# An Ontology-based P2P Network for Semantic Search

*Tao Gu*, Institute for Infocomm Research, Singapore*
*Daqing Zhang, National Institute of Telecommunications, France*
*Hung Keng Pung, National University of Singapore, Singapore*

**ABSTRACT**

*This paper presents an ontology-based peer-to-peer network that facilitates efficient search for data in wide-area networks. Data with the same semantics are grouped together into a one-dimensional semantic ring space in the upper-tier network. This is achieved by applying an ontology-based semantic clustering technique and dedicating part of node identifiers to correspond to their data semantics. In the lower-tier network, peers in each semantic cluster are organized as Chord identifier space. Thus, all the nodes in the same semantic cluster know which node is responsible for storing context data triples they are looking for, and context queries can be efficiently routed to those nodes. Through the simulation studies, we demonstrate the effectiveness of our proposed scheme.*

*Keywords: semantic Peer-to-Peer network; ontology; DHT-based Peer-to-Peer system*

## 1 INTRODUCTION

In recent years, the use of context information has attracted a lot of attention from researchers and industry participates in ubiquitous and pervasive computing. Users and applications are often interested in searching and utilizing widespread context information. Context information is characterized as an application's environments or situations [Dey et al., 2000]. With the vast amount of context information spread over multiple context spaces and the increasing needs of cross-domain context-aware applications, how to provide an efficient context search mechanism is challenging in the context-aware research community.

One approach is to use a centralized search engine to store context data and resolve search requests. Although this approach can provide fast responses to a context query, it has limitations such as scalability, a single processing bottleneck and a single point of failure. Peer-to-peer (P2P) approaches, on the other hand, have been proposed to overcome these obstacles and are gaining popularity in recent years. P2P systems such as Gnutella [Gnutella] and Freenet [Freenet] allow nodes to interconnect freely and have low maintenance overhead, making it easy to handle the dynamic changes of peers and their data. The past years have seen an increased focus on decentralized P2P systems [Han, et al., 2006, Li, et al., 2006, Liu, et al., 2004, Morselli, et al., 2005]. However, a query has to be flooded to all the nodes in a network including the nodes that do not have relevant data. The fundamental problem that makes search in these systems difficult is that data are randomly distributed in the network with respect to their semantics. Given a

search request, the system either has to search a large number of nodes or run a risk of missing relevant data. Other P2P systems such as Chord [Stoica, et al., 2001], CAN [Ratnasamy, et al., 2001], Pastry [Rowstron, et al., 2001] and Tapestry [Zhao, et al., 2004] typically implement distributed hash tables (DHTs) and use hashed keys to direct a search request to the specific nodes by leveraging a structured network. In these systems, a data object is associated with a key which can be produced by hashing the object name. A node is assigned with an identifier which shares the same space as the keys. Each node is responsible for storing a range of keys and corresponding objects. When a search request is issued from a node, the search message is routed through the network to the node responsible for the key. They can guarantee to complete search in a logarithmic number of steps. Over years, many applications have been developed, such as file sharing [LimeWire] and content distribution [Castro, et al., 2003].

In this paper, we propose a two-tier semantic P2P network to search for context information in wide-area networks. The basic idea is to construct a two-level semantic P2P network based on metadata (i.e., context ontologies), which is essentially a semantic approach, to facilitate efficient search. In this system, context data are represented by a collection of RDF [RDF] triples. Peers with the same semantics are grouped together into a semantic cluster in the upper-tier network. All the semantic clusters are constructed as a one-dimensional semantic ring space. This is achieved by dedicating part of hashed node identifiers to correspond to their data semantics. Data semantic is extracted according to a set of schemas. Peers in each semantic cluster can be organized as a structured P2P network such as Chord identifier space in the lower-tier network. Thus, all the nodes in the same semantic cluster know which node is responsible for storing context data triples they are looking for, and context queries can be efficiently routed to those nodes.

The rest of the paper is organized as follows. Section 2 presents the detail of the two-tier semantic P2P network. Section 3 evaluates the performance of our system using simulation and presents the results. Section 4 reviews related works, and finally Section 5 concludes our work.

## 2 THE TWO-TIER SEMANTIC P2P NETWORK

In this section, we first present an overview of the two-tier semantic P2P network, followed by a description of technical details. For ease of discussion, we use the terms node and peer interchangeably for the rest of the paper.

