Available online at www.sciencedirect.com
ScienceDirect
The Journal of Systems and Software 79 (2006) 1619–1631
The Journal of Systems and Software
www.elsevier.com/locate/jss

A semantic-based P2P resource organization model R-Chord

Jie Liu [a,b,*], Hai Zhuge [a]

[a] China Knowledge Grid Research Group, Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100080, China
[b] Graduate School of the Chinese Academy of Sciences, Beijing, 100080, China

Received 15 December 2005; received in revised form 3 March 2006; accepted 6 March 2006
Available online 18 April 2006

Abstract

This paper proposes a semantic-based P2P resource organization model R-Chord by incorporating the Resource Space Model (RSM), the P2P Semantic Link Network Model (P2PSLN) and the DHT Chord protocol. Peers provide services with each other according to the content of their resources and the related configuration information. It incorporates the classification semantics and the relational semantics to provide users and applications with a uniform view on distributed resources. Experiments show that, compared to the Chord approach, the R-Chord approach is more flexible to support semantic-based queries and can significantly decrease the average visiting number of and visiting times on peers for answering queries.

Keywords: Peer data management; Peer-to-peer; Resource space model; Semantic link network; Knowledge grid

1. Introduction

With the rapid development of Peer-to-Peer (P2P) systems, Peer Data Management System (PDMS) has become a promising area.

A P2P system consists of a large number of nodes that can exchange data and services in a decentralized and distributed manner. Peers are autonomous, dynamic and heterogeneous. The original motivation for most early P2P systems was file sharing. In P2P systems, resources are distributed at multiple autonomous sites. Each site has equal functionality and can play roles of both client and server. Usually, a P2P system has the characteristics of local control of data, dynamic addition and removal of peers, local knowledge of available data and schemas, self-organization and self-optimization.

Current P2P systems are of three kinds: (1) the unstructured P2P systems such as Gnutella, where peers may join and leave the network without any notification and may connect to whomever they wish; (2) the structured P2P systems, where peers are organized into a rigid structure and connections between peers are fixed according to a certain protocol and data placement is related to the structure formed by peer connections; and (3) the hybrid P2P systems, where file sharing is decentralized, but the file directory is centralized.

The unstructured P2P systems enable complex queries. However, they provide no search guarantees and are not suitable for large-scale P2P networks. The structured P2P systems guarantee to find matching answers if the answers exist in the network; however, they cannot support complex queries. The hybrid P2P systems use servers for storing file directories and have limited scalability.

Although the issue of heterogeneous data management has been well investigated in database research, it is a novel research in P2P area due to:

(1) *Autonomous* – Lack of centralized control.
(2) *Decentralized and Distributed* – Request data from several peers.

* Corresponding author. Address: Institute of Computing Technology, Chinese Academy of Sciences, No. 6 Kexueyuan South Road, P.O. Box 2704 (28), Beijing 100080, China. Tel.: +86 10 62562703/62565533 5695; fax: +86 10 62562703.
E-mail addresses: lj@kg.ict.ac.cn, ljyying@vip.sina.com (J. Liu), zhuge@ict.ac.cn (H. Zhuge).

0164-1212/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2006.03.001

(3) *Dynamic* – Various nodes joining in and leaving.
(4) *Heterogeneous* – Different naming, different data models, and different data structures.
(5) *Scalable* – Share large amounts of data.

A PDMS consists of many autonomous, dynamic and heterogeneous nodes that exchange data and services in a decentralized and distributed manner. The key issue related to PDMS is how to organize and manage distributed resources in P2P networks for routing queries efficiently. Although there exists much research on peer data management, the issue of sharing data in large-scale networks remains to be resolved (Koloniari and Pitoura, 2005; Ooi and Tan, 2004).

This paper proposes a semantic-based resource organization model R-Chord by incorporating the Resource Space Model (RSM), the P2P Semantic Link Network Model (P2PSLN) and the DHT Chord protocol to enable resources to be effectively retrieved on the underlying network. Peers are encapsulated to provide services with each other. The R-Chord model provides users and applications with a uniform semantic view on distributed resources based on the unification of the classification semantics and the relational semantics.

2. Related work

Many efforts have been devoted to develop semantic overlay networks to organize and manage semi-structured and structured data in large-scale, decentralized, heterogeneous and dynamic environments (Aberer and Cudre-Mauroux, 2005).

According to the topology of the underlying P2P networks, the Peer Data Management Systems (PDMS) can be unstructured and structured. Previous research on unstructured PDMS mainly concerns:

(1) *Resource Management* – including a local relational model for mediating peers in PDMS (Bernstein et al., 2002), an architecture for supporting data coordination between peer databases (Giunchiglia and Zaihrayeu, 2002), the semantic overlay clustering approaches for peer organization (Nejdl et al., 2003), and P2P-based systems for distributed data sharing and management (Ng et al., 2003).
(2) *Query Routing* – including semantic-based content search approach in P2P networks (Shen et al., 2004), and the structure index over XML documents by using multi-level Breadth and Depth Bloom filters (Koloniari et al., 2003).
(3) *Query Reformulation* – including semantic and algorithmic issues of mapping data in P2P systems (Hellerstein, 2004; Kementsietsidis et al., 2003), algorithms for query reformulation and data integration between peers (Lenzerini, 2004; Tatarinov and Halevy, 2004), and the *Piazza* system for mediating between data sources on the Semantic Web (Halevy et al., 2003).

