OntSum: A Semantic Query Routing Scheme in P2P Networks Based on Concise Ontology Indexing

Juan Li, Son Vuong
Computer Science Department, University of British Columbia
{juanli, vuong}@cs.ubc.ca

Abstract

Locating desirable resources is very important for a large distributed system. However, the distributed, heterogeneous, and unstructured nature of the system makes this issue very challenging. The discovering mechanism has to be not only semantically rich, in order to cope with complex queries, but also scalable to handle large numbers of information sources. In this paper, we address these problems by proposing OntSum, an efficient peer-to-peer query routing scheme based on concise ontology indexing. Unlike most existing systems, our system does not assume a global ontology but heterogeneous ontologies. Peers in the system use their own ontologies to describe their resource knowledge and the network topology is adjusted according to peers' ontological properties. A novel indexing strategy enables forwarding queries only to semantically related nodes. The architecture improves interoperability among network participants and aids efficient resource discovery through an expressive query language.

1. Introduction

With the rapid growth of online data, discovering the desirable information from the vast amount of sources has become a central issue to be addressed. Peer-to-peer (P2P) technology has been used as a solution to this problem, since it scales to very large networks, while ensuring high autonomy and fault-tolerance. However, existing P2P systems offer few data management facilities, limited to IR(Information Retrieval) -style keyword search. Keyword search is appropriate for simple file-sharing applications, but it is unable to deal with discovery of complex resources which have various properties and sophisticated relations with each other.

Recent schema-based P2P systems [1, 2, 3] go beyond file-sharing, by providing infrastructures where peers can create and share knowledge. Most existing schema-based P2P systems are built on top of unstructured networks, and often use flooding or maintain a broadcast structure such as a tree or a super cube for searching. For example, to execute an RDF query, Edutella [3] broadcasts the query to the whole hypercube. More recently, a few studies [4, 5, 6] extended the DHT to support complex queries. The basic idea is to map each keyword in a complex query to a key. A query with multiple keywords then uses the DHT to lookup each keyword and returns the intersection. In order to do this intersection, however, large amounts of data have to be transferred from one peer to another, thus creating large traffic load [7]. Systems like [8] avoid this multiple lookup and intersection by storing a complete keyword list of an object on each node. This approach may incur a huge overhead on publishing and storing the keywords. Another problem of DHT approaches is that most of them assume a uniform ontology .

In this paper, we address the routing issues of expressive queries in a large distributed environment by an ontology-based P2P approach. Peers in our system advertise local resources using ontological descriptions. We do not assume peers use a global ontological schema; instead, ontologies are allowed to differ between users. The network topology is reconfigured with respect to peers' ontological similarity, so that peers with similar ontologies are close to each other. An effective and efficient lookup service is built on top of a concise ontology indexing, such that queries can be forwarded only to those peers containing resources that satisfy the query constraints. Results from simulation experiments demonstrate that this routing scheme is scalable and efficient.

2. Peer ontology

Before they can discover resources efficiently, peers have to construct their ontological knowledge of the resources. With this knowledge, peers can describe their resources expressively, pose rich queries, and

21st International Conference on Advanced Networking and Applications(AINA'07)
0-7695-2846-5/07 $20.00 © 2007 IEEE

route queries intelligently. In this section, we specify how nodes construct their ontology knowledge in detail.

2.1. Local ontology repository

Peers describe their resource knowledge with ontologies. The knowledge sharing and discovering are based on the ontological descriptions. As in common DL (Description Logic) systems [26], our system divides the ontology knowledge into two parts: the taxonomical box (T-Box) and the assertion box (A-Box). The T-Box stores conceptual knowledge and can be compared to the schema of a database system. It is created by defining concepts and relationships between concepts. The A-Box represents the concrete knowledge about individuals. A node also uses inference engines to derive additional facts from existing knowledge. Our inter-cluster routing and intra-cluster routing, both will be explained later, are based on the T-Box and A-Box indexing respectively.

2.2. Peer semantic similarity

To organize peers according to their semantic properties, we need a metric to measure peers' ontology similarity. There have been extensive researches ([28], [29], [30]) focusing on measuring the semantic similarity between two objects in the field of information retrieval and information integration. In this paper, we will use a very simple method to compute the semantic similarity function between two peers; this can easily be replaced with other advanced functions for a complex system.