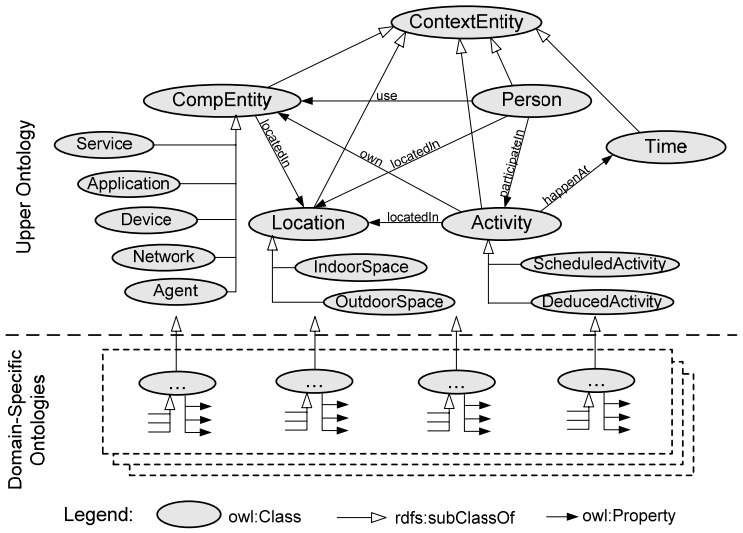
### 2.1 Overview

In this network, a large number of nodes storing context data are grouped and self-organized into a two-tier semantic P2P network, in accordance with their semantics. A node can act as producer, consumer or both. Producers provide various context data for sharing whereas consumers obtain context data by submitting their context queries and receiving results. Each node maintains a local data repository which supports RDF-based query using RDQL [RDQL]. Upon creation, each producer will first go through the ontology-based semantic mapping process to extract the semantics of its local data. It will then join a semantic cluster by applying the SHA1 hash function to the semantics of its main data. These semantic clusters logically form the upper-tier network in which each node builds its routing index based on the small world network model [Kleinberg, 2000]. In the lower-tier network, nodes in each semantic cluster are organized as Chord for storing context data and routing context queries in a logarithmic number of hops. Upon receiving a context query, the node first pre-processes it to obtain the semantic cluster associated with the query, and then routes it to an appropriate semantic cluster. In the lower-tier, the node routes the query using its finger table. Nodes that receive the query do a local search, and return results.

## 2.2 Ontology-based Semantic Clustering

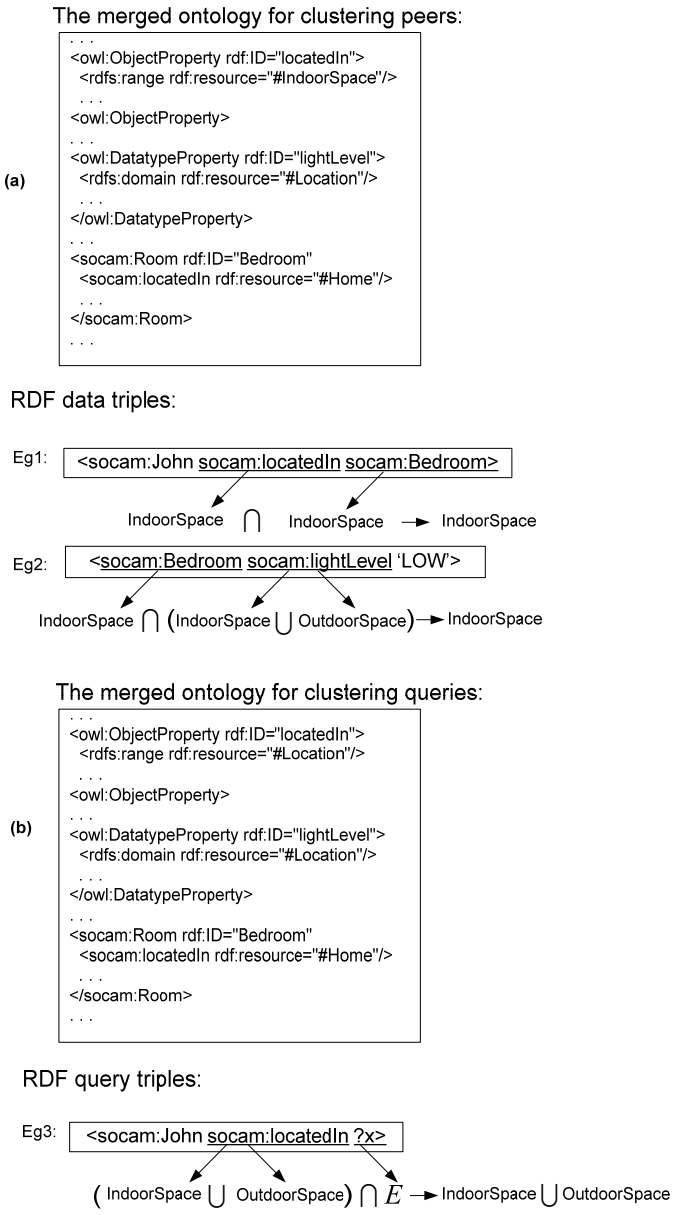
In this section, we describe how to use ontology-based metadata to extract the semantics of both RDF data and queries, and map them into appropriate semantic clusters. In our system, context data are described as RDF triples based on a set of context ontologies. We adopt a two-level hierarchy in the design of context ontologies. The upper ontology defines common concepts in a computing domain, e.g., context-aware computing, and it is shared by all peers. Each peer can define its own concepts in its lower ontologies. Different peers may store different sets of lower ontologies based on their application needs. The upper ontology can be extended with new concepts and properties upon the agreement among all the peers in the network.

