a scalable algorithm because it is based on a distributed knowledge that is avoids making centralized knowledge data base.

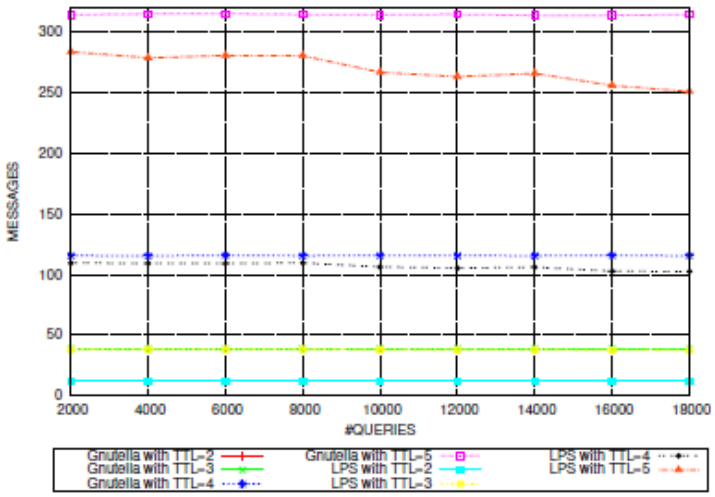

Figure 4. Relation between Nbr of messages per query and total number of queries

6. Conclusion and Future Works

In this paper we presented a new routing method with a full-scale evaluation under unstructured *P2P IR* architecture, focusing also on retrieval effectiveness and search cost. Many approaches suggest using peer's content information in order to route more efficiently queries but few studies explore the information on queries history. In this context, we introduced a novel semantic method for queries routing *LPS*. The experimental results prove the retrieval effectiveness and the search cost of our approach. Obvious pointers for future work include the proposition to improve the effectiveness of our approach during training phase. Indeed, upon joining the *P2P* network, a peer has no prior knowledge; therefore, it is impossible to make smart routing decisions. For this reason, the newly joined peer has to flood a given number of queries and records the returned responses in order to build the initial knowledge base. Hence, methods based on queries history achieve slow improvement in routing effectiveness especially during the training phase.

References

[1] S. Chernov, P. Serdyukov, M. Bender, S. Michel, G. Weikum and C. Zimmer, "Database selection and result merging in p2p web search", in Proceedings of the 3rd International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P 2005), Trondheim, Norway, **(2005)** August 28-29.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A scalable content-addressable network", in Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '2001), New York, NY, USA, **(2001)** August.

[3] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications", IEEE/ACM Transactions on Networking, vol. 11, no. 1, **(2003)**.

[4] M. Deshpande and N. Venkatasubramanian, "The different dimensions of dynamicity", in 4th International Conference on Peer-to-Peer Computing (P2P'04), Zurich, Switzerland, **(2004)** August 25-27.

[5] F. Banaei-Kashani and C. Shahabi, "Criticality-based analysis and design of unstructured peer-to-peer networks as "complex systems"", in Proceedings of the 3rd International Symposium on Cluster Computing and the Grid (CCGRID' 2003), Tokyo, Japan, **(2003)** August.

[6] 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, Dijon, France, **(2009)** April 23-27.
[7] T. Christoph, S. Steffen and W. Adrian, "Semantic query routing in peer-to-peer networks based on social metaphors", in 13th International World Wide Web Conference (WWW' 2004), USA, New York City, **(2004)** May 17-22.
[8] B. Defude, "Organisation et routage s_emantiques dans les syst_emes pair-a-pair", in INFORSID' 2007, Perros-Guirec, France, **(2007)** May 22-25.
[9] A. Kumar, J. Xu and E. W. Zegura, "Efficient and scalable query routing for unstructured peer-to-peer networks", in INFOCOM' 2005, Miami, USA, **(2005)** March 13-17.
[10] A. Crespo and H. Garcia-Molina, "Routing indices for peer-to-peer systems", in Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02), Vienna, Austria, **(2002)** July 2-5.
[11] X. Li and J. Wu, "Searching techniques in peer-to-peer networks", Theoretical and Algorithmic Aspects of Sensor, vol. 11, **(2006)**.
[12] C. Shi, D. Han, Y. Liu, S. Meng and Y. Yu, "A dynamic routing protocol for keyword search in unstructured peer-to-peer networks", Computer Communications, vol. 31, no. 2, **(2008)**.
[13] S. Ciraci, I. Korpeoglu and O. Ulusoy, "Reducing query overhead through route learning in unstructured peer-to-peer network", J. Netw. Comput. Appl., vol. 32, no. 3, **(2009)**.
[14] G. D. Kalogeraki Vana and Z.-Y. D., "A local search mechanism for peer-to-peer networks", in Proceedings of the eleventh international conference on Information and knowledge management (CIKM' 2002), McLean, Virginia, USA, **(2002)** November 4-9.
[15] C. Raja, D. Bruno and H. Georges, "Définition et diffusion de signatures sémantiques dans les systèmes pair-à-pair", in Extraction et gestion des connaissances (EGC'2006), Janvier 17-20, Lille, France, **(2006)**.
[16] S. Ciraci, I. Korpeog lu and O. Ulusoy, "Reducing query overhead through route learning in unstructured peer-to-peer network", Journal of Network and Computer Applications, vol. 32, no. 3, **(2009)**.
[17] Gnutella, Gnutella Web site, http://www.gnutella.com, **(2009)** March.
[18] B. Ganter and R. Wille, "Formal Concept Analysis: Mathematical Foundations", Springer-Verlag New York, **(1997)**.
[19] R. Godin, R. Missaoui and H. Alaoui, "Incremental concept formation algorithms based on galois (concept) lattices", Computational Intelligence, vol. 11, no. 2, **(1995)**.
[20] M. Jelasity, A. Montresor, G. P. Jesi and S. Voulgaris, "The peersim simulator", http://peersim.sf.net, **(2010)** March.
[21] RARE, Le projet RARE (Routage optimissé par Apprentissage de REquêtes), http://www-inf.it-sudparis.eu, **(2010)** March.
[22] S. Goh, P. Kalnis, S. Bakirs and K. L. Tan, "Real datasets for _le-sharing peer-to-peer systems", 10th International Conference on Database Systems for Advanced Applications (DAS-FAA' 2005), 17-20 Beijing, China, **(2005)** April.
[23] TREC, Text REtrival Conference, http://trec.nist.gov, **(2010)** March.