OSQR: A Framework for Ontology-based Semantic Query Routing in Unstructured P2P Networks

DM Rasanjalee Himali[1], Shamkant B. Navathe[2] and Sushil K Prasad[1]
[1]Department of Computer Science
Georgia State University
Atlanta, GA, USA
dmrhimali@student.gsu.edu, sprasad@gsu.edu
[2]College of Computing,
Georgia Institute of Technology
Atlanta, GA, USA
sham@cc.gatech.edu

***Abstract*—Efficient searching for information is an important goal in unstructured peer-to-peer (P2P*)* networks. While several P2P systems have been proposed for data sharing purposes, many support only semantics-free keyword searches or coarser grained file name searches. In this paper, we present an ontology based semantic query routing algorithm that performs efficient semantic search in unstructured P2P overlay networks. In our proposed system, the queries are routed in the network by forwarding to peers with highly relevant content in their local storages. To aid in this semantic query routing, we propose a scheme where each peer in the network adheres to a global ontology and semantically tags its local document collection with concepts in the ontology. Based on the semantic tags, peer level semantic summaries are generated, exchanged with neighboring peers and propagated along search paths which aid in efficient local query processing and overlay query routing. An extensive set of simulations performed to evaluate the effectiveness of the system on P2P networks show 380% and 717% improvement in average recall rate, and 410% and 725% improvement in average precision over Ontology Index based Query Routing [20] and Random Walk [18], respectively for dynamic networks at comparable message overheads. Thus, our approach represents a significant advance in practical terms.**

Keywords- Peer-to-peer search; Semantic searching; Query routing; Ontology-based search; Summarizing semantic knowledge stores

I. INTRODUCTION

P2P systems have become a popular means of sharing large amounts of data among users over the recent years. Based on the network structure, P2P networks can be broadly classified into two groups: structured networks and unstructured networks. Structured P2P networks such as CAN [1] and Chord [2] use a distributed hash table to distribute the data in specified peers. This makes the search overhead of P2P networks very small. The structured pattern of overlay links in this type of P2P networks allows them to provide guaranteed lookups even for rare objects. This however comes with the added disadvantage of low autonomy of shared resources. Structured P2P systems are more suited for exact searches such as object id searches rather than for complex searches such as keyword searches or semantic searches. Unstructured networks, on the other hand, have arbitrarily established overlay links between peers. Therefore, due to the full autonomy a peer exercises over its local resources, there is no correlation between a peer and the resources managed by it. Thus, providing guaranteed lookups is a hard-to-achieve goal in unstructured P2P networks. Unstructured P2P networks, however, can support complex keyword and semantic searches.

Some of the notable works done in the area of semantic P2P searches include pSearch [4], SemereX [5] and SemSearch [6]. Searching for resources in large distributed systems like P2P networks is a challenging task. With the advent of semantic web, more and more users associate their shared resources with semantic meta-information. This not only allows a user to describe resources from a semantic viewpoint but also lets them specify more complex queries addressing several semantic properties or relationships among semantic entities and resources. Majority of search algorithms proposed for P2P networks to date, however, are simple keyword searches or title based searches. These are inadequate for formulating semantic based queries and consequently, do not provide semantically relevant results.

Ontologies are rich data structures that have been successfully used in research as well as in practical applications to represent the knowledge as a set of concepts and relationships between those concepts. Ontologies can be used to represent the semantic concepts, their complex properties, and relationships among the shared documents in a P2P system.

In this paper, we present a novel semantic search approach called Ontology based Semantic Query Routing (OSQR) for an unstructured P2P environment. In this scheme, every peer in the network shares a common ontology and each peer is represented by its own semantic summary vector which is generated based on semantic information extracted from its local document collection and from its neighborhood. The objective of the search is to find more relevant documents at

1 This research is supported by the Brains and Behavior Fellowship, Georgia State University. URL : http://www.cs.gsu.edu/~dmrhimai/osqr.html

978-1-4673-2371-0/12/$31.00 ©2012 IEEE

shorter distances to the query originating peers with higher recall rates at low message cost. In both local query processing and query routing, we exploit the ontology to provide more semantically relevant results. The summary of our main contributions and results in this paper is as follows:

- We develop a novel distributed information retrieval search mechanism. A new concept of *peer semantic vector* is introduced for summary representation of a peer's semantic knowledge. This peer semantic representation effectively combines two important measures to quantify the semantic richness of a peer for a given query: semantic information content of a peer and the total relevant documents reachable through a peer. We incorporate a learning process to incrementally update the peer semantic vectors so the semantic representations improve precision over time when more and more information is acquired by peers about the distributed document collection in P2P system through message exchanges.

- We introduce a novel peer selection algorithm that exploits concept strength in a peer(and a document) and concept relationships in the ontology to provide more semantically relevant results. Compared to a keyword based approach, with little semantic content and laden with a large dimensionality of term vectors, our approach requires limited dimensionality and incorporates semantics in ontology based vectors. We also introduce a novel path-based local knowledge updating mechanism [14] that successfully aggregates semantic information gathered through query paths and incorporates into peer's existing soft state.

- We carry out an extensive set of simulations to evaluate our search system. The experimental results show that our OSQR search mechanism is significantly more efficient than the state-of-the-art search mechanisms, Random Walk[18] and Ontology Index based Query Routing[20], in terms of recall and precision at same message complexity. For dynamic networks where the network topology and number of nodes changed over time, OSQR achieved 380.29% and 410.13% improvement over Random Walk [18] for recall rate and precision, respectively, and 717.15% and 725.67% improvement over Ontology Index Based Query Routing in recall and precision, respectively, at comparable message cost.

The rest of the paper is organized as follows: In Section II we discuss the related research in the area of P2P search. In Section III we present our set of algorithms for ontology based search in unstructured P2P networks. In Section IV, we evaluate the proposed algorithms via simulation. In Section V we discuss the limitations and future roadmap of our work. Finally, Section VI concludes the paper.

II. RELATED WORK

Many search mechanisms for P2P networks have been proposed in the literature over the past few years. While some notable blind search mechanisms proposed for unstructured networks include flooding [15] and random walk [18], popular informed search methods include Routing Indices [16] and intelligent search [19]. However, these methods are intended for simple keyword searches only. Many structured P2P networks like CAN [1] and Chord [2] have also received a lot of attention in the recent past. These systems guarantee that contents can be located in a bounded number of hops. However, they are more suited for exact queries rather than for keyword searches or more advanced queries. Also the data placement and topology in such networks are tightly controlled.

