SAS: Semantics Aware Search in P2P Networks

D M Rasanjalee Himali
Department of Computer Science
Georgia State University
Atlanta, Georgia USA
dmrhimali@student.gsu.edu

Shamkant B. Navathe
College of Computing
Georgia Institute of Technology
Atlanta, Georgia USA
sham@cc.gatech.edu

Sushil K Prasad
Department of Computer Science
Georgia State University
Atlanta, Georgia USA
sprasad@gsu.edu

***Abstract*—Resource management is very important yet challenging in large scale distributed systems like P2P networks. With more and more users incorporating semantic meta-data with their resources, the resource discovery mechanism should not only be able to scale well with the large number of information resources but also be capable of retrieving semantically relevant resources distributed in the P2P network with high accuracy and efficiency. In this paper we address these problems by proposing an ontology-based fully-decentralized peer clustering scheme where the network topology is optimized to perform efficient semantic query routing. The proposed semantic clustering scheme utilizes the structural relationships in ontology to organize peers into clusters based on the semantics of the resources they share. Performance evaluation demonstrates that our proposed approach can dramatically improve the search efficiency of unstructured P2P systems while keeping the communication cost at a comparable level compared with state-of-art unstructured P2P systems.**

***Index Terms*—**
Peer-to-peer search, semantic P2P networks, semantic clustering

I. Introduction

P2P systems have become a popular means of sharing large amounts of data among users over recent years. They are considered an attractive solution of applications requiring high scalability, robustness and autonomy. Efficient resource discovery in basic unstructured P2P systems, however, suffers from high costs mainly due to lack of global knowledge. To make matters worse, with the advent of semantic web, more and more users associate their shared resources with semantic meta-information. This mandates that the P2P networks provide methods that allow a user not only to describe resources from a semantic viewpoint but also allow users to run more complex queries addressing several semantic properties or relationships among semantic entities and resources.

In this paper we propose a semantic clustering and routing scheme which aims to improve the quality and efficiency of search in P2P systems. The basis for semantic clusters are concepts of an already agreed-upon shared semantic ontology. We use this semantic ontology with the view of improving information searching and facilitating interoperability. We propose a novel dynamic cluster construction technique where peers discover semantic neighbors by taking into account the structural relationships of concepts in the ontology. We also propose a novel query routing mechanism that exploits the concept hierarchy to quickly route a query to the target cluster. Both search traffic and gossiping are employed by peers to acquire global information regarding peer interests.

The rest of the paper is organized as follows: In Section II we discuss the related research in the area of P2P search. Section III presents the models and definitions. In Section IV we present our P2P system clustering architecture. Section V presents the cluster construction and maintenance and in section VI presents set of algorithms for ontology based search in unstructured P2P networks. In Section VII, we evaluate the proposed algorithms via simulation. Finally, Section VIII discuss the limitations and future roadmap of our work and concludes the paper.

II. Related Work

To improve the efficiency and quality of search, many semantic search systems have been proposed [3], [6], [10]. Semantic Overlay Networks (SON) [3] propose a peer clustering approach based on semantic content a peer shares. However, peer joining as well as query routing requires flooding. Interest based shortcut [11] model takes a similar approach where peers with similar interests create shortcuts with one another with the aim of efficient content location. However, these interest based shortcuts are built on top of Gnutella overlay imposing the burden of maintaining virtual links on top of the original network overlay. SETS [2] is a Vector Space Model (VSM) based approach for clustering. SETS however, relies on a single dedicated peer to cluster peers according to topic segments. GES [12] yet is a VSM based clustering approach which is a fully distributed cluster management mechanism. However, the node vector representation of GES may be inaccurate in presence of documents falling under multiple semantic categories, and the choice of long range links is inefficient. BF-SKIP [10],another VSM based clustering approach, employs s a combination of biased walk and flooding for topology adaptation. These VSM based approaches however, generally lack the capability to provide high quality semantics representation of nodes or documents. Ontsum [6] assumes heterogeneous ontologies between peers and requires computationally expensive ontology generation and mapping and allow only RDQL based queries.