*Figure 1. An example of ontology for illustration.*




To illustrate the semantic mapping process, we use an example of ontology as shown in Figure 1. All the leaf nodes in the upper ontology are used as semantic clusters, and denoted as set *E = {Service, Application, Device, ...}*. The mapping computation is done locally at each peer. For the mapping of RDF data, a peer needs to define a set of lower ontologies and store them locally. Upon joining the network, a peer first obtains the upper ontology and merges it with its local lower ontologies. Then it creates instances (i.e., RDF data) and adds them into the merged ontology to form its local knowledge base. A peer's local data may be mapped into one or more semantic clusters by extracting the subject, predicate and object of an RDF data triple. Let $SCn_{sub}$ , $SCn_{pred}$ and $SCn_{obj}$ where $n$ = 1, 2, ... denote the semantic clusters extracted from the subject, predicate and object of a data triple respectively. Unknown subjects/objects (which are not defined in the merged ontology) or variables are mapped to *E*. If the predicate of a data triple is of type *ObjectProperty*, we obtain the semantic clusters using $(SC1_{pred} \cup SC2_{pred} \cup ... SCn_{pred}) \cap (SC1_{obj} \cup SC2_{obj} \cup ... SCn_{obj})$. If the predicate of a data triple is of type *DatatypeProperty*, we obtain the semantic clusters using $(SC1_{sub} \cup SC2_{sub} \cup ... SCn_{sub}) \cap (SC1_{pred} \cup SC2_{pred} \cup ... SCn_{pred})$. Examples 1 and 2 in Figure 2a show the RDF data triples about the location and light level in a bedroom provided by a producer peer. In Example 2, we first obtain the semantic clusters from both the subject and predicate, and then intersect their results to get the final semantic cluster – *IndoorSpace*.

*Figure 2. An example of semantic cluster mapping.*




A context query follows the same procedure to obtain its semantic cluster(s), but it needs all the sets of lower ontologies. In real applications, users may create duplicate properties in their lower ontologies which conflict with the ones in the upper ontology. For example, the upper ontology defines the *rdfs:range* of predicate *locatedIn* as *Location* whereas the lower ontology defines its *rdfs:range* as *IndoorSpace*. To resolve this issue, we create two merged ontologies, one for clustering peers and the other for clustering queries. If such a conflict occurs, we select the affected properties defined in the lower ontology to generate the merged ontology for clustering peer and select the affected properties defined in the upper ontology to generate the merged ontology for clustering queries. With this scheme, a peer can extract the semantics of its

data triples more precisely without losing generality for context queries. For example, predicate *locatedIn* may have the *rdfs:range* of *IndoorSpace* in the merged ontology for clustering peers (see Figure 2a) and have the *rdfs:range* of *Location* in the merged ontology for clustering queries (see Figure 2b). Data triple <*socam:John socam:locatedIn socam:Bedroom*> will be mapped to *IndoorSpace*; and query <*socam:John socam:locatedIn ?x*> will be mapped to both *IndoorSpace* and *OutdoorSpace* rather than only *IndoorSpace*. This is most likely the case of real life applications.

### 2.3 The Upper-tier Network

In this section, we describe the process of constructing the two-tier semantic P2P network. After obtaining the semantics from its local context data, a node needs to participate in the network. It will first join an appropriate semantic cluster in the upper-tier network, and then store its data triples and participate in the lower-tier network. As a node may obtain multiple semantics from its local data, we choose the semantic cluster corresponding to the largest set of data to place the node. We call this semantic cluster the major semantic cluster of this node. The remaining semantic clusters which a node's data corresponds to are called minor semantic clusters of this node.

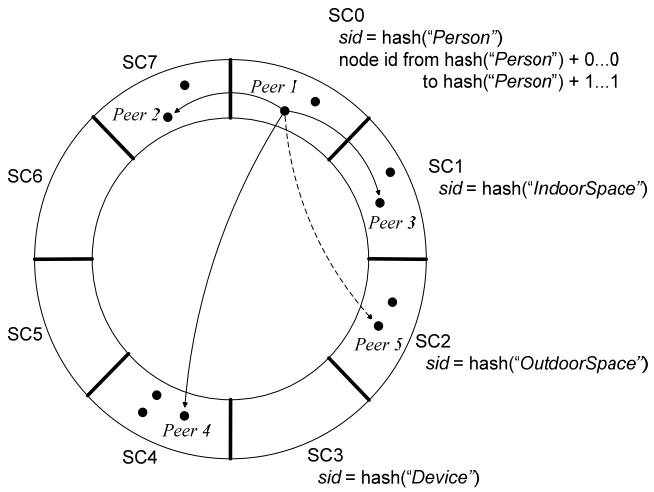
A node is assigned with an ID upon joining the network. We use SHA1 hash function to generate nodes' identifier space. To incorporate semantic information associated with a node, we dedicate part of hashed node identifiers to correspond to the semantic cluster. More specifically, in a *k*-bits identifier space, we allocate *m*-bits for semantic cluster information and *n*-bits for its IP address, where $k = m + n$. An example of a node's ID generated by hashing its semantic cluster *Person* and its IP address "137.132.81.235" is given below.

$$\text{node id} = [\text{hash}_m(\text{"Person"})][\text{hash}_n(\text{"137.132.81.235"})]$$

With this encoding scheme, we are able to construct the two-tier network and identify a node in the network, i.e., the first *m*-bits of a node's ID (called *semantic cluster ID* or *sid* in short) corresponds to the semantic cluster in the upper-tier and the last *n*-bits represents the node's ID in the lower-tier.