Some structured PDMS have been developed, such as

(1) The *PIER* system, for accessing data via DHTs or via an extensible iterator or wrapper that produces a stream of structured data from a local data source (Huebsch et al., 2005).
(2) The *Squid* system, for peer-to-peer information discovery through a dimension-reducing indexing schema, the *Hilbert Space-Filling-Curves* (SFCs) that can effectively map multidimensional information space to physical peers (Schmidt and Parashar, 2004).
(3) The *AmbientDB* prototype, developed at CWI, for providing full relational database functionality in ad-hoc P2P networks (Boncz and Treijtel, 2003).
(4) The *IMAGINE-P2P* platform, for supporting index-based path queries by incorporating the semantic overlay with the underlying structured P2P networks (Zhuge et al., 2005b).
(5) The distributed RDF repository *RDFPeers*, for storing each triple (*Subject*, *Predicate*, *Object*) of RDF documents at three places in a multi-attribute addressable network by applying globally known hash functions (Cai and Frank, 2004).

3. General architecture

As shown in Fig. 1, the basic R-Chord model consists of the RSM or the SLN above the unstructured or structured P2P networks, while the extended model is the combination of the RSM, the SLN and the P2P networks.

Fig. 2 illustrates the overlays of the R-Chord: (1) the top layer is the Resource Space Model (RSM) and the Peer-to-Peer Semantic Link Network (P2PSLN); (2) the middle layer is the structured P2P network, that is the Chord overlay; and (3) the bottom layer is the underlying P2P network including various data files and services at each peer.

Each resource space is stored at a super peer, the peer relatively stable and with good processing capabilities for

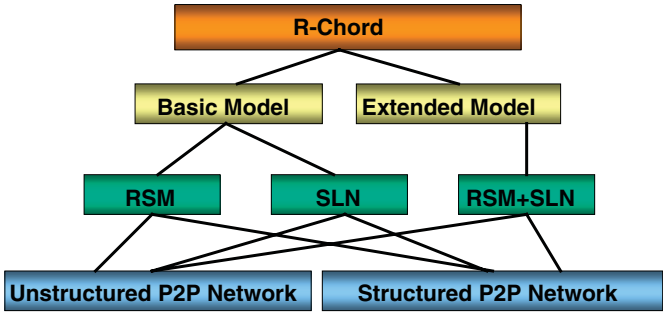


Fig. 1. General architecture of the R-Chord.

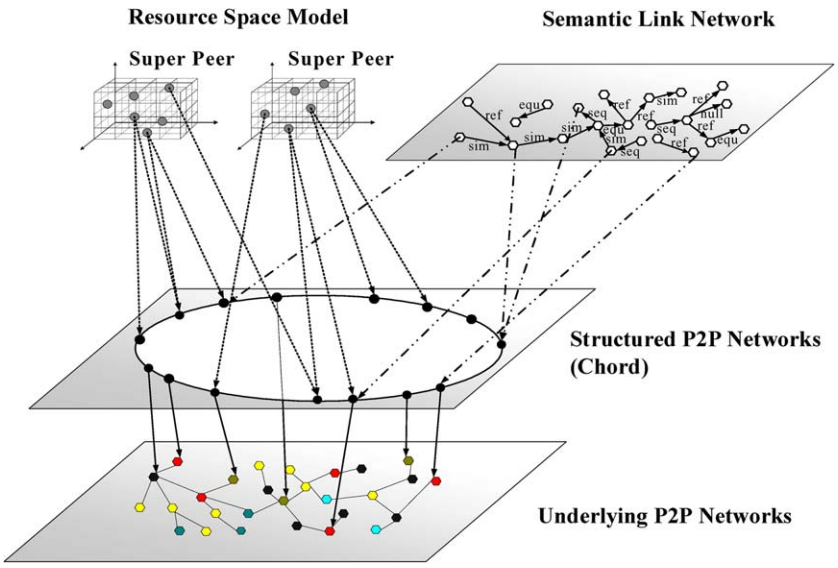


Fig. 2. Overlays of the R-Chord.

organizing and managing ordinary peers. When a peer P_i joins a P2P network, it will register at one of the super peers according to the category of data in P_i. In the following, the RSM and the super peer share the same meaning.

The Resource Space Model, denoted as $RS(X_1, X_2, \ldots, X_n)$ or RS in simple, specifies, organizes and manages resources with a universal view by using n-dimensional spaces where every point uniquely determines one resource or a set of inter-related resources (Zhuge, 2004a). Herein, RS is the name of the resource space and X_i is the name of an axis. $|RS|$ is the number of dimensions of the resource space, $X_i = \{C_{i1}, C_{i2}, \ldots, C_{im}\}$ represents axis X_i with its coordinates, and C_{ij} denotes the coordinate name. Resources in RSM can be uniquely determined by a set of given coordinates. Each point in an RSM can be a resource set, a sub resource space or a P2P semantic link network.

The P2P Semantic Link Network (P2PSLN) model includes the underlying P2P networks and a single or multiple semantic link networks (Zhuge et al., 2005a). When a peer P_i joins a structured P2P network, besides publishing its data using SHA-1 hash function, it will randomly find a peer P_j as its neighbor and establish semantic links between P_i and P_j. By establishing the semantic link networks above the underlying P2P networks can help denote the semantic relationships between peers' data schemas and between peers' services.

The resource space can help users focus on classification semantics of resources, while the SLN enables users and applications to use the semantic relationships between resources. The DHT overlay allows queries on data items that are hashed to keys in a circular address space (Chord ring) to identify the peers responsible for storing and managing data items. Deploying the RSM and the P2PSLN on the DHT overlay is an approach to support semantic-based resource organization and management.

4. Resource space model overlay

As shown in Fig. 3, a super peer keeps information of an RSM in RSM dictionary, which consists of two parts: a table called the *Resource Index* to keep resource indexed by the given coordinates, and k tables defining coordinates of each axis.

To efficiently forward queries between resource spaces, semantic links between resource spaces are established according to the semantic relationship between their corresponding axes. When a super peer P_i publishes a resource space, P_i can get the schema of its neighboring resource space $RSM(P_j)$ by sending SOAP message *Get_RSM_Schema*( ) to P_j. According to the semantic relationship between *Schema* ($RSM(P_i)$) and *Schema* ($RSM(P_j)$), semantic links between $RSM(P_i)$ and $RSM(P_j)$ can be established.