Semantic web allows one to associate semantics of content with World Wide Web resources using a semantic meta-data model. Many current P2P systems incorporate semantic web technologies in their systems with the purpose of providing better knowledge and query representations as well as efficient content discovery.

A good semantic representation is crucial for local query processing as well as informed query routing. Even though there are many well defined semantic representation models such as RDF and OWL freely available, many fail to represent quantified information or probabilistic information regarding semantic concepts or relations. Also, they require non-user friendly query languages to query. Compared to these, our semantic representation model implements a much simpler Peer Semantic Vector (PSV) via the Vector Space Model(VSM), that easily allows one to represent semantically quantified information. Also, unlike many previous methods which solely considered simple statistical information such as total relevant documents per concept, we account for relevance more accurately by evaluating the semantic content of the document using the ontology, the concept frequency based on subsumed concepts, and by setting a relevance threshold. Thus our knowledge representation is semantically richer and can aid in better query routing and improving precision and recall.

Edutella[7] proposes an RDF metadata model and is a super-peer based mechanism and use flooding as a basis for semantic search. However, the notion of super-peers makes the solution less attractive as super-peers need more resources than their sub-peers to maintain all routing indices as well as RDF triples representing sub-peers. This also leads to a substantial overhead in index updating at super-peers when a node joins, leaves, or changes its content. Compared to this, our approach is much simpler and involves minimal overhead in terms of local knowledge maintenance within peers. There is no notion of super peers and each peer only keeps a simple semantic summary vector per its neighbor. pSearch [4] uses a semantic vector to describe and find the resources in the P2P network. SemreX [5] uses a concept tree to construct a semantic overlay network on top of P2P overlay. Both pSearch[4] and SemreX[5] use Latent Semantic Indexing (LSI) [9] to map documents to semantic vectors. LSI however is computationally expensive, and when the network content changes frequently, this method needs to be applied often. In OSQR, we do not employ computationally expensive methods like LSI. Rather, we use concepts in a shared ontology to build peer semantic summary vectors to aid in the query routing

process. The process of constructing peer semantic vectors is much simpler and consumes less memory and time in constructing them. The related work most comparable to ours is [20] where authors propose a semantic based search algorithm for unstructured P2P system. This method employs ontology based indices to route the query in the network. However, the notion of semantic information content is not taken into account in [20] and the index generation and updating is resource consuming and generates considerable traffic. Compared to this, OSQR provides a richer semantic knowledge representation, which combines both total documents reachable from each neighbor of a peer as well as semantic richness of a peer based on its information content. OLI [21] is yet another ontology based indexing method for structured P2P systems. Semantic search methods applied in structured networks such as in [21] however, generally suffer from the shortcoming of structured overlays such as strict enforcement of data placement and high maintenance cost of peers joining and leaving as well as the cost of content updating. To avoid these problems, we consider unstructured P2P overlay for our work. Ontsum [22] is another ontology–based indexing scheme where heterogeneous ontologies between peers are assumed. This approach however requires computationally expensive ontology generation and ontology mapping techniques to be employed in a distributed setting and allow only RDQL based queries which requires some level of end user expertise to be used. Such techniques are not required in OSQR as all peers use a single globally agreed-upon ontology that is downloadable with the application.

III. PRELIMINERIES

A. System Model

We make the following assumptions about our proposed search model:

1) Network Topology

We consider an unstructured P2P network with a set of peers $\{P_1, P_2, ...,P_p\}$with average degree γ. This P2P network is modeled as an undirected graph $G = (V,E)$ where V is the set of nodes that represent peers and E is the set of edges between neighboring peers in the overlay. Our OSQR protocol is built on top of this P2P overlay. We assume that each peer in the network is able to communicate with its direct neighbors (i.e. one hop neighborhood) only. Each peer also maintains a limited amount of soft state about its direct neighborhood which is used for informed query routing.

2) Global Semantic Ontology:

A globally known common semantic ontology is agreed upon by all peers in the network. This ontology O consists of set of semantic concepts $C = \{c_1, c_2, ...,c_m\}$ with total m concepts and relationships among these m concepts. Each peer maintains a copy of the global ontology in its local storage. The ontology is included along with the P2P client application downloadable at the vendor's site and thus can be obtained by a peer connecting to the P2P network the first time.

For simplicity, we assume a semantic hierarchy where the relationships among concepts are only IS-A(i.e. hypernym/hyponym) relations. This semantic hierarchy can be easily replaced by a more complex ontology with relationships other than IS-A.

3) Data Distribution:

Each peer in the system maintains a collection of documents that is shared with the rest of the peers in the network. There can exist multiple copies of the same document distributed among peers. Each document typically contains several concepts in the semantic ontology. A peer constructs, for each document in its storage, a *concept weight vector* that depicts the semantic weight of each concept in ontology for that document. For a peer P with set of *documents* $D(P) = \{d_i, i=1,2,..,n\}$ in its local storage, we use the following Concepts Vs. Documents matrix $CD(P)$ to describe the concept weights of the documents in P for the concepts in the shared ontology:

$$CD(P) = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix}$$

where w_{ij} is the weight of concept s_i in document d_j. The weighting process is described below.

4) Document Semantic Weighting:

We represent each document in a peer's local document collection as a weighted ontology. A weight w_{ij} of concept c_i in document d_j is calculated as follows: First, the concept frequencies of all concepts for each document in peer's local storage are calculated by a process of text mining followed by word sense disambiguation. In a document, an occurrence of a concept is also an occurrence of each ancestor concept in the semantic hierarchy due to the hypernym/hyponym relationship between a concept and its ancestor concepts in the hierarchy. Therefore, once occurrences of individual concepts in a document are determined, they are updated bottom-up, by adding occurrences of the concepts in the sub-trees rooted under them. Then these concept frequencies of each concept are normalized to a [0-1] range by applying *maximum frequency normalization* per concept. In this process, each concept frequency in a document is normalized into the [0,1] range by finding the maximum frequency of that concept (and descendent concepts) in all known documents for the peer. Notice that 'known' documents are not as same as 'locally contained' documents in a peer. Initially at configuration stage, however, the known documents of peer solely come from its local document collection. These *concept weight vectors* are re-normalized whenever a peer acquires more knowledge about the distributed document collection in P2P system. When and how these *concept weight vectors* are later recomputed will be discussed under section IV-B.