We follow the small world network model to construct the upper-tier network. The small network model is characterized as small average path length between two nodes in the network and large cluster coefficient defined as the probability that two neighbors of a node are neighbors themselves. Studies show that searches can be efficiently routed in small world networks when: Each node in the network knows its local neighbors (called short range contacts); and each node knows a small number of randomly chosen distant nodes (called long range contacts), with probability proportional to *1/d* where *d* is the distance [Kleinberg, 2000]. The constant number of contacts and small average path length serve as the motivation for us to build the upper-tier network using the small world network model.

*Figure 3. The construction of the upper-tier network (note: the sign "+" represents appending).*




To construct the upper-tier network, each node maintains a set of short range contacts to a peer in its neighboring semantic clusters and a number of long range contacts. As shown in Figure 3, *Peer 1* maintains *Peer 2* as its left short range contact and *Peer 3* as its right short range contact; and that results all the semantic clusters are linked linearly in a ring fashion. The long range contacts are obtained by randomly choosing a node in the upper-tier based on a distribution function with its probability proportional to *1/d*, where *d* is the semantic distance (e.g., can be represented as Euclidean distance). The long range contacts aim at providing shortcuts to reach other semantic clusters quickly. Via short range and long range contacts, search in the upper-tier network can be guided greedily by comparing *sids* of the destination and the traversed nodes. In addition, if a peer has context data corresponding to its minor semantic clusters, it needs to register the indices of these data to a random node in each of its minor semantic clusters, e.g., *Peer 1* registers its data indices to a random node – *Peer 5* in *SC2* since it has data corresponding to semantic cluster – *OutdoorSpace*. This ensures that a context query is able to reach all the relevant nodes that store the keys responsible for the query. The registration process of data indices is similar to the storing process of data triples in the lower-tier network, and it will be described in the next section.

## 2.4 The Lower-tier Network

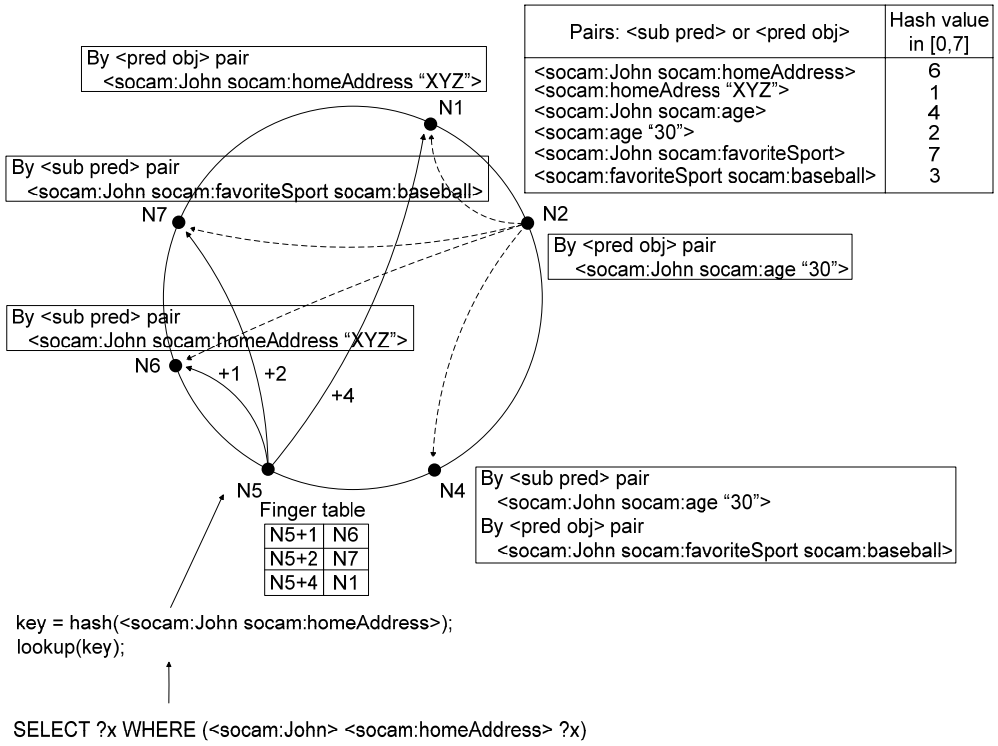
In the lower-tier network, peers in each semantic cluster are organized as Chord for storing data triples and routing context queries. This approach divides the one-dimensional Chord identifier space into multiple Chord identifier spaces. The number of neighbors maintained per node is logarithmic to the number of nodes in its semantic cluster. Hence, the maintenance cost can be reduced as compared to the original Chord.