The super peers provide the functions for answering the schema inquiry and locating other super peers through semantic links and RSM view.

Let $X_i = \{C_{i1}, C_{i2}, \ldots, C_{im}\}$ and $X_j = \{C_{j1}, C_{j2}, \ldots, C_{jn}\}$ be two axes together with their coordinates, and C_{pq} denotes the coordinate name. The semantic link between X_i and X_j can be one of the followings:

1. *Equal-to Link*, denoted by X_i —*Equ*→ X_j, means that the names and the coordinates of X_i and X_j are the same.
2. *Inclusion Link*, denoted by X_i —*Inclusion*→ X_j, means that the names of X_i and X_j are the same, and $\{C_{i1}, C_{i2}, \ldots, C_{im}\} \supset \{C_{j1}, C_{j2}, \ldots, C_{jn}\}$.
3. *Extension Link*, denoted by X_i —*Extension*→ X_j, means that the names of X_i and X_j are the same, and $\{C_{i1}, C_{i2}, \ldots, C_{im}\} \subset \{C_{j1}, C_{j2}, \ldots, C_{jn}\}$.
4. *Overlap Link*, denoted by X_i —*Overlap*→ X_j, means that the names of X_i and X_j are the same, and $\{C_{i1}, C_{i2}, \ldots, C_{im}\} \cap \{C_{j1}, C_{j2}, \ldots, C_{jn}\} \neq \text{NULL}$.

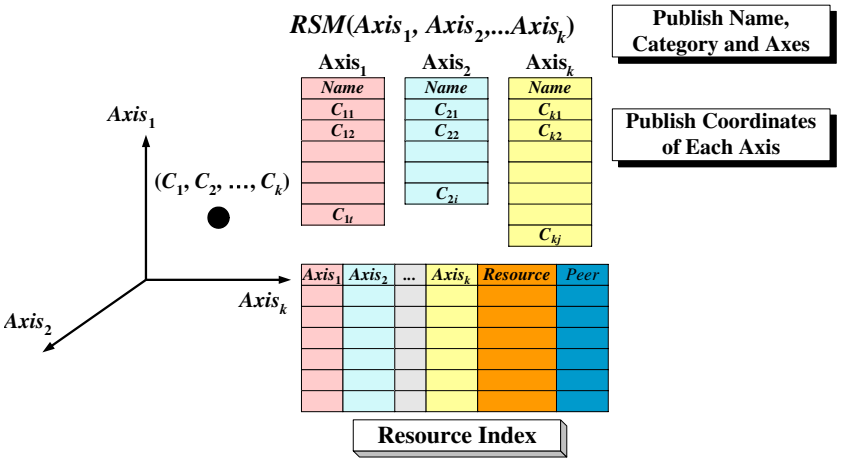


Fig. 3. RSM dictionary storing basic information.

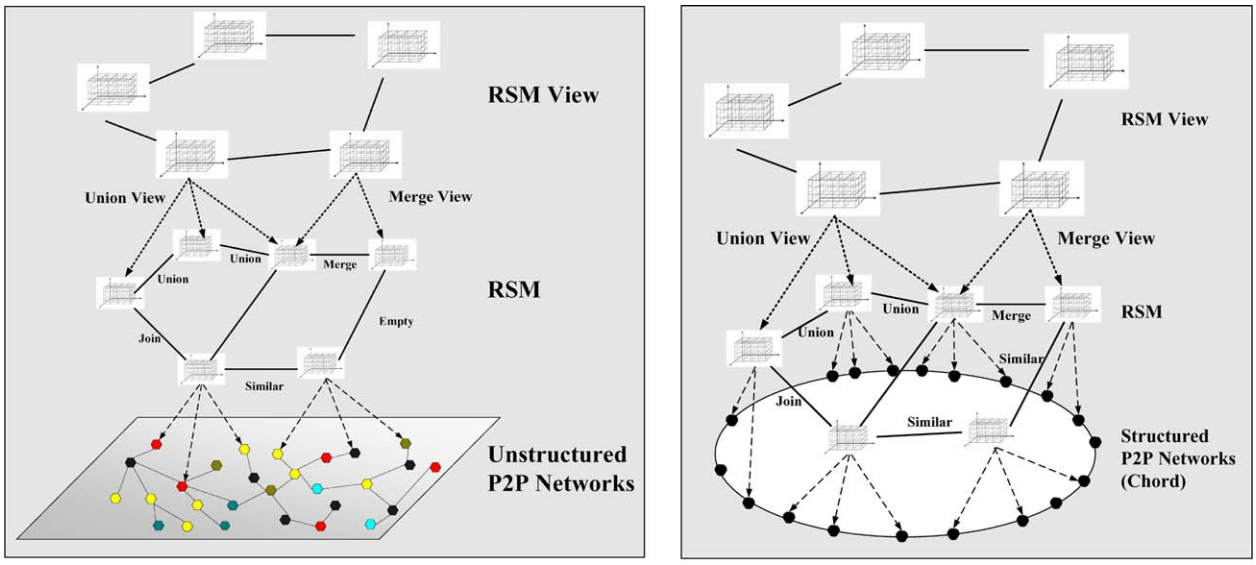
5. *Not-Overlap Link*, denoted by X_i —*NOverlap*→ X_j, means that the names of X_i and X_j are the same, but $\{C_{i1}, C_{i2}, \ldots, C_{im}\} \cap \{C_{j1}, C_{j2}, \ldots, C_{jn}\} = \text{NULL}$.
6. *Empty Link*, denoted by X_i —$\emptyset$→ X_j, means that the names of X_i and X_j are not the same and X_i and X_j are semantically unrelated to each other.
7. *Unknown Link*, denoted by X_i —N→ X_j, means that the semantic relationship between X_i and X_j is unknown.