5) Semantic Query:

We assume that there are a fixed set of queries $Q=\{Q_1,Q_2,...,Q_q\}$ that are issued by peers. The queries we consider in this work are concept queries where a query consists of the conjunction of one to n concepts in the ontology. The document locating problem therefore is to find as many documents in the P2P network that satisfy the query

and that exceed a pre-specified relevance threshold. For example, a query Q_i = *{Semantic search, P2P Networks}* is a query with conjunction of the two concepts, and is intended to find documents related to both "semantic search" and "P2P networks." The query also specifies the relevance threshold the qualifying documents should exceed, and the number of walkers to be deployed. This relevance threshold is set for a system based on the nature of documents and as a consensus among users who are querying. Number of walkers is a parameter used in experimentation. Existing techniques [23,24] proposed by the research community can be employed to convert keyword queries to concept queries.

B. Peer Semantic Vectors

A peer semantic vector (*PSV*) is the semantic knowledge representation of a peer. It shows the goodness of a peer *P* for semantic concepts contained in its local data storage *C(P)*. We formally represent the peer semantic vector of a peer *P* as a set *PSV(P)* as follows:

$$PSV(P) = \{(s_i, c_i) \mid s_i \in \Re, c_i \in C(P)\}$$

where s_i is the total number of relevant document reachable through *P* for concept c_i. How to construct and maintain these PSVs will be discussed in detail under Section IV-B.

C. Data Structures

A peer P's local knowledge consists of its Concepts Vs. Documents matrix *CD(P),* its own PSV *PSV(P)* , the*PSV(P′)* for each neighbor P' and finally a vector containing maximum concept frequency known per concept for its local concept set (i.e. set of concepts existing in its local document collection)*MaxVector(P).* This *MaxVector(P)* is used for document re-normalizing purposes.

IV. OSQR Semantic Search Protocol

In this section we present our OSQR search algorithm that enhances the semantic search capability of unstructured networks.

A. OSQR Search Algorithm

OSQR is in nature a controlled flooding algorithm where a query is intended to find documents that contain all of the concepts specified in a query exceeding a given relevance threshold. A search process is initiated when a peer originates a query by deploying multiple search walkers which is a system specified parameter. Each walker can follow its own path in the network. Upon receipt of a query, a peer searches for matching documents in its local storage by ranking its documents based on a relevance score. The documents that exceed a given threshold of relevance are selected as matching documents for the query. The query is also further propagated to more promising neighbors. A walker propagates TTL hops only. Once all the walkers for the query terminate, the results are returned to the query originator along the reverse path of the query back to the query originator.

Algorithm 1: Query Routing

Input :
P = this peer executing query routing procedure
k = Walker count
x = concept count in MaxVector(P)
C_{Ans} = concepts including $Q_{Mes}.Q_c$ and their ancestor concepts in ontology
Q_{Mes} = Query Message (Q_c = Queried Concept Set, P_v = Visited Peers list, TTL, $MaxVector_{sub}$ = subset of MaxVector of query sender for its most recently updated x concepts, R_Q = RelevantDocs for Q_c per peer in P_v, R_c = relevant docs per concept in C_{Ans} per peer in P_v)
CandidateNeighbors(P) = Neighborhood(P) – $Q_{Mes}.P_v$
NeighborSematics= {PSV(P_i)} | $P_i \in$ Neighborhood(P)}

Procedure:
Q_{Mes}.TTL = Q_{Mes}.TTL-1
Q_{Mes}. P_v.add(P)
RelevantDocs(P) = Perform local search($Q_{Mes}.Q_c$)
$Q_{Mes}.R_Q$.add(P, RelevantDocs(P))
If(Q_{Mes}.TTL>0) **then**
  //Perform local updates
  Perform MaxVector Updating, Data Renormalization and PSV Update using Q_{Mes}
  //Update message by appending necessary information
  $Q_{Mes}.MaxVector_{sub}$ = most recently updated x concepts in MaxVector(P)
  For Concept c in C_{Ans} **do**
    $Q_{Mes}.R_c$.add(P, c, Relevant documents for c in P)
  End For
  //select neighbors to forward query
  For each neighbor P_i in CandidateNeighborhood(P) **do**
    Relevance (P_i,$Q_{Mes}.Q_c$) = RelevanceScore(PSV(P_i), $Q_{Mes}.Q_c$)
  End For
  Sorted_Neighbors = Sort neighbors in descending order of relevance
  SelectedPeers(Q_{Mes}) = get Top k Peers(Sorted_Neighbors)
  Forward Q_{Mes} to SelectedPeers(Q_{Mes})
Else
  //genrate response message
  R_{Mes} = generate response message
  //Update message by appending necessary information
  $R_{Mes}.MaxVector_{sub}$ = most recently updated x concepts in MaxVector(P)
  Send R_{Mes} to Q_{Mes} sender

B. Construction and Maintenance of Peer Semantic Vectors

1) Initialization:

At the configuration stage when a peer joins the network for the first time, it initializes each of its semantic weights s_i per each concept c_i in its PSV with total relevant documents in its local storage for that concept. Once connected to the network and after the construction of its own *PSV*, each peer *P* exchanges its *PSV* with each of its direct neighbor P' (i.e. one hop neighborhood). These collected *PSV*s from a peer's neighbors essentially serve as the local knowledge the peer has about its neighborhood that is exploited by the peer in forwarding queries to most promising neighbors. These neighbor *PSV*s received are further used to update peer's own *PSV*, so that each semantic weight s_i per concept c_i reflects the total reachable relevant documents per concept through *P*. The weight s_i associated with a concept c_i in a *PSV* in peer *P* is calculated as follows:

$$s_i = N_{i,p} + Avg\left(\sum_r \frac{N_{i,P_r \in QueryPath\ (P')}}{\alpha(P, P_r)}\right) \quad (1)$$

Where, $N_{i,P}$ is the total number of relevant documents for concept c_i in *P*'s local storage, P_r a peer reachable to *P*

through neighbor P′ from a query path (or query response path) $QueryPath(P')$. N_{i,P_r} is the total relevant documents for concept c_i in peer P_r's local storage and $\alpha(P,P_r)$ a distance based weighing factor. $\alpha(P,P_r)$ ensures that the farther away a peer P_r is from P, higher the distance based weight applied in s_i calculation. We use the hop distance in number of hops from P to P_r as the current $\alpha(P,P_r)$. Thus $Avg\left(\sum_r \frac{N_{i,P_r \in QueryPath(P')}}{\alpha(P,P_r)}\right)$ represents the average number of relevant documents reachable for c_i per hop from P through neighbor P'. This is the aggregate reachability of documents for concept c_i from P through a neighbor P'. The $QueryPath(P')$ is the best possible path known by P so far that goes through P from a neighbor P' and gives highest number of relevant documents per hop distance for the considered concept. An existing s_i value is replaced by a newly calculated s_i, only if this new value exceeds the current s_i.