In the current system, a node's T-Box concepts are indexed into a vector which is called this node's *ontological signature vector*. This vector is extended by adding new concepts which are semantically related to the original concepts. For example, we can exploit a pre-defined thesaurus like WordNet [31] or knowledge learned from the network. Assume A and B are two peers, and their signature vectors are $V(A)$ and $V(B)$ respectively. The semantic similarity between peer A and peer B is defined as:

$$sim(A,B) = \frac{|V(A) \bigcap V(B)|}{|V(A)|} \qquad (1)$$

In (1), $\cap$ denotes set intersection, while $|\ |$ represents set cardinality. This definition is based on the idea that ontologies which share more common concepts are more similar than those sharing less common concepts. Note that with this definition, similarity is not a symmetric relation (i.e.: $sim(A,B) \neq sim(B,A)$). Peers become semantically related to a peer if their similarity is beyond a specified *similarity threshold: t (0<t<1)*, which can be determined locally by the peer.

2.3. Ontology mapping

As mentioned, it is infeasible to expect a standard uniform ontology inside a large network; instead, different peers are allowed to use different ontologies. When a query is forwarded to a peer, even if the peer is semantically related to the query, it may not know all terms used in the query expression, because they are taken from the local ontology of the asking peer. In order to overcome this problem, we have to align the ontologies of the asking and the answering peer. We have described an appropriate mapping scheme in [9]; here we briefly present the basic idea: The defined mappings between different ontologies either refer to the same concept, relation (*equivalent class/property*), or one is a special or general case of the other (*sub/super class/property*). We also note that various ontologies may contain different supplementary information about the same real world individual; thus we add a special *referentialClass* relation between concepts. This allows individuals to be merged if specific properties match, creating an aggregated entity.

2.4. Ontological query

The system adopts RDQL [10] as the query language. RDQL is a query language in Jena [11] models, which is based on matching {subject, predicate, object} triples. A query is generally constructed by using the user's local ontology; yet, in order to retrieve relevant data from other ontologies, it is extended and reformulated with the inter-ontology mappings.

3. System architecture

In this section, we explain how to construct an ontology-aware network topology and how to route queries intelligently in a large distributed system. In addition, the problem of semantic heterogeneity is taken into account as well.

3.1. Overview

3.1.1. Semantic domains and clusters. Figure 1 shows a high level picture of the network topology. Nodes form multi-layered clusters reflecting the semantic locality: nodes with similar ontological topics form a big domain; inside the domain, nodes may create smaller clusters if they share the same ontology.

For example, in Figure 1, all peers in the medical domain are interested in medically related information. They may be interested in different aspects of the medical resources, and they may use different ontologies to describe their resources, but since they share the similar interests (medicine here), they connect with each other through some links. Inside the medical domain, nodes further organize themselves to finer-grained clusters based on their ontologies. For example, *N1, N2, N5*, and *N8* use the same ontology: *ont1* (e.g., a medical ontology: SNOMED-RT [27]), so they form a same-ontology cluster. In the rest of this paper, we use the term "*domain"* to represent a group of clusters sharing similar ontological topics, and use the term "*cluster*" to denote the same-ontology cluster. Clusters and domains do not have fixed boundaries; they are formed by randomly connecting relevant nodes.

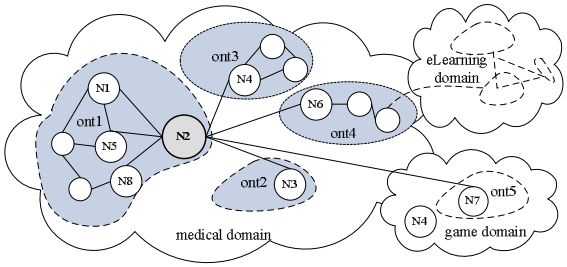


Figure 1. The network topology

To form this multi-level structured network a node distinguishes three kinds of neighbors based on their semantic similarity. A peer *A*'s neighbor *B*, can be one of these three types: (1) zero-distance neighbor (or same-ontology neighbor, intra-cluster neighbor), if *sim(A,B)=1*. (2) short-distance neighbor (or semantically related neighbor) if $sim(A,B) \geq t$ ($0<t<1$ is *A*'s semantic threshold.). (3) Long-distance neighbor (or semantically unrelated neighbor) if $sim(A,B)<t$. A node always tries to find as many close neighbors as possible, but it also keeps some long distance neighbors to reach out to other ontological clusters.