The semantic links between two resource spaces are as follows:

- *Equal-to Link*, denoted by RS_i —*Equ*→ RS_j, means that for each $Axis_k(RS_i)$ $(1 \leqslant k \leqslant n$, and n is the total number of axes in $RS_i)$, there exists $Axis_k(RS_j)$ $(1 \leqslant k \leqslant n)$ satisfying $Axis_k(RS_i)$ —*Equ*→ $Axis_k(RS_j)$ and for each $Axis_k(RS_j)$ $(1 \leqslant k \leqslant n$, and n is the total number of axes in $RS_j)$, there exists $Axis_k(RS_i)$ $(1 \leqslant k \leqslant n)$ satisfying $Axis_k(RS_j)$ —*Equ*→ $Axis_k(RS_i)$.
- *Join Link*, denoted by RS_i —*Join*→ RS_j, means that RS_i and RS_j store the same type of resources and have k $(0 < k \leqslant \min(|RS_i|, |RS_j|))$ common axes that satisfy the join condition.
- *Merge Link*, denoted by RS_i —*Merge*→ RS_j, means that RS_i and RS_j have $n - 1$ common axes and two different axes X_1 and X_2 that satisfy the merge condition.
- *Union Link*, denoted by RS_i —*Union*→ RS_j, means that RS_i and RS_j have n common axes satisfying the union condition.
- *Inclusion Link*, denoted by RS_i —*Inclusion*→ RS_j, means that for each axis $Axis_p(RS_i)$ $(1 \leqslant p \leqslant |RS_i|)$ in RS_i, there exists an axis $Axis_q(RS_j)$ $(1 \leqslant q \leqslant |RS_j|)$ in RS_j satisfying $Axis_p(RS_i) \supseteq Axis_q(RS_j)$.
- *Extension Link*, denoted by RS_i —*Extension*→ RS_j, means that for each axis $Axis_p(RS_i)$ $(1 \leqslant p \leqslant |RS_i|)$ in RS_i, there exists an axis $Axis_q(RS_j)$ $(1 \leqslant q \leqslant |RS_j|)$ in RS_j satisfying $Axis_p(RS_i) \subseteq Axis_q(RS_j)$.
- *Overlap Link*, denoted by RS_i —*Overlap*→ RS_j, means that RS_i and RS_j have the same number of axes and each axis in RS_i satisfies $Axis_p(RS_i)$ —*Overlap*→ $Axis_q(RS_j)$ $(1 \leqslant p, q \leqslant n$, and n is the number of axes in RS_i or $RS_j)$.
- *Not-Overlap Link*, denoted by RS_i —*NOverlap*→ RS_j, means that RS_i and RS_j have the same number of axes and each axis in RS_i satisfies $Axis_p(RS_i)$ —*NOverlap*→ $Axis_q(RS_j)$ $(1 \leqslant p, q \leqslant n$, and n is the number of axes in RS_i or $RS_j)$.
- *Similar Link*, denoted by RS_i —*Similar*($Axis_p(RS_i)$ —α→ $Axis_q(RS_j)$)→ RS_j, means that RS_i and RS_j have semantic relationships with each other, and $Axis_p(RS_i)$ —α→ $Axis_q(RS_j)$ further denotes the semantic relationship between $Axis_p(RS_i)$ and $Axis_q(RS_j)$, where $\alpha \in \{Equal\text{-}to, Inclusion, Extension, Overlap, Not\text{-}Overlap\}$.
- *Empty Link*, denoted by RS_i —$\emptyset$→ RS_j, means that the axes and coordinates of RS_i and RS_j are semantically unrelated.
- *Unknown Link*, denoted by RS_i —N(*Category*)→ RS_j, means that no semantic relationship between the axes and between the coordinates of RS_i and RS_j is certainly known. Herein, the *Category* denotes the category that RS_j belongs to.

When a peer P_i wants to locate a super peer, it will first ask the super peer SP_j where P_i registered. If SP_j is not the super peer that P_i is looking for, SP_j will forward the request to its neighbors through semantic links between resource spaces.

Usually a resource space only keeps a small portion of semantic links to other resource spaces. It is inefficient to search the resource spaces one by one to locate resources. The RSM view is a virtual view over a set of sub resource spaces, which plays the similar role as view in database systems and can help retrieve resources distributed in multiple resource spaces given the axes and coordinates. The RSM view and the RSM are stored at super peers. Fig. 4 is an illustration of unstructured and structured RSM view. For clarity, the RSM view and RSM are represented at different layers.



(a) Unstructured RSM view (b) Structured RSM view

Fig. 4. Unstructured and structured RSM view.

According to the relationship between resource spaces being indexed, three types of RSM view are defined:

1. *Join View* – the view formed by the *Join* operation on the resource spaces being indexed, denoted as $RS_1 \bullet_{\text{join view}} RS_2$.
2. *Merge View* – the view formed by the *Merge* operation on the resource spaces being indexed, denoted as $RS_1 \bullet_{\text{merge view}} RS_2$.
3. *Union View* – the view formed by the *Union* operation on the resource spaces being indexed, denoted as $RS_1 \bullet_{\text{union view}} RS_2$.