For example, consider that for a concept c_i, current s_i value in a peer P is 5, and 2 comes from its local document collection(i.e. $N_{i,p}$). Assume P receives a query along the path $P_A \rightarrow P_B \rightarrow P_C \rightarrow P$ through a neighbor P_C. Also assume that for concept c_i, P_A contains 9 documents in its local storage (i.e. N_{i,P_A}=9), P_B contains 8 documents in its local storage, and P_C contains 5 documents in its local storage and this information is included in query received at P. Now P calculates a new s_i = 2 + Avg(9/3 + 8/2 + 5/1) = 2 + 4 = 6. Therefore previous s_i will be replaced by 6 in P's *PSV*.

As mentioned before, for a document to be considered relevant for a given concept c_i, it should contain a concept frequency exceeding a certain concept frequency threshold for c_i. This concept frequency $CFThresh_{i,p}$ for concept c_i at peer P is calculated as follows:

$$CFThresh_{i,P} = \text{Re}lThresh \times MaxVector_i(P) \quad where\ c_i \in C(P) \quad (2)$$

Where *RelThresh* is the system specified relevance threshold for a document which is a value in [0-1] range. $MaxVector_i(P)$ is the maximum concept frequency known by P for concept c_i. This is the maximum concept frequency for concept c_i that P discovers from its local documents and through messages it received from its neighbors in the past. This threshold calculation ensures P is represented by only those documents having large enough semantic weights for the given concept. For example, let the known maximum number of occurrences of concept c_i in a single document be 40, If the threshold is set to 0.7, those documents that have 40×0.7=28 or more occurrences of c_i will be considered relevant.

Algorithm 2: MaxVector Updating

Input :

NeighborMaxVector = <*concept, Max(w_{ij})*>key-value pair for last updated n concepts of neighbor

MaxVector(P) = <*concept, Max(w_{ij})*>key-value pair for concepts of this peer's local concept set

LocalConceptSet = set of concepts in this peer's local document collection

P : This peer Id

N: Neighbor peer Id

Output:

Updated MaxVectpr(P)

Procedure:

```
For each concept cn in NeighborMaxVector do
  Ancestor concepts ← get ancestor concepts of c
                      in local document collection
  For each concept cp in Ancestor concepts do
    If Max(w_nj) at N > Max(w_pj) at P then
      //replace existing Max(w_pj) with Max(w_nj)
      MaxVector(P).add(cp, Max(w_nj))
    End If
  End For
End For
```

2) Maintenance:

Each time a peer sends or forwards a query or a query response to one of its neighbors, it piggybacks its total relevant documents for most recently updated concepts in its PSV and $MaxVector(P)_i$ values for intelligently selected set of concepts to its message recipient. As a result, the following three updating mechanisms (i.e. (i)MaxVector learning (ii) data renormalization and (iii)PSV Updating) occur in a peer for improving the semantics of a peer's document collection during query forwarding (*forward update*) as well as in return message sending (*backward update*). The ultimate goal of these updating mechanisms is to achieve highly precision in query execution. Over time, peers' *MaxVector* values will reach their global equivalents with high accuracy resulting in the its normalized data collection being highly precise thus leading to high precision in local query evaluation.

(i) MaxVector Learning:

We incorporate a learning process with *MaxVector* of a peer *P*. *MaxVector(P)* in *P* ideally should represent the maximum concept frequencies of each concept over all documents distributed in the P2P network. Since acquiring such global information is not feasible in a completely decentralized network, each peer *P* initializes its *MaxVector(P)* with local maxima of concept weights per each concept c_i calculated from its local document collection. These are later replaced by new concept weights whenever new concept weights that exceed peer's current maxima are discovered by that peer during the query process.

MaxVector learning is achieved with the aid of normal query traffic. Whenever a peer receives query or a query response from one of its neighbors, it uses the neighbor's subset of *MaxVector* appended to the message to update its own *MaxVector*. Then it forwards the query (response) to the next message recipient by replacing the subset of *MaxVector* in the received message by its most recently updated set of

concepts in its own *MaxVector*. For a message sending peer *P*, the subset of its *MaxVector* in message is a set of key-value pairs that takes the form $<c_i, MaxVector_i(P)>$.

Updating *MaxVector(P)* at a given peer *P* requires some computation, as the peer and its neighbor *P′* have different local concept sets. However, these concepts may still be related by *hypernym/hyponym* relations in the semantic ontology. Due to these hypernym/hyponym relations, for each $<c_i, MaxVector_i(P')>$key-values pair received from a neighbor *P′* ,*MaxVector(P)* values for concept c_i and its ancestor concepts existing in *P*'s *MaxVector* need to be updated. Over time, these $MaxVector(P)_i$ values will propagate over the network through query messages and query responses, and thus each will eventually approach its global maximum equivalent. Algorithm 2 states the $MaxVector(P)_i$ updating process in a given peer. Note however, that the number of updates made to a *MaxVector(P)* at a peer *P* should ideally diminish over time as maximum concept weights for those concepts maintained locally reach their global maxima equivalent from previous updates. Therefore, subsequent messages exchanged between peers should not carry maximum concept weights for unnecessarily large number of concepts selected blindly. Hence, to reduce the time required for updates, each peer intelligently selects a subset of its *MaxVector* values to be piggybacked to its neighbor in the following manner:

Assume a scenario where a peer *P* has two neighbors *P′* and *P″*. *P* receives a query from *P′* which in turn it sends to *P″*. Upon receipt of the corresponding query response from *P″*, *P* forwards it to *P′*. Each peer in the system keeps track of its set of most recently updated *n* concepts in its *Maxvector* and piggybacks some entries in query messages it sends to its neighbors. Therefore when *P* receives a set of $MaxVector(P')_i>$ key-values pairs from *P′* through a query, it updates only those concepts in its *MaxVector* that has a significantly lower maximum concept frequency than that is stated in the received $<c_i, MaxVector(P')_i>$key-values pairs (set to 50% in experiments) . For example, assume an over-simplified scenario, where *P′* sends P in the query message, {(c_1, 150), (c_2,90)} as the set of $<c_i, MaxVector(P')_i>$ key-values pairs for the most recently updated two concepts in its *MaxVector(P')*, *P* has {(c_1,30),(c_2,80)(c_3,50) } in its *MaxVector(P)*, and in ontology structure, c_3 happens to be an ancestor concept of c_2. Given that the update threshold is 50%, *P* will therefore update its *MaxVector* to {(c_1,150),(c_2, 80),(c_3,150)}as concepts c_1 and c_3 have lower than 150*50%=75 maximum concept frequency recorded in *P*'s *MaxVector*. c_2 however already has greater than 90*50%= 45 maximum concept frequency recorded in *P*'s *MaxVector* and therefore will not be updated. *P* also temporarily keeps a copy of this received$<c_i, MaxVector(P')_i>$ key-value pairs from the query sending neighbor until it receives a query response from *P″*. Upon receipt of query response from *P″*, *P* updates its own *MaxVector(P)* with $<c_i, MaxVector(P'')_i>$key-values pairs embedded in the query response and then compares it's just updated *MaxVector(P)* with temporarily stored$<c_i, MaxVector(P')_i>$ key-value pairs sent from query sending neighbor *P′* to select only those concepts in its local concept set that have a drastic increase over maximum concept frequency stated in relevant concept in received $<c_i, MaxVector(P')_i>$ key-value pairs (again set to 50% in experiments) to be piggybacked in the query response forwarded to its neighbor *P′*.