A peer is organized into Chord based on the randomly chosen node identifier by applying the SHA1 hash function to its IP address. To facilitate efficient context query, we build distributed indices for each data triple. Each data triple is in the form of subject, predicate, and object. Since the predicate of the triple is always given in a context query, we store each data triple two times in Chord. We apply the hash function to the <*sub pred*> and <*pred obj*> pairs to generate the keys for storing each data triple. Each data triple will be stored at the successor nodes of the hashed key values of <*sub pred*> and <*pred obj*> pairs. We define the *Store* procedure to perform the above storing process for each data triple. Figure 4 illustrates the process that node *N2* stores the following data triples in a 3-bit Chord identifier space of 6 nodes.

```
<socam:John socam:homeAddress "XYZ">
```

```
<socam:John socam:age “30”>
<socam:John socam:favoriteSport socam:baseball>
```

*Figure 4. An example of 3-bit Chord identifier space of 6 nodes (could hold up to 8 nodes) for the illustrating of storing data triples and query routing.*




To register the indices of data corresponding to the minor semantic cluster(s), a node first sends a *Register* message to a random node in each of its minor semantic clusters, and then it follows the same procedure as above to store the indices.

## 2.5 Query Routing

The query routing process involves two steps: inter-cluster routing and intra-cluster routing. A context query will be first forwarded to the appropriate semantic cluster and routed to destination peers in the lower-tier network. When a node receives a context query, the destination semantic cluster can be extracted from the query using the ontology-based semantic mapping technique (described in Section 2.2). First, we obtain the search key by hashing the destination semantic cluster. We then compare the search key with the most significant *m*-bits of its neighbors' identifiers, and forward the query to the closest neighboring node. This forwarding process is recursively carried out until the destination semantic cluster is reached.

When the query reaches a node in the destination semantic cluster, the node will use its finger table to route the query in the lower-tier network. An example of the finger table of node *N5* is

shown in Figure 4. If a context query in the form of *SELECT ?x WHERE (<socam:John> <socam:homeAddress> ?x)* reaches node *N5*, node *N5* will look up the hashed *<sub pred>* pair using its fingers. Finally, node *N6* and the result *<socam:John socam:homeAddress "XYZ">* will be returned.

For a given network with *N* nodes and *M* semantic clusters, a query can be first routed to any semantic cluster in $O(\frac{1}{s} log^2 M)$ hops where *s* is the total number of long range contacts, and then routed to the destination in *log(N/M)* hops.

# 3 EVALUATION

We move on to evaluate our system using simulation and compare its performance to the original Chord. We first describe our simulation model and the performance metrics. Then we report the results from a range of simulation experiments. We also report the measurement results from the prototype system we developed.

## 3.1 Simulation Model and Metrics

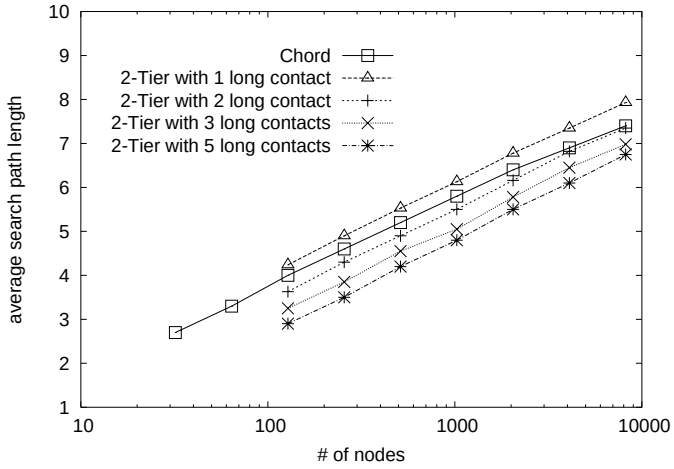
We use the AS model to generate network topologies as previous studies [Saroiu, et al., 2002] have shown that P2P topologies follow both small world and power law properties. The simulation starts with having a pre-existing node in the network and then performing a series of join operations invoked by new coming nodes. A node joins its major semantic cluster based on its local data, and then stores its data triples and registers its data indices. After the network reaches a certain size, a mixture of node joining and leaving operations is invoked to simulate the dynamic characteristic of the network. Each node is assigned with a query generation rate, which is the number of queries that it generates per unit time. In our experiments, each node generates queries at a constant rate. If a node receives queries at a rate that exceeds its capacity to process them, the excess queries are queued in its buffer until the node is ready to read the queries from the buffer. Queries are selected randomly among various semantic clusters. We set the same number of nodes for each semantic cluster in our experiments; however, in reality they can be different.

We use the following metrics to measure the performance of our system: *the search path length* measured as the average number of hops traversed by a query to the destination; *the cost of node joining/leaving* measured as the average number of messages incurred when a node joins or leaves the network.