According to the normal form theory of RSM (Zhuge, 2004a), we have the following lemmas:

Lemma 1. *Let* $RS_1 \bullet_{join\ view} RS_2 \Rightarrow RSJoinView$

(1) *RSJoinView is in* 1*NF if and only if both* RS_1 *and* RS_2 *are in* 1*NF.*
(2) *RSJoinView is in* 2*NF if and only if both* RS_1 *and* RS_2 *are in* 2*NF.*
(3) *RSJoinView is in* 3*NF if and only if both* RS_1 *and* RS_2 *are in* 3*NF.*

Lemma 2. *Let* $RS_1 \cup_{union\ view} RS_2 \Rightarrow RSUnionView$

(1) *RSUnionView is in* 1*NF if and only if both* RS_1 *and* RS_2 *are in* 1*NF.*
(2) *RSUnionView is in* 2*NF if and only if both* RS_1 *and* RS_2 *are in* 2*NF.*
(3) *RSUnionView is in* 3*NF if and only if both* RS_1 *and* RS_2 *are in* 3*NF.*

In the R-Chord model, the RSM dictionary defines how to form the RSM view by the join, merge or union operations on the resource spaces being indexed. The schemas of RSM dictionary and RSM view are shown in Fig. 5.

5. Extended Chord protocol

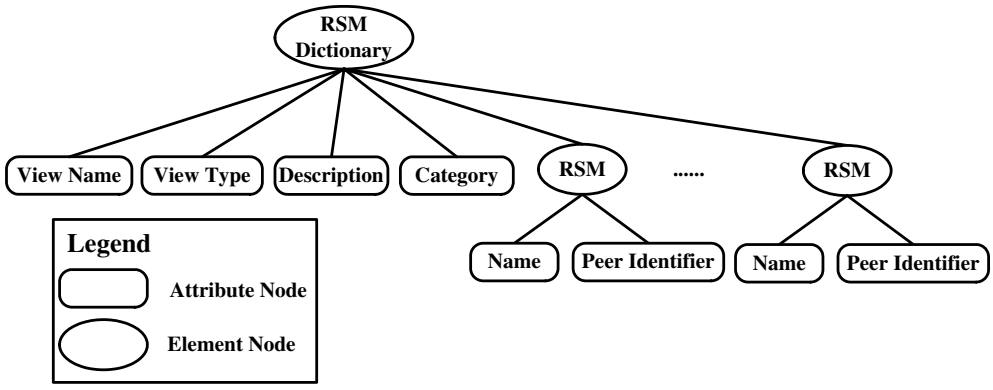
Data locating is implemented in Chord by associating a key with each data item, and storing the key/data pair at the node to which the key maps. In an N-node network, each node in Chord maintains information about only $O(\log N)$ other nodes (Stoica et al., 2001).

The R-Chord uses a suffix-tree-based hashing approach to enable complex query in structured P2P networks. A *suffix tree* is a trie-like data structure representing all the suffixes of a string. It is efficient in string matching as it only needs linear time and space for construction, and with linear time for searching strings (Ukkonen, 1995). The suffix-tree-based hashing approach consists of three steps:

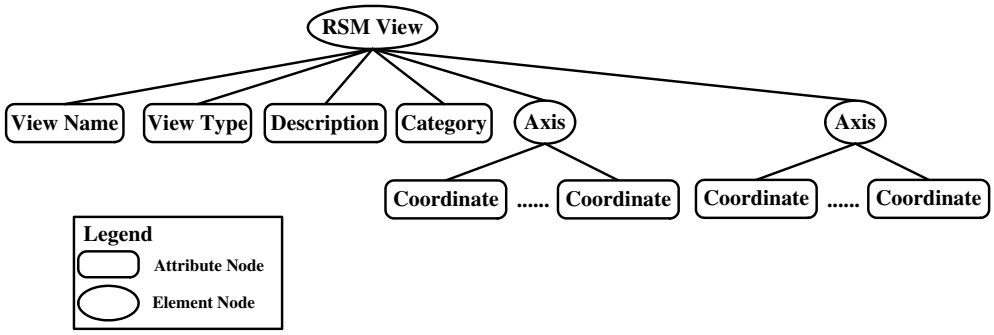
1. *Preprocessing* – Analyze the input string and filter out the preposition, the article, the conjunction, and the punctuation in data items.
2. *Suffix Tree Construction* – Construct the suffix tree corresponding to the input string.
3. *Suffix Tree Hashing* – Hash the constructed suffix tree on Chord by using *SHA*-1 function. The tree node with value v is assigned to the node with the identifier closest to *SHA*-1(v).

Let T be a given string and N be the number of words in T. The suffix tree construction algorithm starts with an empty tree and progressively adds the N prefixes of T into the suffix tree (Nelson, 1996).

After the suffix tree is constructed, each direct child *DChild* of the root is hashed on Chord by using *SHA*-1 hash function. The suffix rooted at *DChild* will be stored at the peer storing *DChild*.



(a) RSM dictionary that defines the constitution of RSM view



(b) RSM view

Fig. 5. RSM dictionary and RSM view.

Besides the finger table, each node P_i maintains a data index (*LValue*, *Suffix*, *LNode*, *LPath*, *Identifier*) that stores data hashed to P_i. The *LValue* is the value being hashed; the *Suffix* is the suffix from *LValue* to each leaf node; the *LNode* is the element name of *LValue*; the *LPath* denotes the path from the root to *LValue* in peer's schemas; and the *Identifier* is the DHT key, a 128-bit ID corresponding to the metadata including *LValue*. The extended Chord API is shown in Table 1.

Let's take a data item "*Semantic Web*, *Semantic Grid*, and *Knowledge Grid*" as an example. The suffix tree constructed is shown in Fig. 6(a), where the black nodes "*S*" (*Semantic*), "*W*" (*Web*), "*G*" (*Grid*), and "*K*" (*Knowledge*) are the nodes to be hashed on the Chord ring. As shown in Fig. 6(b), the *Suffix* from *LValue* to each leaf node is hashed to the same node as *LValue* in Chord overlay. For example, the suffix "*Semantic Web Semantic Grid Knowledge Grid*" and "*Semantic Grid Knowledge Grid*" are hashed to the same node as "*Semantic*" in the Chord overlay. The *LNode* "*Title*" is the element name, and the *LPath* "*DBLP\Article\Title*" denotes the path from the *Root* to *LValue* in peer's data schema, and the *Identifier* is the DHT key corresponding to the peer storing the data item.