(ii) Data Re-normalization:

The updating of $MaxVector(P)_i$ values for certain concepts of a peer *P* in turn triggers re-computing of normalized concept weight vectors for its local documents where these concepts exist. Therefore over time, the normalized weights in *DC(P)* of *P* reach the more accurate globally normalized values. This helps in retrieving more accurate results at local query processing reducing the opportunity to incorrectly identify irrelevant documents as relevant to a query. To reduce the computation cost, this process can be done for only those concepts updated in *P*'s *MaxVector* and only when a significant change in *MaxVector* values for those concepts are detected.

(iii) PSV Updating:

Updating of a peer *P*'s own *PSV* is triggered by two events (a) receipt of a set of $<c_i, MaxVector(P'')_i>$ *key*-values pairs of and (b) receipt of $< c_i, N_{i,P''}>$ *key*-values pairs per hop distance. The c_i values in $< c_i, N_{i,P''} >$ include not only queried concepts, but also the ancestor concepts of those queried concepts in the ontology. Thus, the updating of PSV using (b) allows a peer to gather goodness of neighbors for concepts in the ontology prior to querying for the same and thus this greatly reduces the initial random choice of neighbors in query forwarding. While (a) affects the former part (i.e. $N_{i,P}$, the locally relevant documents per concept) of (1) , (b) effects the latter part of (1) (i.e. $Avg\left(\sum_r \frac{N_{i,P_r \in QueryPath(P')}}{\alpha(P,P_r)}\right)$,the aggregate relevant documents reachable per concept through the best known query path so far).Each time a peer receives a query or a query response, it utilizes the information embedded in the message to update its *PSV*. Before sending, each peer appends its ID , the total number of relevant documents for queried concepts and their ancestor concepts in its local storage to the message. Therefore, while a $<c_i, MaxVector(P'')_i>$ *key*-values pairs propagate only one hop distance, $< c_i, N_{i,P''} >$ *key*-values pairs of each peer in query path are embedded in the forwarded message propagated along query (or query response) path. These are utilized for PSV updating by each message receiver peer. In (a), receipt of $<c_i, MaxVector(P'')_i>$ *key*-values pairs will result in receiver peer *P* updating its necessary entries in its *MaxVector*. This in turn leads to data renormalization in *P* which in turn leads to recalculating of its *PSV*, as the total local relevant documents for those concepts whose $MaxVector(P)_i$ values have been just updated may have changed. In (b), the query path information is available to a receiver to calculate the latter part of (1): $Avg\left(\sum_r \frac{N_{i,P_r \in QueryPath(P')}}{\alpha(P,P_r)}\right)$ and recalculate (1). An s_i value

recalculated this way replaces current s_i value in *PSV* only if this newly calculated value exceeds current value.

Also, when a new set of $PSV_i(P')$ values are received from a neighbor P', the appropriate entries in PSV maintained for neighbor P' in P's soft state are replaced by these most recent values.

When a peer leaves the network, it informs its direct neighborhood that it is leaving the network by sending a message. This results in the neighbors removing the leaving peer's *PSV* from their soft state and updating their *MaxVector* accordingly. When a peer joins the network, it goes through the same initialization process given in step 1 to construct its *PSV* and exchange the *PSV*s among neighbors. When changes in peer contents occur, the peer updates its *PSV* and notifies its neighbors who will, as a result, update the current *PSV* maintained for neighbor by the new information provided in the notification.

C. Peer Selection

A peer forwards queries to the most promising set of neighbors. In our work, the most promising neighbors are selected based on the neighbor *PSV*s a peer maintains of its immediate neighborhood. Peer P calculates a relevance score for each neighbor P' for the given query Q based on the *PSV* of that neighbor. For a neighbor P' with $PSV(P')$ at P, the relevance score of P' for query $Q = \{ c_i \mid c_i \in C)\}$can be formally defined as follows:

$$Relevance(P',Q) = Min(s_i) \quad where\ (s_i,c_i) \in PSV(P')\ at\ P\ and\ s_i \in Q$$

The relevance of neighbor P' for a query is computed as the minimum of semantic weights for queried concepts. This is because the query is a conjunction of concepts and minimum is the conservative outcome (i.e. relevant documents) from P'. Once the relevance score of each neighbor is calculated, the neighbors are ranked in descending order of their relevance scores and top k neighbors are selected to deploy the query. The query originator peer selects a predefined number of most promising neighbors to send the query while the other peers receiving the query merely process and forward it to only one promising peer if the message TTL has not exceeded.

D. Local Search inside a Peer

Once a query is received at a peer, it evaluates the query against its local storage. For a document to be considered relevant for the query, it must contain all the concepts in query and furthermore, should have a relevance exceeding a predefined relevance threshold. For a document d_i in peer P's local storage, the relevance score of d_i for query Q, $RelevanceScore(d_j,Q)$, is computed as the minimum of its normalized concept weights for those queried concepts:

$$Relevance\ Score(d_j,Q) = \underset{i}{Min}(w_{ij}) \quad where\ c_i \in Q$$

where w_{ij} is the weight of concept c_i in document d_j.

V. Design of Experiments

In this section we present our experimental design parameters such as query generation, document generation, shared ontology, and state-of-the-art algorithms used for comparison purposes.

A. Semantic Ontology

We used the Reuters21578 text classification [13] corpus of newswire documents as the basis of our semantic ontology. The Reuters ontology text classification contained four core concepts into which the documents were categorized: *Organization*, *Exchange*, *Person* and *Country*. We used hypernym/hyponym relations of Wordnet ontology to further extend this classification by adding descendent sub trees of each of these core concepts and also to create the core ontology by adding all ancestor concepts of these core concepts in all ancestor concept paths toward the root concept *entity* in Wordnet ontology. The ontology built this way contains a total of 11,813 concepts. A partial view of the used Reuters semantic hierarchy is given in Fig. 1.