3.1.2. Query and routing. Peers in our system may pose two kinds of queries: *neighbor-discovery query* and *resource-discovery query*. The neighbor-discovery query is used to construct the ontology-based network topology. When a new node joins the network, it issues neighbor-discovery query to find semantically related neighbors, so that it can join their domain and cluster by connecting to them. The resource-discovery query is to locate desirable resources in the network.

To efficiently route queries, we propose two routing schemes: inter-cluster routing and intra-cluster routing. The former quickly locates semantically related clusters; while the latter efficiently finds desirable resources satisfying the query constraints. Related with the two routing schemes, two routing tables are maintained at each node: inter-cluster routing table and intra-cluster routing table (inter-table and intra-table for short). A node's routing tables maintain finer-grained knowledge of neighbors semantically closer to it, but coarser-grained knowledge of neighbors further from it. This reflects the characteristic of our routing strategy: the query first walks around the network, once it reaches the target domain, it zooms in on that domain and investigates the domain's ontology properties.

3.2. Routing tables

3.2.1. Inter-cluster routing table. A node's inter-table stores the abstract semantic knowledge of its neighboring clusters. Specifically, it keeps the contacts to those clusters: its short-distance and long-distance neighbors, their semantic similarities to this node, their semantic signature vectors mapped in a compressed Bloom Filter [15]. To reconcile the semantic differences between the node and its short-distance neighbors, inter-ontology mappings are also stored in the inter-table. A query then can be forwarded to a neighbor after being translated according to the inter-ontology mapping. A neighbor-discovery query is mainly routed through the long distance links to quickly locate related clusters. A resource-discovery query is forwarded only through the short-distance links because of the topology's semantic locality property.

Table 1 shows the inter-table of *N2*, a node in Figure 1. *N3, N4*, and *N6* are short-distance neighbors of *N2* (assume the similarity threshold is *0.6*). *N7* is a long-distance neighbor which links *N2* to a semantically unrelated domain. Neighbors' semantic signature vectors are compressed into a Bloom filter, thus they are sequences of 0s and 1s. The last column of the table stores the inter-ontology mappings between *N2* and other semantically related neighbors. For example, the last column of the first row stores ontology mappings between *N2* and *N3*, which includes equivalent class mapping *Ca=Ca'* and equivalent property mapping *P1=P1'*. The representation of the mapping is defined in our previous paper [9]. Several semi-automatic tools such as Protégé-PROMPT [12] and Chimaera [13] exist for eased ontology mapping.

To control the overhead of routing table maintenance, a soft-state update mechanism is used to keep the routing information up-to-date: nodes periodically probe their neighbors and propagate

updated ontology information to them. At any given time, the resource routing information may potentially be stale or inconsistent, but in a long run, they are good enough to direct the query forwarding to the right peers.

Table 1. Inter-cluster routing table of node N2.

neighbor	semantic similarity	compressed signature vector	Inter-ontology mappings
N3	0.8	ont2 [1001010…]	*Ca=Ca',P1=P1'...*
N4	0.7	ont3 [0111010…]	*Cm⊃Cm',P2⊃P2'*
N6	0.6	ont4 [1100010…]	*Ct⊂Ct' ...*
N7	0	ont5 [0001010…]	none

3.2.2. Intra-cluster routing table. After the target cluster has been located through the inter-cluster routing, intra-cluster routing table is used to forward queries inside the cluster. Since clusters are relatively small, it is possible to index more detailed ontology information into the intra-table. Unlike inter-table which stores the abstract T-box knowledge, the intra-table records detailed A-Box knowledge from neighbors inside the same cluster, i.e., zero-distance neighbors. Table 2 is the intra-cluster routing table of node *N2*. It includes a concise summary of individual resources which can be reached through *N2*'s zero-distance neighbors: *N1, N5 and N8*. The resource information is summarized into a concise structure called *triple filter*, which will be explained in detail in Section 3.5.

Table 2. Intra-cluster routing table of node N2.

neighbor	A-Box summary in the format of triple filters
N1	sub:[01030212..], pre:[2100010..], obj:[31312021..]
N5	sub:[00120212..], pre:[1103013..], obj:[02212010..]
N8	sub:[01002110..], pre:[0100012..], obj:[01132010..]