III. Preliminaries

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. Each peer in the network is able to communicate with its direct neighbors (i.e. one hop neighborhood) only.

2) Global Semantic Ontology: A globally known reference ontology is agreed upon by all peers in the network. This ontology O consists of a set of semantic concepts $C = \{c_1, c_2, ..., c_m\}$ with total m concepts and relationships among them. For simplicity, we assume a semantic hierarchy where the relationships among concepts are only IS-A(i.e. hypernym/hyponym) relations.

3) Data Distribution: Representing knowledge using a reference ontology can be applied to any type of resource (e.g. text, video etc.) which is properly annotated. In this work we assume shared resources are text documents. There can exist multiple copies of the same documents distributed among peers. For each document a peer has in its local storage, it constructs a *concept weight vector* that depicts the semantic weight of each ontology concept present in that document. A weight $w_{i,j}$ of concept c_i in document d_j in peer P is calculated as follows: First, the concept frequencies of all concepts for each document in P's local storage are calculated by a process of text mining followed by word sense disambiguation. The concept frequencies of each document are added bottom up the semantic hierarchy to account for the contribution from each concept to its ancestor concepts in the hierarchy due to IS-A relationships they share. Then the concept frequencies of each concept are normalized to a [0-1] range by applying maximum frequency normalization by dividing them by the maximum concept frequency for that concept known so far by the peer.

4) Semantic Query: 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.

5) Semantic Similarity Calculation: We used the following well-known similarity measure *Sim* to calculate the similarity between two concepts c_1 and c_2 in the hierarchy [7]:

$$Sim(c_1, c_2) = \begin{cases} e^{\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & if c_1 \neq c_2 \\ 1 & otherwise \end{cases}$$

Where l is the shortest path distance between c_1 and c_2 in taxonomy and h is the depth of the least common subsumer of c_1 and c_2, α and β are parameters scaling the contribution of shortest path length l and depth h, respectively where the optimal values were defined as 0.2 and 0.6, respectively [7].

IV. System Architecture

Here we describe in detail our design of SAS that utilizes the synergy between ontologies and P2P systems to provide high quality search performance.

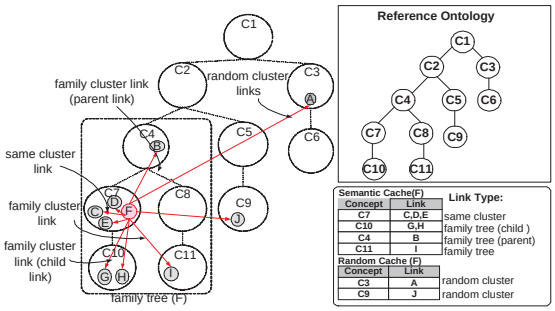


Fig. 1. SAS overlay

A. System Overview

Peers in a distributed network are rich in variety of semantic interests that are based on their sharable local document collections. Our clustering policy is based on the intuition that given a global ontology that describes the semantics of those shared documents, each peer's expertise can be described by a set of concepts in that ontology. This allows one to model different semantic relationships between peers using ontological relationships that exist between concepts in the ontology thereby physically organizing peers according to semantic clusters. More specifically, peers specializing in the same concept are physically grouped together as a cluster. A peer can specialize in zero or more concepts based on its shared document collection, thus resulting in the peer participating in zero or more concept clusters.

In our SAS framework, we designed different types of semantic relationships to model the semantic relationships between peers. A peer P can maintain a *same cluster link* to another peer for common concept they both specialize in. Similarly, P can maintain a *family tree link* to another peer that specializes in a concept that falls under the *family tree* of P. The family tree of a peer is the minimal sub-tree in the shared ontology semantic tree that subsumes all the concepts the peer is rich in. Any concept falling inside its family tree is considered a semantically close concept to its semantic expertise by a given peer. P can also maintain few random cluster links with some selected peers in the network. These would be peers for whom the concepts they specialize in fall out of the family tree of P and thus are considered semantically distant neighbors. We propose a scheme where these different types of relationships are effectively exploited to route queries to peers specializing in requested semantics, thus enabling efficient query routing.