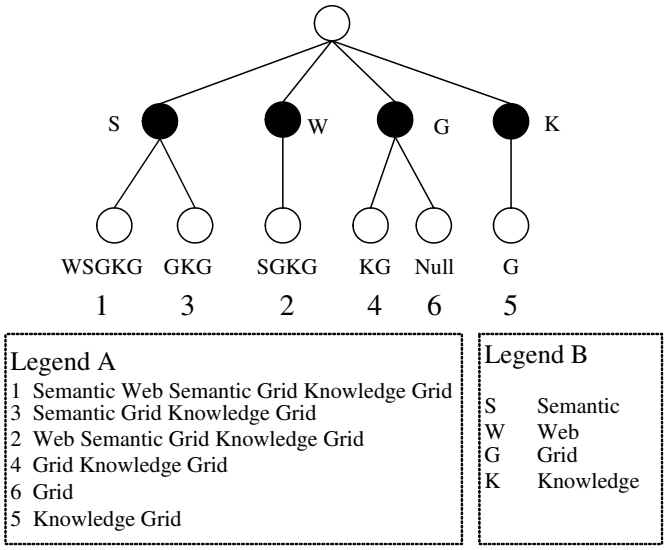
6. Semantic-based query routing

6.1. General architecture

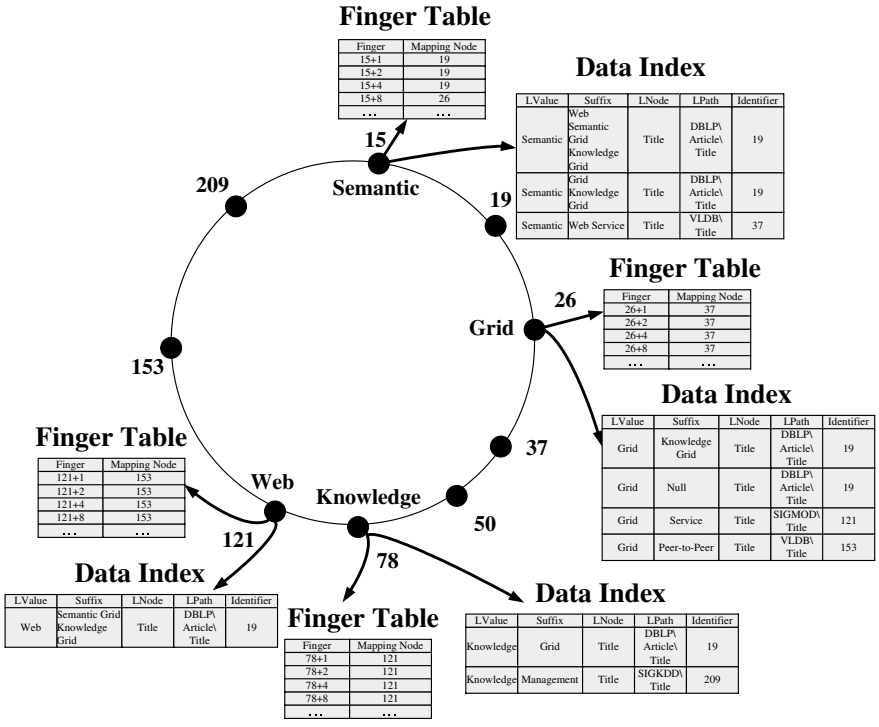
The general architecture of query routing approach is shown in Fig. 7. Users can query a peer through a Graph-

Table 1
Extended Chord API

ID	Name	Function
1	*RetrieveResource*(*Key*)	To get resource by giving its key
2	*PublishResource*(*Resource*)	To publish resource to the Chord ring
3	*Join*(*Peer*)	To join the Chord ring by giving a peer as an introducer
4	*Departure*( )	To leave the Chord ring
5	*Stabilize*( )	To periodically verify peer's immediate successors and tell the successors about the current peer
6	*Put*(*LValue*, *Suffix*, *LNode*, *LPath*, *Identifier*)	To send (*LValue*, *Suffix*, *LNode*, *LPath*, *Identifier*) to the node responsible for *SHA*-1(*LValue*)
7	*Get*(*LNode*, *LValue*) ⇒ *Identifier*	To get the node *Identifier* (the node physically stores the required data item) by giving the element name (*LNode*) and the search value (*LValue*)
8	*Get*(*LPath*, *LValue*) ⇒ *Identifier*	To get the node *Identifier* by giving the matching path (*LPath*), from the root to the current node, and the search value (*LValue*)



(a) Suffix tree corresponding to the sample title



(b) Extended Chord protocol

Fig. 6. Suffix tree based Chord protocol.

ical User Interface or by using SSeIQL (Single Semantic Image Query Language). Peers can communicate with their neighbors using SOAP messages.

Upon receiving a request, P_i will first parse and decompose the query, check whether its local repository can satisfy the requirement, and then hash the given keywords using *SHA*-1 functions for searching on the Chord overlay. However, the keyword-based Chord lookup has no semantics. Similar data having slightly differences in keyword descriptions may be hashed to different peers on Chord.

To support semantic-based queries and get more accurate results, after routing in Chord, the same request will be forwarded to neighbors of P_i through RSM index and P2PSLN index. Whenever the query reaches a peer that holds the matching data, it will be processed and the results will be returned to the query initiator.

As shown in Fig. 8, each peer in R-Chord has three types of neighbors:

(1) The neighbors on Chord overlay, which are maintained by the finger table.