3.3. Neighbor-discovery query routing

The construction of ontology-based topology is a process of finding semantically related neighbors. A node joins the network by connecting to one or more bootstrapping neighbors. Then the joining node issues a neighbor-discovery query, and forwards the query to the network through its bootstrapping neighbors. The neighbor-discovery query is routed mainly according to the inter-cluster routing table.

A neighbor-discovery query message includes several parts: (1) The querying node's ontology signature vector. To reduce its size, and to accelerate ontology matching, the signature vector is compressed into a Bloom filter bit map. (2) A similarity threshold which is a criteria to determine if a node is semantically related to the query. (3) A query TTL to decide how far the query should be propagated. (4) A list of clusters (represented by the ontology namespace of the cluster) the query has passed through, so that the query would not be forwarded to the same cluster again and again.

When a node N receives a neighbor-discovery query Q which tries to find neighbors for a new joining node X, N computes the semantic similarity between X and itself; if N is semantically related to X, N will send a Neighbor Found reply. If the query's TTL does not expire, N computes the semantic similarity between X and each of its neighbors, and forwards the query to semantically related neighbors. If no semantically related neighbors are found, the query will be forwarded to N's long-distance neighbors. The detailed query processing algorithm is illustrated in Figure 2.

```
/* When a node N receives a neighbor-discovery query Q
issued by a new joining node X, N calls this function to process
the query*/

process_neighbor_discovery_query (query Q)
{
1.  if Q has been received before, discard it, return
2.  compute the semantic similarity between X and N: sim(X,N)
3.  if (sim (X,N) =1)
4.      send a reply indicating N is X's zero-distance neighbor;
        the reply also contains N's zero-distance neighbors
5.  if (threshold ≤sim(X,N) < 1)
6.      send a reply indicating N is X's short-distance neighbor
7.  if (TTL does not expire)
8.      for each neighbor Nj in N's inter-cluster table
9.          compute the semantic similarity sim(X, Nj)
10.         if (sim(X, Nj) ≥ threashold)
11.             forward Q to Nj
12.     if no Nj found
13.         forward Q to N's long distance neighbors
}
```

Figure 2. Pseudo code of neighbor-discovery query

3.4. Resource-discovery query routing

With the semantics-based network topology and the inter-cluster and intra-cluster routing tables, queries can be efficiently forwarded to only a small set of related peers. When a peer initiates a query, it first chooses a subset of zero-distance neighbors to forward the query. Since they use the same ontology, the zero-distance neighbors are best candidates to forward the query to. The selection of neighbors is based on the intra-cluster routing algorithm, which we will describe in Section 3.5. The query is also translated according to the inter-ontology mappings, and forwarded to related clusters through the short-distance neighbors whose semantic signature vectors match the translated query. Then the query is propagated in those clusters. When a node receives a query, it uses similar strategy to

forward the query to their neighbors until the query TTL expires.

3.5. Intra-cluster routing

The objective of intra-cluster routing is to find the right individual resources which satisfy the query constraints. The intra-cluster routing is guided by peer's compact A-box summaries. When processing a query, the summaries are used in a pre-processing step to filter the members that are likely to provide a relevant answer to the query. Here we will give a brief introduction of the routing scheme; the detailed routing algorithm was explained in our previous paper [14].

3.5.1. Idea of triple filter. The individual-level (or instance-level) indexing can be very expensive due to the large number of instances. Our solution is to extract the building block of the RDF A-Box statement: *subject, predicate* and *object*, and summarize them in a compact structure: a triple filter, which is based on Bloom filters. A triple filter includes three Bloom filters: the *subject filter*, the *predicate filter*, and the *object filter*. An RDF triple can be hashed to these three filters. With this technique, the network traffic can be significantly reduced since instead of transferring all data through the network, only an aggregated bit map needs to be transferred.

3.5.2. RDV intra-cluster routing. The query routing inside a cluster is based on our resource-distance-vector (RDV) routing algorithm [14]. It uses a distance vector approach to route the query to the nearest matching nodes. Every peer maintains a resource index table. This table uses the triple filters we mentioned above, and includes distance information (in number of hops). Peers exchange the resource indices with their neighbors, and update relevant entries in their table. The distance information is updated whenever passing through a node. To reduce false positives brought by the result of resource information aggregation, we set a hop count limit, which we call radius, to limit the number of hops the resource information can travel. When a node receives a query request, the algorithm chooses the shortest routes to forward the query. In addition, a "heuristic jump" method is used to expedite the searching process by skipping over the "barren" areas.