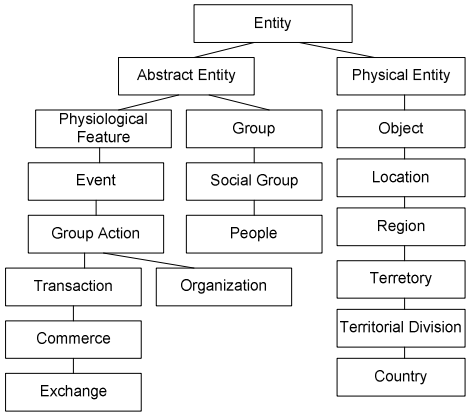

Figure 1. Extended Reuters21578 Core Semantic Hierarchy

B. Data Generation

We used the actual documents from the Reuters21578 dataset in our simulation experiments. There were 21,578 newswire documents in dataset and after text processing and words-sense disambiguation, a total of 15,191 documents that produced non-zero length *concept frequency vector*s were selected. For each of these documents, a concept frequency vector was generated by analyzing the document. Then these concept frequencies per document were updated by recursively adding the descendants' concept frequencies. The documents were replicated according to a Zipf distribution into multiple copies for allocating to peers.

C. Network Generation

We implement our system on top of Peersim [12] simulator which is a discrete-event time-stepped simulator for P2P systems. The simulation was carried out over networks generated with Gnutella scheme following the power law topology. The network size was varied from 100 to 100000 nodes Average peer degree was set to 10. The common ontology used is the extended Reuters21578 semantic hierarchy [15]. The document distribution among peers follows a Zipf (α=1.0) distribution and each peer contained 100 documents on an average in its local storage. In total we selected 15,191 documents distributed in the P2P network. The queries were

also distributed according to Zipf distribution among peers in the network (α=1.2). The default TTL was set to 7 to limit the number of hops of walkers. Percentage increment threshold for *MaxVector* update is set to 50% to ensure that recalculation of document-wide *concept frequency vector*s is infrequent.

To simulate the dynamic behavior of the network under peer churn, we inserted online nodes to the network while removing active nodes at varying frequencies. On an average, 80 nodes each are added and removed from the network during each simulation run.

D. Query Generation

We generated 100 random queries as conjunctive queries from the concepts in the ontology. The root concept *Entity* was excluded from the candidate set of concepts to generate a query. The number of concepts in a query was varied from 1 to 2. Only meaningful queries were generated. A query is defined to be meaningful if no two concepts in the query are involved in an ancestor-descendent relationship.

E. Comparison Algorithms

We compare the performance of our algorithm with the state-of-the-art algorithm Ontology based Index for Unstructured Networks [20] and Random Walk [18] based search. They work as follows:

1) Ontology Index Based Query Routing (OIQR):
Authors of [20] introduce constant size ontology-based indexing approach for unstructured P2P networks.A matching function which uses the number of documents accessible via a link to rank and select neighbors for query forwarding. OIQR assumes a global ontology. For a fair comparison, to obtain best-case performance of OIQR, we assumed that the predefined index size is equal to the total number of concepts in the ontology. We set its terminating condition to be TTL.

2) Random-Walk-based Query Routing (RWQR):
Random walk [18] is a popular blind search mechanism where a querying peer deploys k search walkers by sending query to k random neighbors. The peer selection mechanism does not require a peer to maintain any local knowledge about its neighborhood in this method.

F. Performance Metrics

For our evaluation we rely on three key measures:

*1) Recall:*Recall is the ratio of the number of relevant results obtained against the total number of relevant results in the entire P2P network.

*2) Precision:*Precision is the fraction of documents retrieved that are relevant to a search query.

*3) F-Score:*This is the harmonic mean of precision and recall. This provides an overall measurement of system efficiency by considering both recall and precision. We use F1 score as F measure:

$$F1\ Score = 2.\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

*4) Message Cost:*This is the average number of messages generated per search query.

*5) Hits Per Query:*This is the average number of distinct relevant documents discovered per search query.

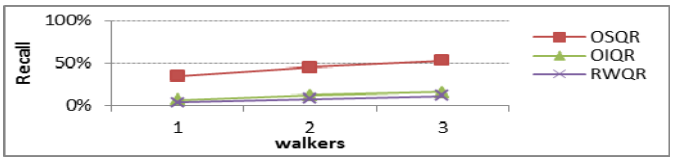

Figure 2. Average recall rate comparison between OSQR, RWQR and OIQR.

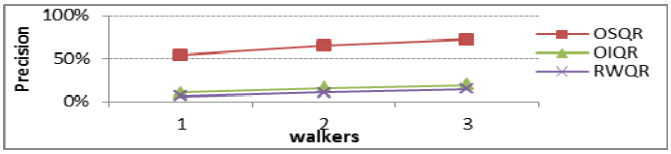

Figure 3. Average precision comparison between OSQR, RWQR and OIQR.

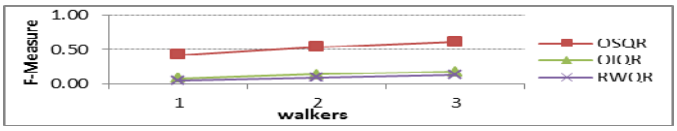

Figure 4. F-Measure comparison between OSQR, RWQR and OIQR.

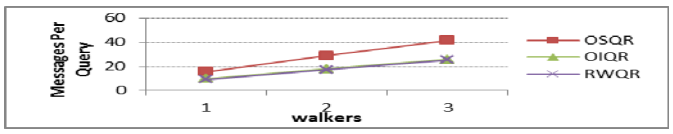

Figure 5. Search cost comparison between OSQR, OIQR and RWQR.

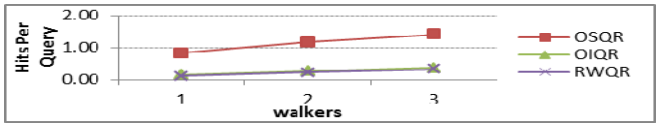

Figure 6. Hits Per Query comparison between OSQR, RWQR and OIQR.