## 3.2 Simulation Results

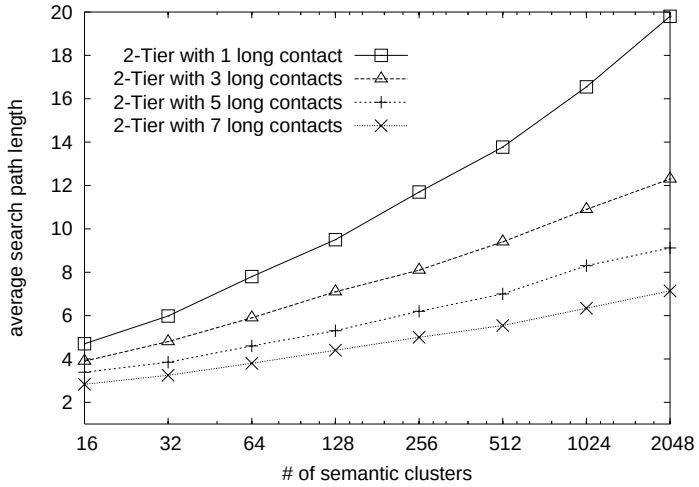
First, we evaluate the efficiency of query routing in our system and compare it to Chord. We built the two-tier network by defining a number of semantic clusters in the upper-tier. In this experiment, we fix the number of semantic clusters to 16 and vary network size from $2^5$ to $2^{13}$. Hence, each semantic cluster in the lower-tier has a number of nodes ranged from 2 to $2^9$. Figure 5 plots the average search path length of our system with 1 to 5 long range contacts on a logarithmic scale in comparison with Chord. The result shows that the two-tier network with 2 or more long range contacts has shorter search path as compared to Chord for a network size of $2^{13}$ nodes or less. It also shows that the search path length of the two-tier network is logarithmic to the number of nodes with a fixed number of semantic clusters.

*Figure 5. Average search path length vs. number of nodes for the various numbers of long range contacts.*




In this experiment, we evaluate the impact of semantic clustering in our system. We fix the semantic cluster size to 8 (i.e., 8 nodes in each semantic cluster) and vary the number of semantic clusters in the upper-tier from $2^4$ to $2^{11}$. Since the number of nodes in each semantic cluster is fixed in this experiment, the average search path length in the lower-tier is a constant. Figure 6 plots search path length vs. number of semantic clusters in our system in the various settings of numbers of long range contacts. The result shows that increasing the number of long range contacts reduces search path length significantly. Figure 6 also reveals that search path length in the upper-tier matches the small world phenomenon.

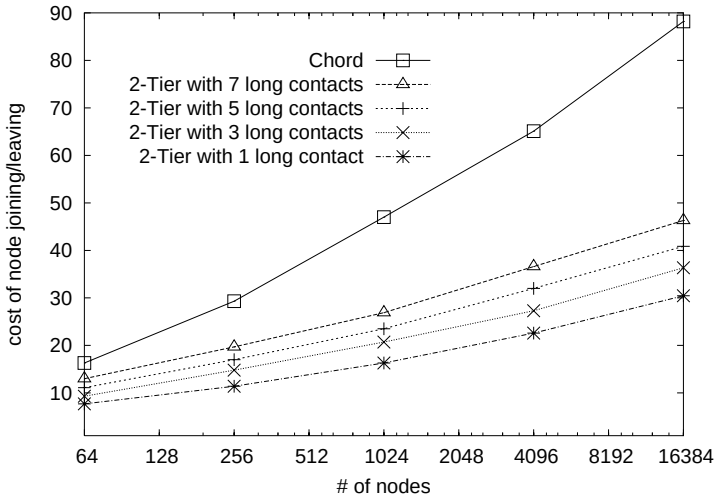
*Figure 6. Average search path length vs. number of semantic clusters in the various settings of numbers of long range contacts.*




We compare the cost of node joining and leaving between our system and Chord in this experiment. We vary network size from $2^5$ to $2^{14}$. In reality, the number of semantic clusters may increase when the network size increases. To simulate this behavior, we increase the number of semantic clusters with proportional to network size by making the number of semantic clusters equal to the number of nodes in each semantic cluster. Figure 7 plots the average number of

messages incurred when a node joins or leaves the network. The results show that our system reduces the cost of node joining/leaving significantly as compared to Chord whose update cost of node joining/leaving is $O(log^2N)$, where $N$ is the total number of nodes in the network. This is also the effect of clustering, i.e., the number of nodes in a semantic cluster is much smaller than the number of nodes in the whole network. Hence, each node needs maintain a smaller size of finger table in our system as compared to Chord.

*Figure 7. Cost of node joining/leaving.*




### 3.3 Prototype Measurement

Aim to explore practical issues in our proposed system, we develop a prototype system. We are interested in finding the bootstrapping behavior and dynamic characteristic of the network.

In the prototype, peers run on Pentium 800MHz desktop PCs with 256MB memory. The network is constructed when peers randomly join the network. We test the bootstrap process by connecting all the peers to the network in different joining sequences; hence, the structure of the network obtained may differ from one to another. When a peer starts, it first goes through the semantic clustering mapping process to identify which semantic cluster to join. The mapping process is done by iterating each of the RDF data triples and identifying its corresponding semantic cluster. Then the peer chooses the major semantic cluster to join. On average, the program initialization process takes about 4.26 seconds, and the mapping process for each RDF data triple takes about 0.251 ms. The initialization process involves reading and merging the ontology files stored locally and generating internal data structures for mapping. It is done only once when a peer starts and is only repeated if there is a change in these ontologies. Upon joining the network, each node creates and maintains a set of peers in its routing table. The joining process involves initiating the Join message, connecting to those nodes in the JoinReply message received and registering its reference if needed. The results for different steps in the bootstrap process are summarized in Table 1.

