

SemanticPeer: An Ontology-Based P2P Lookup Service*

Jing Tian, Yafei Dai, and Xiaoming Li

CNDS Lab, Peking University, Beijing, China
tianjing@net.pku.edu.cn
{dyf, lxm}@pku.edu.cn
<http://net.pku.edu.cn>

Abstract. Locating content efficiently and conveniently is the key issue in P2P systems. In this paper we present SemanticPeer, an Ontology-Based P2P lookup service, to address this problem, which is an extension of Chord protocol. In SemanticPeer, one can easily locate desired data items using the domain knowledge, beyond the keywords scheme. By the separate of share knowledge and private knowledge, SemanticPeer provides a high degree of node autonomy. SemanticPeer clusters the documents according to the domain knowledge, so we can answer a query with a specified characteristic efficiently.

1 Introduction

Locating a content is the core operation in a P2P system, and there are two classes of solutions for P2P content location, flooding queries as Gnutella[1] and direct locating based on Distributed Hash Table(DHT) as Chord[2]. Based on DHT, Chord has been proved to be efficient. But unfortunately none of these two methods support semantic lookup, and we can only use some keywords to locate the content. But Human brains remember objects based on their features, namely the concept of “role” in AI technology. Fortunately now more and more data gives out its metadata, and ontology description language, like RDF[3], has been developed. However, how to deploy knowledge in a completely distributed manner is challenging, because of the management of knowledge. In this paper, we present SemanticPeer, an Ontology-Based P2P lookup service, to address this problem.

SemanticPeer is a semantic extension of Chord protocol (refer to [2] for details), and it places its indices and routes queries according to the combination of ontology and DHT. SemanticPeer is built upon Chord, so a node in SemanticPeer also has the unique ID and the finger table same as in Chord, and it can use the operations defined in Chord. And SemanticPeer uses RDF to describe its knowledge. RDF is the W3C metadata standard, which is based on description logic, and it can define concepts and individuals, as while as properties.

* This work is supported by National Advanced Technology Project Fund #2001AA111013 and National Grand Fundamental Research of China G1999032706

2 The SemanticPeer Protocol

SemanticPeer protocol specifies how to distribute domain knowledge, how to distribute the indices according to the knowledge and how to find the results of a given query based on ontology. In the following, we will introduce how to distribute the indices and how to deal with queries in this section.

Knowledge Distribution. SemanticPeer uses a layered knowledge management, which includes *share knowledge* and *private knowledge*. Share knowledge is composed of domain concepts and the instances upon which most users agree, so each peer will get a copy when it joins SemanticPeer. Private knowledge is close to the concrete data items. Therefore, node n , who stores data file f , is responsible to establish private knowledge file f_{pk} with f . Fig. 1 gives out example ontology. Some resources in f_{pk} may have relationship with resources in share knowledge, therefore, f_{pk} and share knowledge will act as a whole KB k , and it should be consistent within k .

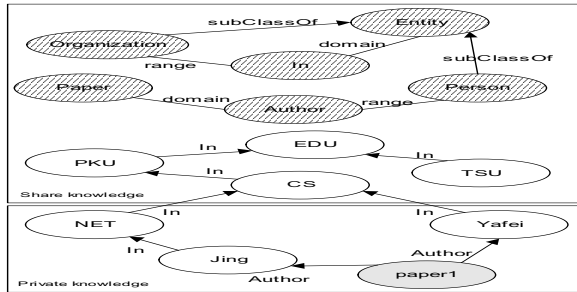


Fig. 1. Structure of knowledge in SemanticPeer. The upper part is share knowledge, including concepts (denoted by the shading ellipses) and the acknowledged instances (e.g. EDU) in publication field. The private knowledge at the bottom attaches to concrete data closely, and is easy to change with data items

Here, we introduce two data structure in SemanticPeer, *express table* and *index table*. For each resource r in share knowledge, there is a corresponding express table of r on node $n_r = successor(r)$ (*successor* is defined in [2]). And express table of r records information of $\langle r, p, r', \text{null}, n' \rangle$ if resource r' , in share knowledge, has immediate relationship p with r , and express table of r' resides on n' , as well as the information of $\langle r, p', \#FILE, fname, n' \rangle$ if resource r'' has immediate relationship p' with r , and r'' is in a private knowledge file $fname$ stored on node n'' . There is one express table for every resource in share knowledge, while index table is for every node. Node n 's index table includes the information of $\langle k, h, fname, n' \rangle$, k is the key of some resource in private knowledge file $fname$ stored on node n' , h is hash value of k and $successor(h) = n_r$. Fig. 2 illustrate the express table and index table in a SemanticPeer network with 3 nodes. The node, who establishes a private knowledge, is responsible to register the corresponding tables, and when node joining and leaving these tables

may shift in Chord protocol, due to the space limit we do not discuss in detail in this paper.