Figure 3 illustrates a query routing example. In this example, the *radius* is set to *3*, so nodes are only aware of resources within 3 hops. Node *A* receives a query for resource *e*, which is mapped to two positions: *3* and *6* in the filter. It checks its routing table and finds two matches: through *C* with *2* hops *(C_3=2, C_6=2)* and through *D* with *3* hops *(D_3=3, D_6=3)*. So the shortest distance to the resource is *2* through neighbor *C*. Therefore, the query is forwarded to *C*. Similarly, *C* forwards the query to *E*. *E* finds a match in its local vector, and then it checks the RDF database with the original RDQL query. Readers can refer to our paper [14] for detailed explanation of the construction, maintenance, and usage of the routing table.

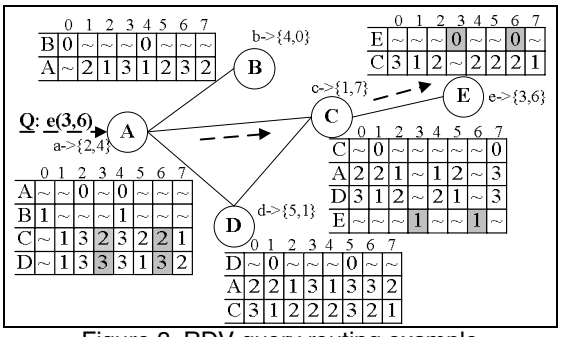


Figure 3. RDV query routing example

4. Experiments

4.1. Methodology

Extensive simulations have been preformed to evaluate the performance of our searching scheme. For comparisons, we simulated our searching scheme OntSum in conjunction with the learning-based short-cut scheme [23] and a randomly connected Gnutella scheme [33]. The short-cut approach was chosen as one comparison reference since it is simple yet effective, and many popular applications (e.g., [23], [32], [24], [25]) use this approach as their routing scheme. Moreover, it is comparable to our approach in a sense that it creates clusters on top of the unstructured network. Similarly, we used the Gnutella searching as another reference approach for its simplicity and prevalence, which, in fact, made it a widely used benchmark approach for many researches.

In the simulation, the semantic data follows a Zipf (α=1) distribution between different peers. The simulation is initialized by injecting nodes one by one into the network until a certain network size has reached. After that, a mixture of join, leave and query are injected into the network based on certain ratios. The proportion of join to leave operations is kept the same to maintain the network at approximately the same size. On the average, each peer issues 50 queries during each run of the simulation. One assumption of our simulation is that a node only issues queries with local ontologies. The resource-discovery query is propagated exponentially, i.e.: each node chooses a certain number of neighbors (called *walkers*) to forward the query. The neighbor-discovery query is

propagated linearly, i.e.: only the node that issues the query forwards the query to a certain number of walkers, while all other nodes only forward the query to one neighbor. In the rest of the paper, we use the term "*query*" to refer to "*resource-discovery query*". The simulation parameters and their default values are given in Table 3.

Table 3. Parameters used in the simulations

Parameter	Range and default value
network size	2^{10}~2^{15} default: 10,000
initial neighbors (node degree)	5
avg node degree	14
TTL	6
resource-discovery query walkers	3 (propagate exponentially)
neighbor-discovery query walkers	2 (propagate lineally)
ontology categories	1~41 default: 20
ontology schemas per domain	4
distinct resources per domain	100
Triple filter radius	2
query possibility per time slice	20%~35% default:20%
churn possibility per time slice	5%~20% default:5%
neighbor probe possibility per time slice	5%~20% default:10%

4.2. Results

In this part, we present the experimental results which demonstrate the performance of our routing algorithm.

4.2.1. Lookup efficiency. To evaluate the performance of our semantic search, we compare our algorithm with the other two algorithms in the metric of recall rate, which is defined as the number of results returned divided by the number of results actually available in the network. In terms of scalability to network size, we vary the number of nodes from 2^9 to 2^{15}.