*Table 1: The results for the bootstrapping process*

| Processes | Average Time Taken |
|---|---|
| Program Initialization | 4.26 s |
| Semantic Clustering Mapping | 0.251 ms/RDF triple |
| Joining Process | 2.56 s |

We evaluate the dynamic characteristic of the network in our prototype by forcing peers to join and leave different semantic clusters randomly. Cluster splitting/merging may occur when the cluster size is greater/lower than the default size. For testing the dynamic characteristic of the network, we introduce a parameter: ***T**ime-to-**S**tability (**TS**)*. We define the steady state of a peer as the state in which a peer maintains live connections to the peers in its routing table. The steady state of a peer may collapse if one of the following events occurs:

- Its *short range contacts* or *long range contacts* leave the network or some of these peers change their major semantic clusters (due to their local data change).
- Its reference peer(s) leave the network or their major semantic clusters change.

Queries routing may be affected when peers are not in the steady state. The ***TS*** parameter is measured from the time when the steady state of a peer collapses until it reaches the steady state again. We measure the ***TS*** of the affected peers for different test cases and the results are summarized in Table 2 (note that no backup links are used in these cases).

*Table 2: Results on **TS***

| Test Cases (without backup links) | Average *TS* |
|---|---|
| Case 1: The short range contacts or long range contacts leaves the network or changes its major cluster or cluster splitting/merging occurs | 271 ms per connection |
| Case 2: Reference hosting nodes leave/change | 87 ms per reference |

In a highly dynamic network, peers leave and join frequently; this may result in high relapse rate. A high relapse rate may affect query routing in the network. To prevent this, we use a backup link for each type of connections. Once the steady state collapses, a peer can switch to the backup link immediately for the affected connection. With this backup scheme, we can minimize the disruption to query routing in the highly dynamic network where peers frequently leave and join.

## 4 RELATED WORK

Centralized RDF repositories and lookup systems, such as RDFStore [RDFStore] and Jena [Jena 2], have been implemented to support the storing and querying of RDF documents. These systems are simpler to design and reasonably fast for low to moderate number of triples. However, they have the common limitations of centralized approaches, such as single processing bottlenecks and single points of failure.

Schema-based P2P networks, such as Edutella [Nejdl, et al., 2003], are proposed to combine P2P computing and the Semantic Web. These systems build upon peers that use explicit schemas to describe their contents. They use super-peer based topologies, in which peers are organized in hypercubes to route queries. However, current schema-based P2P networks still have some shortcomings: queries have to be flooded to every node in the network, making the system difficult

to scale. Crespo et al. [Crespo, et al., 2003] proposed the concept of Semantic Overlay Networks (SONs) in which peers are grouped by semantic relationships of documents they store. Each peer stores additional information about content classification and route queries to the appropriate SONs, increasing the chances that matching objects will be found quickly and reducing the search load. However, queries still need to be flooded in each overlay network resulting in redundant query messages in the network. Cai et al. [Cai, et al. 2004] proposed a scalable and distributed RDF repository called RDFPeers based on a structured P2P system. RDFPeers organize into a multi-attribute addressable network (MAAN) [Cai, et al., 2003] which extends Chord to efficiently answer multi-attribute and range queries. When an RDF triple is inserted into the network, it will be stored three times by applying a globally-known hash function to its subject, predicate, and object. We take a similar approach to deploy Chord as the substrate for the lower-tier network, however, we store the *<sub pred>* and *<pred obj>* pairs for each data triple as the predicate is always known in a context query. Thus, the cost of inserting RDF triples into the network can be reduced. In addition, the identifier space of the lower-tier in our network is much smaller than the one in RDFPeers. Hence, the maintenance cost is lower as compared to RDFPeers since each peer maintains fewer neighbors. Tang et al. [Tang, et al., 2003] applied classical Information Retrieval techniques to P2P systems and built a decentralized P2P information retrieval system called pSearch. The system makes use of a variant of CAN to build the semantic overlay and uses Latent Semantic Indexing (LSI) [Deerwester, et al., 1990] to map documents into term vectors in the space. Li et al. [Li, et al., 2004] built a semantic small world network in which peers are clustered based on term vectors computed using LSI. They proposed an adaptive space linearization technique for constructing link structures. While we take the semantic approach which is conceptually similar to [Tang, et al., 2003] and [Li, et al., 2004], we propose the use of schema-based metadata to extract data semantics. The formal design of ontologies minimizes the problems of synonyms and polysemy incurred by VSM, and incurs a lower overhead than LSI does. Kleinberg [Kleinberg, 2000] proposed the small world network model where every node maintains four links to each of its closest neighbors and one long distance link to a node chosen from a probability function. He has shown that a query can be routed to any node in $O(\log^2 n)$ hops, where $n$ is the total number of nodes in the network. We build the upper-tier network based on the small world network model. The small world model has many advantages, such as it is easy to construct and the number of state information that each node maintains is fixed and not proportional to the number of semantic clusters. In our earlier work [Gu, et al., 2005], we have proposed a semantic P2P network for context search by using a Gnutella-like network as the substrate. However, the flooding-based routing mechanism is not very efficient in terms of search path and scalability. This paper proposes a more efficient and scalable semantic network based on a structured P2P network (i.e., Chord).