Fig.1 shows a high level view of the network topology. The dotted circles and lines show the concept clusters and their IS-A relationships in the ontology respectively. While many links are maintained between peers, the different cluster connections of only peer F are shown for clarity assuming a simplified scenario where F belongs only to concept cluster $C7$.

B. Clustering Policy

Physically grouping peers to create peer clusters based on their semantic richness mandates a good clustering policy. In our work, we use ontology concepts as the basis for peer clustering. Since peers interests can be represented by its local data, the ontology metadata is used to extract semantics of shared documents and thus the conceptual interests of the peer. Peers rich in certain concepts in terms of both information content and the number of relevant documents for those concepts, categorize themselves as belonging to those concept clusters.

Each document in SAS is represented as a concept vector. These concept vectors contain log weighted concept frequencies *1+log(concept frequency)*. Compared to traditional *tf-idf* measure, log frequency weighting produces higher quality clusters [9] and does not require global information for computation. For a document to be considered relevant for a given concept c_i, the concept frequency of c_i in document should exceed a certain threshold. This threshold concept frequency for concept c_i at peer P is calculated as follows:

$$CFThredh_{i,P} = RelThresh \times MaxCF_i(P) where\ c_i \epsilon C(P)$$

Where $RelThresh$ is the system specified relevance threshold for a document which is a value in [0-1] range. $MaxCF_i(P)$, the maximum concept frequency known by P for concept c_i, is initially computed from P's local document collection and later on is kept updated by any new higher c_i values discovered from other documents in the network through message exchanges ensuring that eventually reaches its globally equivalent value with time. The calculation ensures P is represented by only those documents having large enough semantic weights for the given concept. This mechanism improves the precision of search over time by allowing a query to identify only those documents that are actually semantically rich and thus relevant for a queried concept. This precision improvement mechanism is briefly described in section V-B and more details can be found in our previous paper [4].

C. Types of Semantic Links

Peers maintain three types of semantic links with its neighbors. To illustrate these links consider Fig.1. The figure shows the shared ontology containing concepts $C1$ to $C11$. Let us assume a peer P is rich in concepts $C7$ and $C11$.

1) Same Cluster links: These links are established between peers who belong to the same concept cluster. For example, according to Fig.1, peer P can establish same cluster links with other peers in clusters $C7$ and $C11$.

2) Family Tree links: To achieve high semantic reachability, each peer maintains a relatively few number of links called family tree links which are selected among the member concepts in the family tree of the peer. A family tree of a peer P, $familyTree(P)$, is the sub-tree of the ontology rooted at the least common concept that subsumes semantic cluster concepts a peer P belongs to. This least common ancestor (lca) concept of $familyTree(P)$ is called the $lca(P)$. To ensure hierarchical connectivity of concept clusters based on semantic taxonomical relationships, a peer always maintains family tree links for its parent concept cluster and child concept clusters called *parent links* and *child links* respectively. These parent and child links essentially allow the network topology to be organized to mimic the global semantic hierarchy. Looking at the example given in Fig.1, $lca(P)$ is $C4$ as this is the least common ancestor that subsumes both $C7$ and $C11$. The $familyTree(P)$ is the sub-tree rooted under $C4$ and is marked in the figure. Any concept in $familyTree(P)$ excluding $C7$ and $C11$ are candidates for establishing family tree links.

3) Random Cluster Links: Establishing parent and child cluster links by each peer satisfies the minimum requirement of being able to navigate a query from any concept cluster to another by following semantic hierarchical structure. For faster access to semantically distant clusters, peers can also maintain a few random cluster links in its random neighbor cache to their semantically distant clusters. According to Fig.1, all the concepts in ontology outside family tree of P such as $C5$, $C6$, and $C6$ are perfect candidates for P to establish random cluster links.

D. Local Knowledge of a Peer

1) $MaxVector(P)$: The $MaxVector(P)$ is used for document re-normalizing purposes and holds global statistics of highest information content (calculated as concept frequency) for each concept in the ontology, for entire data set distributed in the network. This information is obtained gradually using message passing mechanisms.