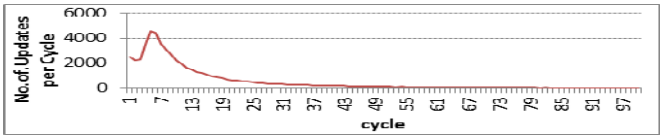

Figure 7. PSV updating cost of OSQR. The updating cost is calculated as the number of updates made per P2P simulation cycle

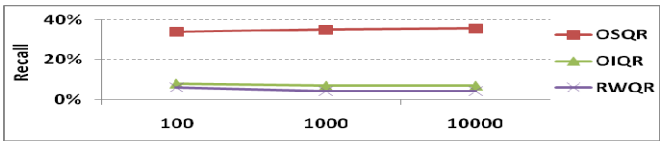

Figure 8. Average recall rate comparison between OSQR, OIQR and RWQR for varying network sizes

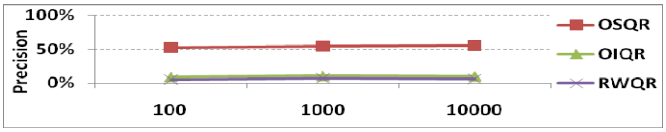

Figure 9. Average precision comparison between OSQR, OIQR and RWQR for varying network sizes

VI. Results And Analysis

In this section, we discuss the results we obtained for P2P environment. Figures 2 through 9 summarize the results. We measured performance of the search algorithms for various network sizes, and results were consistent regardless of the network size. Therefore, we present the results for network of 1000 nodes only. Our simulations show that our OSQR algorithm improves recall and precision significantly over Random Walk (RWQR) as well as Ontology Index based search (OIQR) counterparts at a comparable message cost while keeping PSV update cost low.

1) Recall

We measured the recall of OSQR, RWQR and OIQR for different number of walkers while fixing the TTL to 7. The experimental results are shown in Fig. 2 for deploying 1, 2 and 3 search walkers per query. One of the major goals of search is to achieve high recall rates with fewest search walkers. While our OSQR algorithm achieved 34.70%, 45.39%, and 53.49% recall rates on average for 1, 2 and 3 walkers, respectively, OIQR achieved only 7.22%, 12.31% and 16.26% recall and RWQR achieved only 4.25%, 8.13% and 12.12% for the respective number of walkers. Thus, OSQR shows an outstanding 380.29% improvement over OIQR and 717.15% improvement over RWQR even when walkers per query is as low as 1. The reason for better performance of OSQR comes from the fact that our semantic representation, PSV, is superior to their counterparts as it incorporates the twin notions of semantic richness (i.e., information content) of a peer and total qualified documents reachable through a peer per ontology concept. Therefore, in query routing, these semantic structures are better exploited by incorporating ontology knowledge to route queries to more promising peers. OIQR and RWQR, on the other hand, do not incorporate information content measures in query routing or query evaluation. Mere existence of query concepts in a given document is considered sufficient criterion for document relevance in OIQR.

2) Precision:

Fig. 3 shows experimental results for precision for 1, 2, and 3 search walkers for OSQR, OIQR and RWQR algorithms. OSQR achieves a precision of 53.98%, 65.21% and 72.32% for 1, 2, and 3 search walkers, respectively. For the same number of walkers correspondingly, OIQR achieved only 10.58%, 16.00%, 19.59% precision and RWQR only, 6.54%, 10.96%, 14.70% precision. Thus, an average of 410.13% and 725.67% improvement in precision is observed in OSQR over OIQR and RWQR, respectively, for a single walker. The mechanisms of re-normalizing local document collections and updating PSVs contribute to this improvement in precision of OSQR over their counterparts. Re-normalizing the dataset essentially gears the normalized concept weight vectors of documents toward their globally normalized steady-state values, reducing the possibility of incorrectly identifying documents irrelevant to a query as relevant in local query processing. Updating PSVs with information discovered from the network makes the semantic representations more accurate and precise over time leading to better judgment of peer selection in query routing. Fig.8 and Fig. 9 shows the recall and precision comparisons respectively for the three algorithms for different network sizes. The results demonstrate that OSQR clearly outperforms both OIQR and RWQR in the P2P network at scale.

3) F-Measure

Fig. 4 depicts the F-Measure values which is considered a combined metric of retrieval performance for the three search algorithms for 1, 2, and 3 search walkers. F-Measure for OSQR is clearly superior to the other two comparison algorithms. OSQR achieves 0.42, 0.54, and 0.61 for F-Measure for 1, 2, and 3 walkers, respectively, whereas OIQR only achieves 0.09, 0.14 and 0.18 and RWQR 0.05, 0.09, and 0.13 for the respective number of walkers.

4) Search cost

As mentioned before, our goal is to achieve better recall with fewer walkers deployed per query in order to minimize the cost of search. As experimental results show, OSQR achieves this goal successfully. The search cost was measured in terms of the average number of messages generated for a given query. Fig. 5 shows the experimental results for search cost. As shown in figure, the search cost of OSQR is twice that of OIQR and RWQR. This is because, in OSQR, a search response returns in the reverse query path back to the query originator compared to OIQR and RWQR, where search response message is directly sent to query originator from the query response generator. Since each walker in OSQR travels *TTL* hops, the number of message exchanges per query is $2 \times k \times TTL$ for k-walker query. Note that this measure captures search traffic only. OIQR has a separate index updating mechanism which results in non-search based message traffic generation that is not represented in Fig. 5. Our OSQR algorithm, on the other hand, effectively utilizes search messages and their responses for PSV updating purposes without generating extra messages not intended for search purposes.

5) Hits Per Query

Fig. 6 depicts the hits generated per search query for the three algorithms. It was observed that OSQR performs better compared to the other two algorithms. While OSQR generated 0.82 hits per single search walker, OIQR generated 0.15 hits and RWQR generated 0.12 hits only for one walker.

6) Time Complexity

Receipt of a query or a query response message results in a peer updating its *MaxVector*, which triggers re-normalizing its document collection. This in turn triggers recalculation of its PSVs for the affected concepts. Therefore, in the worst case, all $|C|$ concepts need to be updated. Therefore, for a given peer P, the time complexity of the algorithm whenever an renormalization is triggered is $O(|C|) \times O(|D(P)|) \times O(|PSV(P)|)$ where $|C|$ is the total number of concepts in the network, $|D(P)|$ the total number of documents in peer P and $|PSV(P)|$ the size of PSV of P or its local concept set. Fig. 7 shows how the updating cost of a PSV undergoes initial transitions and quickly diminishes over time. Simulation cycles here represent the time, while number of updates per cycle represents the *PSV* update cost. Each peer in the network issues 1 query per

simulation cycle. The total number of *PSV* updates keeps increasing till 7th cycle. This is because peers start gaining knowledge from the network when queries start to be issued thus resulting in more updates per-cycle. In our simulation engine, a walker travels only 1 hop per cycle and, thus, 7 cycles are needed for a walker to travel TTL hops. Due to this intensity of new knowledge gained from this forward updating, *PSV* update cost increases drastically. We noticed that, after 7th cycle, the total number of updates per cycle starts to diminish and fewer updates will be necessary when peers gain decreasingly fewer knowledge. At the end of the simulation, we observed that number of updates wear off to zero. Therefore, our algorithm effectively limits the computation time required to perform *PSV* updates while keeping the accuracy of *PSV* sufficiently high.