## 5 CONCLUSION

In this paper, we present an ontology-based semantic P2P network for searching context information in wide-area networks. The preliminary results have shown that our system has good search efficiency and low cost of node joining and leaving, and our system can scale to a large number of peers. The use of our system is not limited to the context-aware computing domain; in fact, it applies to any P2P searching system where schemas are explicitly defined.

## REFERENCE

Cai, M. & Frank, M. (2004). *RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network*. Paper presented at the Proceedings of the 13th International World Wide Web Conference, New York, May 2004.

Cai, M., Frank, M., Chen, J. & Szekely, P. (2003). *MAAN: A Multi-attribute Addressable Network for Grid Information Services*. Paper presented at the Proceedings of the 4th International Workshop on Grid Computing, 2003.

Castro, M., Druschel, P., Kermarrec, A.-M., Nandi, A., Rowstron, A. & Singh, A. (2003). *Splitstream: High-bandwidth Content Distribution in a Cooperative Environment*. Paper presented at the Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS 2003), February, 2003.

Crespo, A. & Garcia-Molina, H. (2003). *Semantic Overlay Networks for P2P Systems*. Technical report, Stanford University. January 2003.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W. & Harshman, R. A. (1990). *Indexing by Latent Semantic Analysis*. Journal of the American Society of Information Science, 41(6):391–407, 1990.

Dey, A. & Abowd, G. (2000). *Towards a Better Understanding of Context and Context-Awareness*. Paper presented at the Proceedings of the Workshop on the What, Who, Where, When and How of Context-awareness at CHI 2000, April 2000.

Freenet. http://freenet.sourceforge.net.

Gnutella. http://gnutella.wego.com.

Gu, T., Tan, E., Pung, H. K. & Zhang, D. (2005). *A Peer-to-Peer Architecture for Context Lookup*. Paper presented at the Proceedings of the International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2005), San Diego, California, July 2005.

Han, J. & Liu, Y. (2006). *Rumor Riding: Anonymizing Unstructured Peer-to-Peer Systems*. Paper presented at the Proceedings of IEEE ICNP, Santa Barbara, CA, November 2006.

Jena 2 - A Semantic Web Framework. http://www.hpl.hp.com/semweb/jena2.htm.

Kleinberg, J. (2000). *The Small-World Phenomenon: an Algorithm Perspective*. Paper presented at the Proceedings of the 32nd ACM Symposium on Theory of Computing, 2000.

LimeWire. http://www.limewire.com/english/content/home.shtml.

Li, M., Lee, W. C., Sivasubramaniam, A. & Lee, D. L. (2004). *A Small World Overlay Network for Semantic Based Search in P2P*. Paper presented at the Proceedings of the Second Workshop on Semantics in Peer-to-Peer and Grid Computing, in conjunction with the World Wide Web Conference, May, 2004.

Li, M., Lee, W.-C. & Sivasubramaniam, A. (2006). *DPTree: a Balanced Tree Based Indexing Framework for Peer-to-Peer Systems*. Paper presented at the Proceedings of IEEE ICNP, Santa Barbara, CA, November, 2006.

Liu, Y., Liu, X., Xiao, L., Ni, L. M. & Zhang, X. (2004). *Location-aware Topology Matching in P2P Systems*. Paper presented at the Proceedings of IEEE INFOCOM, Hong Kong, China, March 2004.

Morselli, R., Bhattacharjee, B., Srinivasan, A. & Marsh, M. A. (2005). *Efficient Lookup on Unstructured Topologies*. Paper presented at the Proceedings of ACM PODC, Las Vegas, NV, USA, July 2005.

Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I. & Lser, A. (2003). *Super-peer-based Routing and Clustering Strategies for RDF-based Peer-to-Peer Networks*. Paper presented at the Proceedings of the 12th World Wide Web Conference, May 2003.

Ratnasamy, S., Francis, P., Handley, M., Karp, R. & Shenker. S. (2001). *A Scalable Content Addressable Network*. Paper presented at the Proceedings of ACM SIGCOMM, 2001.

RDF. http://www.w3.org/RDF. World Wide Web Consortium: Resource Description Framework.

RDFStore. http://rdfstore.sourceforge.net.

RDQL. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/.

Rowstron, A. & Druschel, P. (2001). *Pastry: Scalable. Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems*. Lecture Notes in Computer Science, 2218:161–172, November 2001.

Saroiu, S., Gummadi, P. & Gribble, S. (2002). *A Measurement Study of Peer-to-Peer File Sharing Systems*. Paper presented at the Proceedings of Multimedia Computing and Networking, 2002.

Stoica, I., Morris, R., Karger, D., Kaashoek, F. & Balakrishnan, H. (2001). *Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications*. Paper presented at the Proceedings of ACM SIGCOMM, 2001.

Tang, C. Q., Xu, Z. C. & Dwarkadas, S. (2003). *Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks.* Paper presented at the Proceedings of ACM SIGCOMM 2003, Karlsruhe, Germany, August 2003.

Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D. & Kubiatowicz. J. D. (2004). *Tapestry: A Resilient Global-scale Overlay for Service Deployment*. IEEE Journal on Selected Areas in Communications, 22(1):41–53, January 2004.