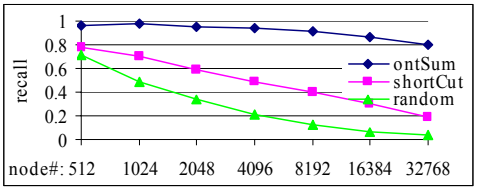


Figure 4. Recall vs. network size

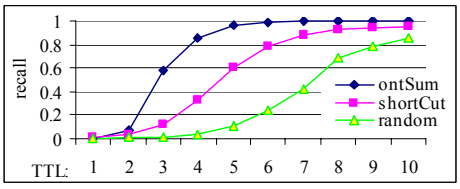
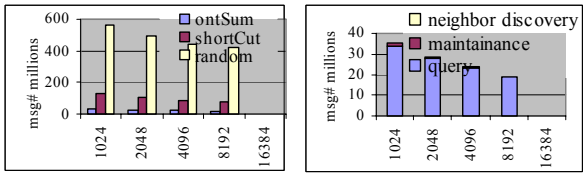


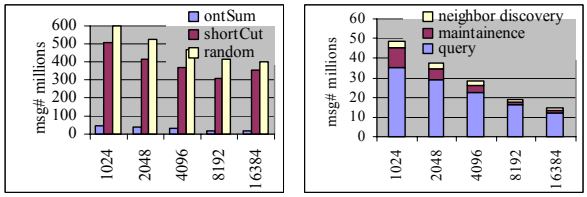
Figure 5. Recall vs. TTL (with walker#=5)

Figure 4 compares the recall rate of the three routing strategies in different sized networks. OntSum dramatically outperforms the other two algorithms. Short-cut and Random's recall rates decrease substantially as the network size increases, while OntSum achieves a very high recall rate even when the network size is very large. Figure 5 illustrates the relationship between query recall rate and query TTL when the number of walkers are set to 5. OntSum achieves a high recall rate with a small TTL, which means OntSum can resolve the query much faster. This is because OntSum searches the query only in its local domain; in addition, because of the RDV routing, OntSum always forwards the query to the right peers in a cluster.

4.2.2. Searching and maintenance cost. Figure 6 depicts the overhead in terms of the number of messages per hit under different query and churn frequencies.



(a) In each time slice a node has a 35% possibility to issue a query, a 5% possibility to check neighbors' update, and a 5% possibility to join/leave.



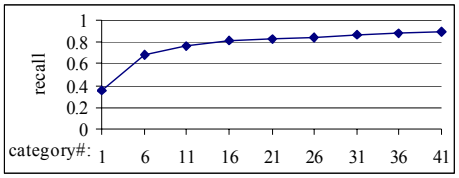
(b) In each time slice a node has a 20% possibility to issue a query, a 20% possibility to check neighbors' update, and a 20% possibility to join/leave.

Figure 6. Message overhead per query hit

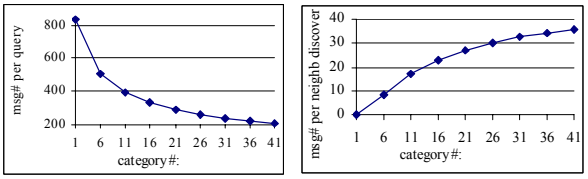
Experiments in Figure 6(a) are configured to have higher query frequency and lower churn frequency, compared with experiments in Figure 6(b). We can see that OntSum produces much fewer messages in both configurations. The overhead of OntSum includes resource-discovery query, neighbor-discovery query, and routing table update; while the other two algorithms only create resource-discovery query. We can see from the right sub-figures in Figure 6 (a) and 6 (b) that the proportion of each part of the OntSum overhead is affected by the ratio of query frequency, node join/leave frequency, and routing table update frequency. When the churn rate and routing table

update frequency are high, the overhead created by routing table update accounts more in the total message overhead. Neighbor-discovery does not create many messages at all; the messages are too few to be visible in the figures.

4.2.3. Effect of clustering. In our system, the node joining process configures the network topology with respect to ontology categories. The effect of the number of ontological categories on the performance of OntSum algorithm is given in Figure 7. The number of category is increased from 1 to 41 for a network of size 10,000. In Figure 7 (a), the recall increases as the number of category grows. This is easy to understand: more domains and clusters are created when the category number increases; therefore, searching has a smaller space to explore. From Figure 7 (b), we can see that the average message cost per query decreases, but the cost of neighbor-discovery query increases when the number of categories increases.