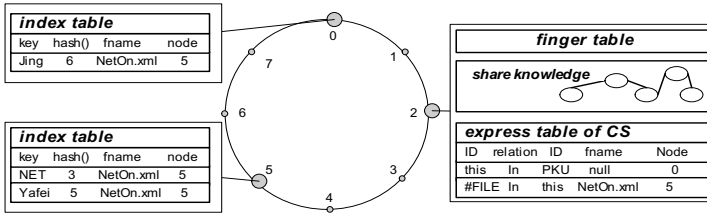


Fig. 2. A SemanticPeer network with 3 nodes. The ontology here is defined in Fig. 1. The private knowledge is stored in file “NetOn.xml” on node 5. 2’s index table and other express tables are ignored here

Deal With Query. In order to locate a concrete data file f , our main task will focus on how to navigate the query q to node n holding f_{pk} , for q can be resolved at n in the combination KB of share knowledge and f_{pk} . SemanticPeer protocol supports query in the form of RQL[4], for instance, to the knowledge illustrated in Fig. 1, we can write our needs “Return me all the papers written by a person in organization ‘NET’” as

$$\begin{aligned}
 & \text{Select } X, Y \\
 & \text{from Paper}\{X\} . \text{Author}\{Y\} . \text{In}\{ 'NET' \}
 \end{aligned}
 \tag{Q1}$$

There are two different classes of queries in SemanticPeer network, *local query* and *data query*. Local query is mainly to get what resources are in share ontology. Therefore, local query is just a query to local share knowledge, and we can resolve it as in a traditional knowledge base. Data query is to locate concrete data files in SemanticPeer, and this process is divided into three phases as follows, (a) *Consistency checking*, to check whether the properties applied to the resources in this query are appropriate according to the schema defined in share ontology. (b) *Ontology-based private knowledge locating*, to forward this query to all nodes that may hold the private knowledge file of desired data file, let N denote this nodes set. (c) *Query resolving*, to resolve this query on every node in N using combination knowledge of share knowledge and corresponding private knowledge, and return the results. Here, (a) and (c) are only operations on one node; therefore the key issue is how to forward the query to every node in N . To do the locating in (b), we study two different situations. One is to query with some resources in private knowledge like (Q1), then we can simply use the index table to locate the desired nodes (to access index table, we should fall back on Chord’s function). Consider (Q1) in SemanticPeer network of Fig. 2 for example, we first compute the hash value of ‘NET’, and get node 5 as $successor('NET')$, then use Chord protocol to access index table of node 5, finally we get ‘NetOn.xml’ on node 5. The other situation is to query with only resources in share knowledge, then we can not use index table to locate private knowledge directly, so we should navigate to private knowledge by express table. First, we contact node $n = successor(r)$, r is the known resource in query, and then we can bind

the variant r' neighbor to r in query by express table of r . If r' is bound to a share knowledge resource, we will access the express table of r' recursively (we can get the address of the node holding the express table of r' directly from express table of r); otherwise if r' is in a private knowledge file, then we get its file name and the node's address from express table.

3 Related Works

In SON[5], nodes with semantically similar content are “clustered” together in one SON, thus one query can be directed to the SONs that are better suited to answer it. To cluster the peers and queries, SON relies on a fixed classification hierarchy, what makes it less flexibility than SemanticPeer.

HyperCup[6] presents a hypercube topology of P2P network. HyperCup clusters its nodes based on a set of global ontologies. Similar to SON, this limits the flexibility of knowledge management. HyperCup relies on the super-peer to direct a query to subset of nodes that may answer it, so the super-peer will be the bottleneck.

4 Conclusions and Future Works

In this paper, we advocate the layered knowledge management in P2P system to make the management simpler and more flexible. Based on the consistent knowledge bases, SemanticPeer provides a convenient semantic lookup service. In contrast to clustering peers in other P2P systems, SemanticPeer reflects the properties in share knowledge to neighbor relationship in identifier ring, and thus makes clusters of documents according to their characteristic, so that we can find the similar documents within one cluster.

In fact, there are some problems remaining in SemanticPeer protocol for further research. For example, how to get a pretty good availability in the face of failure, and what strategy to take to enhance performance when there are more than one known resources in query. And also, we should focus on the performance analysis and simulation in our research.

References

1. “Gnutella” <http://gnutella.wego.com>.
2. I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In Proceedings of the ACM SIGCOMM, 2001. 13.
3. O. Lassila and R. R. Swick: W3C Resource Description framework (RDF) Model and Syntax Specification. <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>
4. G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl: RQL: A Declarative Query Language for RDF. In Proceedings of the World Wide Web Conference (WWW2002), Honolulu, Hawaii, USA, May 7-11 2002.
5. A. Crespo and H. Garcia-Molina: Semantic Overlay Networks for P2P Systems <http://www-db.stanford.edu/~crespo/publications/op2p.pdf>
6. M. Schlosser, M. Sintek, S. Decker, W. Nejdl: HyperCup - Hypercubes, Ontologies and Efficient Search on Peer-to-peer Networks. In Proceedings of the AP2PC, July 2002.