7) Space Complexity

Each peer *P* stores a *MaxVector*, *PSV*, neighbor *PSV*s, and the local document concept weigh vectors *CD(P)*. Given that the size of the local concept set is $|C(P)|$, *MaxVector*, *PSV* and each document concept weight vector in *CD(P)* takes $O(|C(P)|)$ space each, while each neighbors' *PSV* at *P* takes $O(|C(P')|)$ space.

VII. Conclusion and Future Work

In this paper, we propose a novel semantic query routing algorithm called OSQR for unstructured P2P networks. We effectively exploit semantic properties and their relationships in a semantic hierarchy to achieve more informed query propagation as well as local search and best path for each concept [14] to provide more relevant results. OSQR allows peers in the network to maintain limited soft state per neighbor. It is a simple and easy to implement algorithm. Experimental results show that OSQR outperforms Random Walk and Ontology Index based Query Routing, in terms of recall, precision and hits per query for comparable message costs. We plan to extend this work by developing a search algorithm that exploits a more complex ontology structure with relations other than hypernym/hyponym to perform query routing and query processing. We plan to use alternate metadata structures other than the PSV at each peer for exploration. We also plan to experiment with more complex and realistic queries on different document corpuses.

References

[1] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, A scalable content-addressable network. *SIGCOMM Comput. Commun. Rev.* 31, 4 (August 2001), 161-172, 2001.

[2] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan.. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev.* 31, 4 (August 2001), 149-160, 2001.

[3] Ning Qian and Guoxin Wu, Search Using Semantic Inference in Unstructured P2P Networks, *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on* , vol., no., pp.1-5, 10-12 Dec. 2010.

[4] Chunqiang Tang, Zhichen Xu, and Mallik Mahalingam. pSearch: information retrieval in structured overlays. *SIGCOMM Comput. Commun. Rev.* 33, 1 (January 2003), 89-94 , 2003.

[5] Hai Jin and Hanhua Chen. 2008. SemreX: Efficient search in a semantic overlay for literature retrieval. *Future Gener. Comput. Syst.* 24, 6 (June 2008), 475-488, 2008.

[6] Chuanyun Xu; Yang Zhang; , Ontology based P2P Semantic Search Routing Algorithm, *Networking, Sensing and Control (ICNSC 2010), International Conference on* , vol., no., pp.689-692, 10-12 April 2010.

[7] Nejdl, W., B. Wolf, C. Qu, S. Decker, M. Stintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch, Edutella: A P2Pa Networking Infrastructure Based on RDF, *in Proc. of the 11th International World Wide Web Conference (WWW 2002)*, Hawaii, USA, May 2002.

[8] Sungsu Kim; Strassner, J.; Hong, J.W.-K.; , Semantic overlay network for peer-to-peer hybrid information search and retrieval, *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on* , vol., no., pp.430-437, 23-27 May 2011

[9] M. W. Berry, Z. Drmac, and E. R. Jessup, "Matrices, vector spaces, and information retrieval," SIAM Review, vol. 41, pp. 335-62, 1999.

[10] Mei Li, Wang-Chien Lee, and Anand Sivasubramaniam. 2004. Semantic Small World: An Overlay Network for Peer-to-Peer Search. In *Proceedings of the 12th IEEE International Conference on Network Protocols* (ICNP '04). IEEE Computer Society, Washington, DC, USA, 228-238, 2004.

[11] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems. Technical report, Department of Computer Science, Stanford University, May 2004.

[12] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi, and Spyros Voulgaris. The Peersim simulator. http://peersim.sf.net

[13] Reuters Ontology http://trec.nist.gov/data/reuters/reuters.html

[14] Himali, D.M.R. and Prasad, S.K. , SPUN: A P2P Probabilistic Search Algorithm Based on Successful Paths in Unstructured Networks *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on* , May 2011

[15] The Gnutella home page, http://gnutella.wego.com/

[16] Yang, B. Garcia-Molina, H. , Improving search in peer-to-peer networks, *Distributed Computing Systems*, 2002. Proceedings. 22nd International Conference on , vol., no., pp. 5- 14, 2002

[17] Chuanyun Xu; Yang Zhang; , "Ontology based P2P Semantic Search Routing Algorithm," *Networking, Sensing and Control (ICNSC), 2010 International Conference on* , vol., no., pp.689-692, 10-12 April 2010

[18] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. 2002. Search and replication in unstructured peer-to-peer networks. *In Proceedings of the 16th international conference on Supercomputing (ICS '02)*. ACM, New York, NY, USA, 2002.

[19] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-yazti. A local search mechanism for peer-to-peer networks. *In Proceedings of the 11th ACM Conference on Information and Knowledge Management (ACM CIKM'02)*, 2002.

[20] Mashayekhi H., Habibi J. and Rostami H. Efficient Semantic Based Search in Unstructured Peer-to-Peer Networks. *In Proceedings of the second Asia International Conference on Modeling and Simulation (AICMS 08)* , vol., no., pp.71-76, 13-15 May 2008.

[21] Rostami H., Habibi J., Abolhassani H.. Amirkhani M. and Rahnama A. An ontology based local index in P2P networks. *Second International Conference on Semantics, Knowledge and Grid. (SKG '06) 2006.* pp.11, Nov. 2006.

[22] Juan Li and Son Vuong , OntSum: A Semantic Query Routing Scheme in P2P Networks Based on Concise Ontology Indexing. *21st International Conference on Advanced Information Networking and Applications (AINA '07) 2007,* vol., no., pp.94-101, 21-23 May 2007

[23] E. Frank, G.W. Paynter, I.H. Witten, C. Gutwin, and C.G.Nevill-Manning. "Domain-specific keyphrase extraction".*Proceedings of the Sixteenth International Joint Conference on Arti_cial Intelligence* (IJCAI-99), pages 668-673, 1999

[24] Michael Bendersky and W. Bruce Croft. 2008. Discovering key concepts in verbose queries. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (SIGIR '08). ACM, New York, NY, USA, 491-498, 2008.