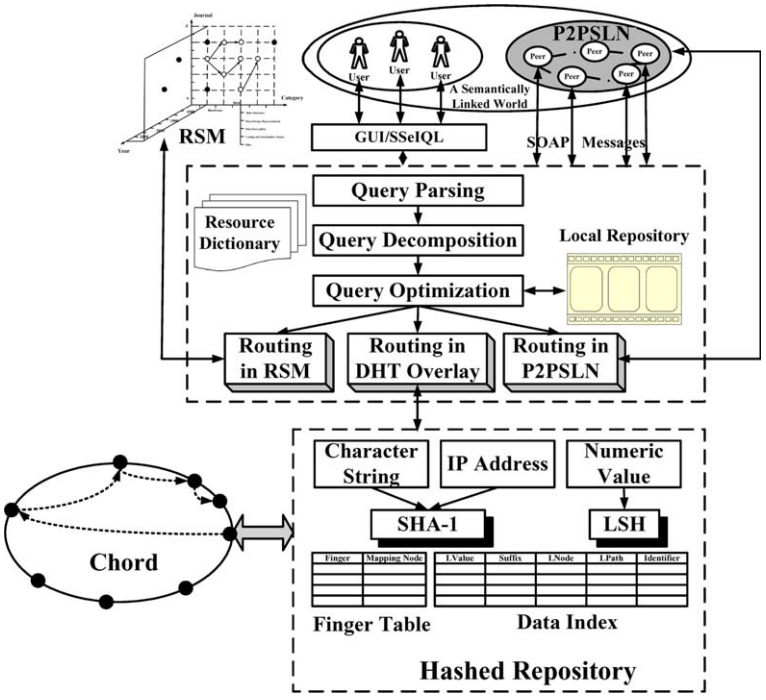


Fig. 7. General architecture of query processing.

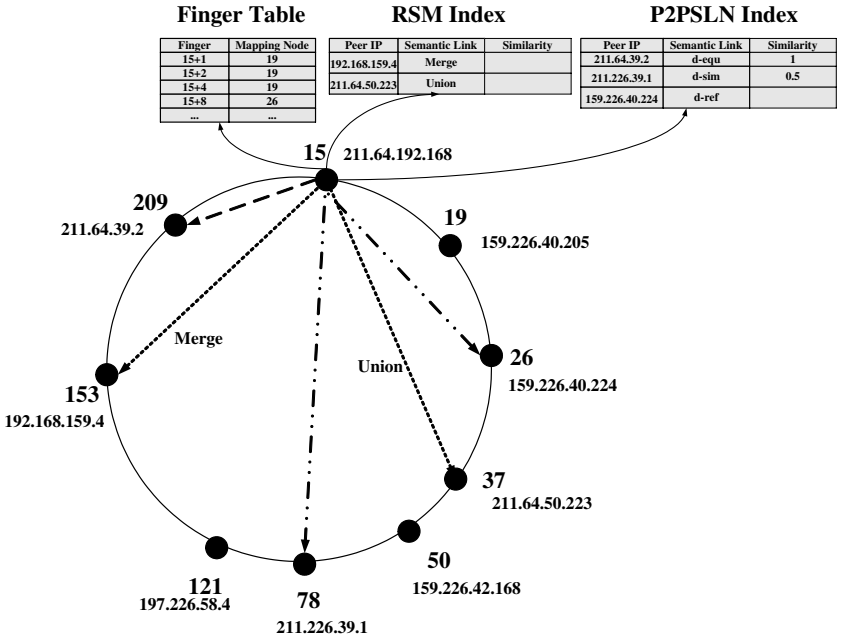


Fig. 8. Peer neighbors in the R-Chord.

(2) The neighbors in RSM overlay, which are maintained by the RSM index.
(3) The neighbors in P2PSLN overlay, which are maintained by the P2PSLN index.

6.2. Queries in RSM overlay

There are three types of routing strategies in RSM: (1) routing in the local resource space; (2) routing in the RSM view; and (3) routing in the neighboring resource spaces. Upon receiving a query $Q = \langle Q_1, Q_2, \ldots, Q_n \rangle$ ($Q_i = \langle Element, Value \rangle$, $i = 1, \ldots, n$), besides routing in the local resource space, peer P_i will route Q to its neighboring resource spaces within a predefined TTL (Time-to-Live) value.

Step 1. IF *Axis* $X_i = Q_i(Element)$ and $Coordinate(X_i) = Q_i(Value)$ $(i = 1, \ldots, n)$, THEN the neighboring resource spaces having *Equal-to*, *Union* and

Extension Link have a higher priority in query routing.

Step 2. IF *Axis* $X_i = Q_i(Element)$ $(i = 1, \ldots, n)$ and *Coordinate*$(X_i) = Q_i(Value)$ $(i = p, \ldots, q, \quad 1 \leqslant p, q \leqslant n)$, THEN the neighboring resource spaces having *Extension*, *Merge*, *Overlap*, and *Not-Overlap* Link with RSM (P_i) have a higher priority in query routing.

Step 3. IF *Axis* $X_i = Q_i(Element)$ $(i = p, \ldots, q, \quad 1 \leqslant p, q \leqslant n)$, THEN the neighboring resource spaces having *Join*, *Merge*, *Similar*, *Overlap* and *Not-Overlap* Link have a higher priority in query routing.

Step 4. Otherwise (Q cannot be answered by $RSM(P_i)$), P_i will send SOAP message *Get_RSM_Axis*( ) to P_j to get the axes of $RSM(P_j)$. IF $Axis_i(RSM(P_j)) = Q_i(Element)$ THEN Q will be forwarded to P_j.