2) $MyClusters(P)$): This is the set of concept clusters in which peer P categorizes itself as a member.

3) $SemanticNeighborCache(P)$: Semantic neighbor cache keeps track of different concept clusters the peer has discovered and regards as being semantically close to its semantic interests. An entry in the semantic cache takes the form $< c, neighbors >$ where c is the concept cluster for which the peer has established a connection with the given neighbors.

4) $RandomNeighborCache(P)$: A peer also keeps a second cache called random neighbor cache where it includes pointers to peers, whose clusters are semantically far from the clusters the peer belongs to. An entry in the random neighbor cache takes the form $< c, neighbors >$ where c is the concept cluster with which the peer has established a connection with the given neighbors.

Both caches are refreshed using Least Recently Used(LRU) policy. The maximum number of entries allowed in each cache including the maximum number of neighbors allowed per each cluster entry are system specified parameters.

V. Cluster Construction and Maintenance

A. Cluster Construction

We assume that the initial network is a pure unstructured power law network. At the configuration stage, each peer constructs document concept vectors for each of its document in the shared document collection. These document concept

vectors are used by peers to classify themselves into a set of concept clusters to which they belong.

1) Neighbor Discovery: To establish inter-cluster connectivity, it is crucial that each peer acquire cluster connections by neighbor discovery. We propose two methods to acquire cluster links: (i) a gossip based neighbor discovery which actively discovers new semantic peers and (ii) a passive query traffic based neighbor discovery algorithm.

(i)Gossip Based Neighbor Discovery: Gossip protocols are highly scalable and resilient communication protocols that are widely used to solve problems such as information dissemination, data aggregation etc. when the underlying network structure is inconvenient or extremely large. To implement gossip based neighbor discovery in a P2P network, each peer at each given fixed intervals of time, randomly chooses a peer from its neighborhood to exchange information about the concept clusters links they maintain to update their random neighbor cache and semantic neighbor caches with new links learnt. Due to the characteristics of gossip-based algorithms, it is guaranteed that every peer could establish cluster connections with every other concept cluster it is interested in with high probability in logarithmic steps of the size of the network.

(ii)Query Traffic Based Neighbor Discovery: Alternative to gossip based neighbor discovery, we propose a query traffic (i.e. query messages and query response messages) based neighbor discovery method. Here a peer's local knowledge is propagated along a query path by piggybacking its $MyClusters$ data structure and links maintained in its caches in query traffic messages in a passive fashion.

B. Cluster Maintenance

1) Cluster Merging: A peer initially starts by creating a concept cluster per each concept it belongs to by adding itself as the only member in the cluster. Understandably, this will create many clusters for the same concepts scattered in the P2P network. Over time, when peers acquire knowledge through gossiping or querying, they will also discover such new clusters for the same concepts they belong to. When this happens, the peer merges its cluster with the newly discovered cluster to form a larger cluster. To achieve this, each peer assigns a unique ID to its concept cluster at the configuration stage generated by a consistent hash function. When a peer merges its cluster with a new cluster, the lower ID of the two is assigned to the merged cluster.

2) Peer Joins: When a peer joins the network for the first time, it first connects to a random set of peers. The peer first classifies itself to a set of clusters based on its shared document collection and sends a join request to all its neighbors, piggybacking the set of clusters it belongs to. Upon receipt of a peer join request, an existing peer provides its existing semantic and random cache links and the set of clusters it belongs to, to the new peer. New peer then constructs its initial random cache and semantic cache links by using this information.

3) Peer Leaves and Failures: When leaving the network, a peer provides its semantic and random cache link information when it notifies its neighbors that it is leaving the network. This information is utilized by a notification receiving peer to update its semantic and random caches with appropriate links, removing soft state maintained for the leaving neighbor and also acquiring appropriate links from the links maintained by the neighbor leaving the network. If a neighbor crashes, peers eventually discover absence of the neighbor when query routing to that neighbor fails and eventually updates its soft state to remove that neighbor. Gossip and query traffic based neighbor discovery eventually reestablishes lost links.