(a) recall vs. number of categories



(b) Relation of query cost, neighbor-discovery cost and the number of semantic categories

Figure 7. Effects of clustering

4.2.4. Sensitivity to dynamics. Dynamics is common in large distributed systems. Thus a robust system needs to be resilient to the dynamics. Dynamics can affect both the routing table's freshness and the underlying connectivity. To evaluate the adaptability to different levels of dynamics, we measure the system performance under different level of peer churn rate.

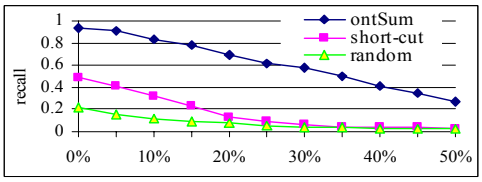


Figure 8. Recall vs. peer churn possibility per time slice

Figure 8 shows the relation of recall and the peer churn rate in a 5000-node network. When peers join or leave/die frequently, performances of short-cut and OntSum deteriorate, because the former loses its short-cuts, and the latter loses its routing state freshness. On the other hand, we see OntSum has a good tolerance of the system dynamics. It achieves good recall even in scenarios with a high churn rate.

5. Related work

Research has harnessed the power of semantic technologies to aid in information representation, retrieval and aggregation over large distributed systems. They use the standard RDF language [16, 17] to describe data. Ontology languages such as DAML+OIL [18] and OWL [19] built on top of RDF allow describing relations between resources, thus defining a more abstract and expressive resource sharing environment. P2P technology has been used to improve the scalability and efficiency of the semantic searching. For example, systems such as Edutella [3] and InfoQuilt [21] use broadcast or flooding to search their semantic metadata, while many other projects, like RDFPeer [22] and OntoGrid [20] attempt applying DHT techniques to the retrieval of the ontology encoded knowledge. Applications like REMINDIN [23], Helios[24] and Bibster [25] add semantic short-cuts to intelligently forward queries to the right peers.

6. Conclusion

In this paper, we presented an efficient model for sharing and searching resources in an ontologically heterogeneous environment. In particular, we propose an ontology-aware topology construction method to group nodes sharing similar ontology together, so that queries only propagate in a relevant subset of peers. In addition, we adopt an intelligent query routing scheme, which routes queries to peers that are most possibly possessing answers to the query. This system has been evaluated by a group of simulations, which show that the proposed routing schemes are both efficient and scalable.

7. References

[1] HALEVY, A., IVES, Z.,MORK, P., and TATARINOV, I. 2003. Piazza: Data management infrastructure for semantic web applications. In *Proceedings of the 12th International Conference on World Wide Web*. Budapest, Hungary.

[2] J. Broekstra et al. "A Metadata Model for Semantics-Based Peer-to-Peer Systems". *In Proc. of the WWW SemPGRID 2003 Workshop*, Budapest, Hungary, May 2003.

[3] W. Nejdl et al. “EDUTELLA: a P2P Networking Infrastructure Based on RDF”. *In Proc. of the 11th Int. World Wide Web Conference (WWW 2002),* Honolulu, Hawaii, USA, May 2002.
[4] S. Shi, Y. Guanwen, D. Wang, J. Yu, S. Qu and M. Chen “Making Peer-to-Peer Keyword Searching Feasible Using Multi-level Partitioning”. *Proc. Of the 3rd International Workshop on Peer-to-Peer Systems*, San Diego, CA, USA.
[5] P.Reynolds and A. Vahdat. “Efficient Peer-to-Peer Keyword Searching”. *In Proceedings of ACM/IFIP/USENIX Middleware*, June 2003
[6] M. Cai, M. Frank, J. Chen and P. Szekely, “ MAAN: A Multi-Attribute Addressable Network for Grid Information Services” *The 4th International Workshop on Grid Computing,* 2003.
[7] LI, J., LOO, B. T., HELLERSTEIN, J., KAASHOEK, F., KARGER, D. R., AND MORRIS, R. “On the Feasibility of Peer-to-Peer Web Indexing and Search”. *In Proceedings of the 2nd International Workshop on Peer-to-Peer Systems, (IPTPS ’03).* (Berkeley, CA, Feb. 2003).
[8] C.Tang and S.Dwarkadas. “Hybrid Gloablal-Local Indexing for Efficient Peer-to-Peer Information Retrieval”. *In Proceedings of USENIX NSDI*, March 2004
[9] Juan Li, Iulian Radu and Son Vuong, "GODIS: An Ontology-Based Resource Discovery Framework for Large-Scale Grids" *in Proceedings of the 18th IASTED International Conference on Parallel and Distributed Computing and Systems,* November 13-15, 2006. Dallas, Texas, USA
[10] A. Seaborne. “RDQL: A Data Oriented Query Language for RDF Models.” www-uk.hpl.hp.com/people/afs/RDQL/
[11] Brian McBride. “Jena: Implementing the RDFmodel and syntax specification.” *Technical report, Hewlett Packard Laboratories*, Bristol, UK, 2000.
[12] N.F. Noy and M.A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. *In Seventeenth National Conference on Artificial Intelligence (AAAI-2000),* Austin, TX, 2000.
[13] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. “An environment for merging and testing large ontologies”. *Proceedings of the Seventh International Conference (KR2000)*, San Francisco, CA, 2000.
[14] Juan Li and Son Vuong "Semantic Overlay Network for Grid Resource Discovery" *in Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, November, 2005, Seattle, USA
[15] B.Bloom. “Space/time tradeoffs in hash coding with allowable errors”. *Communications of the ACM*, pages 13(7):422-426, July 1970.
[16] Ora Lassila and Ralph R. Swick, “W3C Resource Description framework (RDF) Model and Syntax Specification”.
[17] Dan Brickley and R.V.Guha. “W3C Resource Description Framework (RDF) Schema Specification”.
[18] I. Horrocks, F. van Harmelen, and P. Patel-Schneider. DAML+OIL.
http://www.daml.org/2001/03/daml+oil-index.html, March 2001.
[19] W3C. Web-ontology (webont) working group.
http://www.w3.org/2001/sw/WebOnt/.
[20] OntoGrid project: http://www.ontogrid.net/
[21] M. Arumugam, A. Sheth, and I. B. Arpinar. “Towards peer-to-peer semantic web: A distributed environment for sharing semantic knowledge on the web.” *In Proc. of the International World Wide Web Conference 2002 (WWW2002),* Honolulu, Hawaii, USA, 2002.
[22] Cai, M., Frank, M., “RDFPeers: A scalable distributed RDF repository based on a structured peer-to-peer network”, *in proc of WWW conference*, May 2004, NewYork, USA, pp 650-657
[23] Tempich, C., Staab, S., Wranik, A., “REMINDIN’: semantic query routing in peer-to-peer networks based on social metaphors” *International World Wide Web Conference (WWW)*, New York, USA, 2004.
[24] S. Castano, A. Ferrara, S. Montanelli, and D. Zucchelli. Helios: a general framework for ontology-based knowledge sharing and evolution in P2P systems. *In IEEE Proc. of DEXA WEBS 2003 Workshop*, Prague, Czech Republic, September 2003.
[25] Castano, A., Ferrara, S., Montanelli, S., Pagani, E., Rossi, G.: Ontology addressable contents in p2p networks. *In: Proceedings of the WWW’03 Workshop on Semantics in Peer-to-Peer and Grid Computing.* (2003)
[26] F. Baader, D. Calvanese, D. L. McGuiness, D. Nardi, P. F. Patel-Schneider: *The Description Logic Handbook: Theory, Implementation, Applications*. Cambridge University Press, Cambridge, UK, 2003. ISBN 0-521-78176-0
[27] College of American Pathologists. SNOMED RT - Systematized Nomenclature of Medicine Reference Terminology, VERSION 1.1, USER GUIDE, 2001.
[28] M. Andrea Rodriguez, Max J. Egenhofer, "Determining Semantic Similarity Among Entity Classes from Different Ontologies". *IEEE Transactions on Knowledge and Data Engineering,* VOL. 15, NO. 2, MARCH/APRIL 2003.
[29] J. Jiang and D. Conrath, “Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy,” *Proc. Int’l Conf. Computational Linguistics (ROCLING X)*, 1997.
[30] J. Lee, M. Kim, and Y. Lee, “Information Retrieval Based on Conceptual Distance in IS-A Hierarchies,” J. Documentation, vol. 49, pp. 188-207, 1993.
[31] G. Miller, “WordNet: A lexical database for English”, Communications of the ACM, vol. 38, no. 11, 1995.
[32] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems, In INFOCOM’03.
[33] http://www.gnutella-music-download.com