4) Precision Improvement Mechanism (MaxVector Learning): Each peer keeps track of most recently known highest concept frequencies for each concept it specializes in, in a data structure called $MaxVector$. At configuration stage, a peer initializes its $MaxVector$ using its local document collection. Utilizing piggybacked $MaxVector$ information in the message a peer receives over time, it discover and updates its $MaxVector$ accordingly to reflect highest known concept frequencies in the network so far. This in turn results in peers re-normalizing its local document concept vectors making their semantic representations more accurate, thus increasing query precision over time.

VI. Query routing strategy

The goal of query routing strategy is to locate as many relevant documents as possible. When a query contains a conjunction of multiple concepts, the querying peer constructs a sub-query per queried concept by including the same message id as the original query. The basic idea is to propagate the concept based sub-queries to relevant target clusters, where the original query will be evaluated. The query is propagated until a system specified TTL is reached. The results will return in the reverse path of the query propagation. The query originator then collects and returns the merged results to the end user upon receipt of responses to all sub-queries. Below we describe the inter-cluster routing mechanism that propagates the query to the target cluster and the intra-cluster routing mechanism which takes care of forwarding the sub-query to all cluster neighbors with high probability.

A. Intra-cluster routing

When a peer receives a query targeted at a concept cluster that a peer belongs to, the query enters the intra-cluster routing mode. Here, the peer simply broadcasts the query through its same cluster links to exhaustively propagate the query within the concept cluster. If the peer does not have any same-cluster-links established yet, it chooses a family-tree-link semantically closest to the target concept cluster, preferably a parent link or a child link to forward the query as those will have the highest probability to maintain a direct link to the target concept cluster

B. Inter-cluster routing

If one of the concept clusters the peer belongs to cannot satisfy the query, peers then check if the target concept falls within its family tree. For this to happen, the queried concept

should be a member of the family tree of the peer and the peer must have a direct or semantically close family tree link in its semantic cache for the target concept cluster. In this case, the query will be forwarded to that concept cluster. If the queried concept falls outside of its family tree, the peer will dispatch the query to the random-cluster-link semantically closest to queried concept. In situations where same-cluster links or family-tree-links are not obtainable, the peer resorts to random-cluster-links to forward a query to a randomly selected cluster.

When the network eventually reaches a steady state, every peer P will establish a minimum of one parent link and child link per concept in $Myclusters(P)$. Therefore, these parent links and child links provide a guaranteed path to navigate from one concept cluster to another by mimicking IS-A links in the ontology. Additional family tree links and random cluster links allow faster location of the target cluster regardless of its semantic proximity to the peer.

VII. Experiments

A. Design of Experiments

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

1) Ontology: The Reuters21578 text classification corpus [5] was used with the category Country as the basis for constructing the semantic taxonomy. We then used the hypernym/hyponym relations of Wordnet ontology to further extend this classification by adding descendant 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 235 concepts.

2) Data Generation: We used the documents from the Reuters21578 dataset in our simulation experiments. There were 21,578 newswire documents in the dataset and after text processing and words-sense disambiguation, a total of 15,191 documents that produced non-zero length concept frequency vectors were selected.

3) Network Generation: We implement our system on top of Peersim simulator [8]. The initial network generated follows the power law topology. The default network size was set to 1024 nodes. The initial average peer degree was set to 5. The document distribution among peers follows a Zipf ($\alpha = 1.0$) distribution and each peer contained 100 documents on an average in its local storage.The TTL varied from 2 to 4 and the default was set to 2. The dynamic behaviour was simulated by inserting 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.

4) Query Generation: We generated 100 random queries each for single and two concept queries from the concepts in the ontology. On average, every peer issues 50 queries during its lifetime randomly selected from generated queries.
In addition to this, the maximum entries allowed per random cluster cache and semantic cluster cache are set to 5 and 100 respectively. The maximum number of neighbors allowed per cluster entry is 5,5 and 10 respectively for random cluster links, familytree cluster links and same cluster links. A combination of gossip and query-traffic based neighbor discovery is used for neighbor discovery mechanism.

5) Comparison Algorithms: We compare the performance of our algorithm with the state-of-the-art algorithm BF-SKIP [10] and Gnutella [1] based search.

a) BF-SKIP: Authors of [10] introduce BF-SKIP, a semantic clustering algorithm based on the VSM model. Given a query, BF-SKIP first relies on biased walks through random links to locate a relevant semantic group, then uses flooding within this group through semantic links to retrieve relevant documents in only one hop. Once in the target cluster, the number of flooded messages in query is iteratively reduced at each hop to reduce unnecessary message production. We set the maximum links to 8 and relevance threshold to 0.7. The iteration depth k is set from 1 to TTL.

b) Gnutella: Gnutella flooding [1] is the most fundamental blind search mechanism where a querying peer floods a query within a TTL hop radius.

B. Performance Metrics

For our evaluation we rely on three major retrieval performance 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.

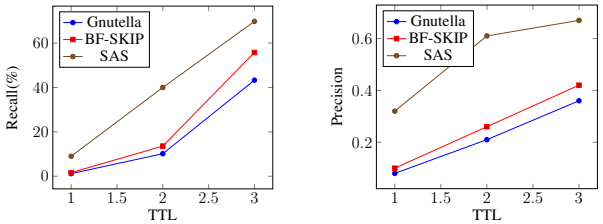
2) Precision: Precision is the fraction of documents among those retrieved that are relevant to a search query.

3) Message Cost: This is the average number of bytes transferred per search query.

C. Results and Analysis

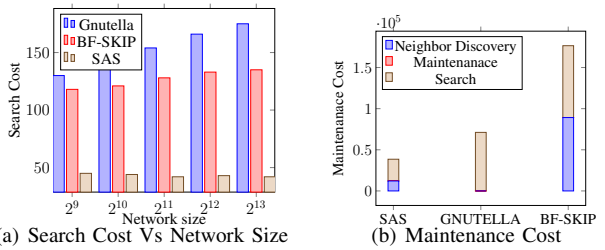
In this section, we discuss our results for the above P2P environment. We measured performance of the search algorithms for various network sizes, and results show that performance of SAS is scalable regardless of the network size. Experiments were carried out for both single concept queries and two concept conjunction queries. The results were averaged over simulation of five network seeds. Due to space limitation, we report only partial results. Results are reported after network convergence based on clustering protocol. Our simulations also show that our SAS algorithm improves recall and precision significantly over BF-SKIP as well as Gnutella flooding counterparts at much lower message cost and produces high quality clusters compared to BF-SKIP.

1) Query Efficiency: Fig. 2(a) illustrates the relationship between the query recall and the TTL. The experiments were conducted on a default 1024 nodes network while varying the TTL from 1 to 3. According to Fig. 2(a), SAS achieves a high recall rate with small TTL. While our SAS protocol



(a) Recall Vs TTL (b) Precision Vs TTL

Fig. 2. Search Efficiency: (a) the recall and (b) the precision



(a) Search Cost Vs Network Size (b) Maintenance Cost

Fig. 3. Message Overhead

achieves 9.00%, 42.01% and 72.12% recall rates for TTL values 1, 2, and 3, respectively, BF-SKIP achieved only 1.64%, 13.58% and 55.77% recall and Gnutella achieved only 1.18%, 10.15% and 43.28% for the respective TTL values. The reason behind this is that SAS can locate target clusters much faster thereby resolving the query quickly. This indicates that a greater portion of the number of hops travelled by a query is spent within the target cluster flooding semantically relevant nodes thus proving the effectiveness of our link establishment strategy within clusters and between clusters. Fig. 2(b) illustrates the relationship between precision and TTL. The experiments were conducted on a default 1024 nodes network and results were averaged over five networks. The error margin reported was $\pm$ 0.05. According to figure 2(b), SAS achieves high precision values compared to BF-SKIP and Gnutella due to the MaxVector learning employed in SAS.

2) Search and Maintenance Cost: Fig. 3(a) show that our SAS protocol sends significantly fewer search messages than Gnutella and BF-SKIP to resolve a query. This is because when given a request, SAS can efficiently locate the target cluster rapidly, so that the search space is reduced and queries get more results with certain TTL. It also shows that our algorithm clearly outperforms BF-SKIP and Gnutella even for large network sizes. While a 70.23% average improvement of search message cost was observed over BF-SKIP, a 99.44% improvement was observed over Gnutella. Fig. 3(b) depicts this message overhead for a network of size 1024 nodes with search TTL set to 2. As shown in the figure, SAS generates the least search cost for search queries and neighbor discovery queries. The maintenance messages generated are too few to be visible in the figure. The production of a few neighbor discovery messages is a result of using the combination of search query traffic and gossip messages for neighbor discovery. Therefore greater portion of neighbors can be discovered with zero cost just by utilizing search traffic thus lowering the additional overhead of generating gossip messages.

VIII. Conclusion

In this paper we presented an ontology-based clustering and routing protocol that optimizes search performance of unstructured P2P networks by ontology-aware topology construction. Our scheme organizes the overlay structure based on semantic concepts and their taxonomical links in the shared ontology. In addition to these taxonomical links, a peer also establishes other types of links to ensure faster location of target clusters and proper dissemination of a query to only relevant peers within a target cluster while querying. SAS employs a combination of search messages and gossip to discover semantically relevant neighbors. In the future development of the system, we plan to address more effective dynamic cluster construction techniques, and use of richer ontologies which include relationships beyond simple classification hierarchies.

References

[1] Gnutella home page. http://gnutella.wego.com/.

[2] Mayank Bawa, Gurmeet Singh Manku, and Prabhakar Raghavan. Sets: search enhanced by topic segmentation. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 306–313, New York, NY, USA, 2003. ACM.

[3] Arturo Crespo and Hector Garcia-Molina. Semantic overlay networks for p2p systems. In *Proceedings of the Third international conference on Agents and Peer-to-Peer Computing*, AP2PC'04, pages 1–13, Berlin, Heidelberg, 2005. Springer-Verlag.

[4] Rasanjalee Dissanayaka, S.B. Navathe, and Sushil K. Prasad. Osqr: A framework for ontology-based semantic query routing in unstructured p2p networks. In *Proceedings of international IEEE HiPC conference on High Performance Computing*, HiPC '12. IEEE, 2012.

[5] D. D Lewis. Reuters Dataset. http://trec.nist.gov/data/reuters/reuters.html, 2004.

[6] Juan Li and Son Vuong. Ontsum: A semantic query routing scheme in p2p networks based on concise ontology indexing. In *Proceedings of the 21st International Conference on Advanced Networking and Applications*, AINA '07, pages 94–101, Washington, DC, USA, 2007. IEEE Computer Society.

[7] Y. Li, Z.A. Bandar, and D. McLean. An approach for measuring semantic similarity between words using multiple information sources. *Knowledge and Data Engineering, IEEE Transactions on*, 15(4):871 – 882, july-aug. 2003.

[8] Alberto Montresor and Márk Jelasity. PeerSim: A scalable P2P simulator. In *Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09)*, 2009.

[9] Hinrich Schütze and Craig Silverstein. Projections for efficient document clustering. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '97, pages 74–81, New York, NY, USA, 1997. ACM.

[10] Wenwu Shen, Sen Su, Kai Shuang, Fangchun Yang, and Jingshu Xia. An efficient search mechanism in unstructured p2p networks based on semantic group. In *Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology*, CIT '10, pages 2982–2986, Washington, DC, USA, 2010. IEEE Computer Society.

[11] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 2166 – 2176 vol.3, march-3 april 2003.

[12] Yingwu Zhu and Yiming Hu. Enhancing search performance on gnutella-like p2p systems. *IEEE Trans. Parallel Distrib. Syst.*, 17(12):1482–1495, December 2006.