Let $X_i = \{C_{i1}, C_{i2}, \ldots, C_{im}\}$ and $X_j = \{C_{j1}, C_{j2}, \ldots, C_{jn}\}$ be two axes. The similarity between X_i and X_j is defined as $Sim(X_i, X_j) = 2 \times Number(C_{ip}, \ldots, C_{iq})/(Number(C_{i1}, \ldots, C_{im}) + Number(C_{j1}, \ldots, C_{jn}))$, where $C_{ip}, \ldots, C_{iq}$ are the common coordinates in $\{C_{i1}, C_{i2}, \ldots, C_{im}\}$ and $\{C_{j1}, C_{j2}, \ldots, C_{jn}\}$.

Let $RSM_1(X_1, \ldots, X_m)$ and $RSM_2(Y_1, \ldots, Y_m)$ be two resource spaces. Let $Sim_i(RSM_1(X_i), RSM_2(Y_i))$ $(i = 1, \ldots, m)$ be the similarity between axis X_i and axis Y_i. Let $w_1, \ldots, w_m \in [0, 1]$ be the weights of $Sim_1, \ldots, Sim_m$ satisfying $w_1 + \cdots + w_m = 1$. The progressive weight distribution function $HW(Sim_i) = \sum_{k=1}^{i} w_k$ is 0-fuzzy metric. Using fuzzy integral approach, the similarity between RSM_1 and RSM_2 can be calculated by $Sim(RSM_1, RSM_2) = \vee_{i=1}^{m}[Sim_i \wedge HW(Sim_i)]$, where "$\wedge$" means by the minimum operation, and "$\vee$" means by the maximum operation.

6.3. Index update

In order to ensure that lookup is correct as peers join, leave or update, the R-Chord model must ensure each layer of the RSM, the P2PSLN, and the Chord overlay up to date. Each peer runs a "stabilization" protocol periodically in the background to learn about newly joining and leaving peers to update the RSM pointers, the P2PSLN links and the Chord successor pointers.

6.3.1. Index update in resource space overlay

When a peer P_i joins a P2P network, it will act as follows to build RSM index:

(1) Find the super peer $RSM(P_i)$ that P_i belongs to.
(2) If the classification value that appears in the data items of P_i cannot be described by $RSM(P_i)$, then $RSM(P_i)$ will forward the request to its neighboring resource spaces to find the matching.
(3) If the classification value can be described by $RSM(P_i)$, but the value does not appear in the coordinates, the RSM dictionary will add a new coordinate with the classification value and add an index to the data item to be published.
(4) If the classification value can be described by $RSM(P_i)$, and there exists a coordinate matching the value, then the system will add an index to the data item corresponding to the classification value.

To delete data items of peer P_i from an RSM, the system will first look up the RSM to find the resource index for that data items, and then delete the index.

6.3.2. Index update in RSM view

If there is any modification to resources in a resource space, the set of resources in RSM view changes as well. View is typically implemented as follows: When an RSM view is defined, the RSM dictionary stores the definition of the view, rather than the result of the command that defines the view. Wherever an RSM view is used in a query, it is replaced by the stored view expression. Thus, whenever the query is evaluated, the RSM view is recomputed.

To keep the RSM view up-to-date, peer P_i that stores the view definition carries out the following steps for stabilization periodically.

(1) Send SOAP messages to each peer P_j indexed by $RSMView(P_i)$.
(2) Compare each axis of $RSMView(P_i)$ with that of $RSM(P_j)$.
(3) Add axis X_i of $RSM(P_j)$ to $RSMView(P_i)$ if X_i is not indexed by $RSMView(P_i)$.
(4) Delete axis X_i from $RSMView(P_i)$ if X_i is not in all the $RSM(P_j)$ being indexed.
(5) Add coordinate C_j of $RSM(P_j)$ to $RSMView(P_i)$ if C_j is not indexed by $RSMView(P_i)$.
(6) Delete coordinate C_i from $RSMView(P_i)$ if C_i is not in all the $RSM(P_j)$ being indexed.
(7) Modify the data index of each point in $RSMView(P_i)$ according to the index in $RSM(P_j)$.

6.3.3. Index update in P2P semantic link networks

To ensure the semantic links up-to-date, each peer P_i in a P2PSLN issues $P_i.Stabilization(P2PSLN, P_j)$ periodically to have the semantic link types and the predecessor and successor pointers updated. If a predecessor or successor P_j of P_i exists in the network, it will notify P_i of its existence, of any schema change and service change. Otherwise, P_i will remove P_j from its predecessor or successor list and modify its neighbor index accordingly. When the XML schemas or services of a peer change, it will automatically notify its predecessors and successors of the new schemas or new services through SOAP messages.

6.3.4. Index update in DHT overlay

To maintain the consistent hashing mapping, when a peer P_i joins the network, certain keys previously assigned

to P_i's successor now become assigned to P_i. The following steps are performed when P_i joins the Chord overlay:

(1) Initialize the predecessor and finger tables of P_i.
(2) Update the finger tables and the predecessors of existing nodes to reflect the addition of P_i using a basic "stabilization" protocol in Chord.
(3) Move the data associated with each key for which P_i is now responsible for to P_i.
(4) Publish data items in P_i on Chord using *SHA*-1 hash function.
(5) For each published attribute value, add an index to P_i.

The number of nodes that need to be updated when a node joins the network is $O(\log N)$ with high probability. Finding and updating these nodes takes $O(\log^2 N)$ time.

When P_i leaves the network, the following tasks are performed to keep pointers in Chord up-to-date:

(1) Reassign all of P_i's assigned keys to its successor.
(2) Modify the index on P_i to its successor.
(3) Update the finger tables and the predecessors of existing nodes to reflect the addition of P_i using a basic "stabilization" protocol in Chord.

7. Experiment and comparison

To illustrate and evaluate the R-Chord approach, a simulation environment including the RSM, the P2PSLN, and the Chord overlay is established. The metadata of 610,000 papers are collected from DBLP XML databases, among which 218,509 papers are selected from 483 journals and distributed uniformly over 1000 peers.

7.1. Experiment setup

7.1.1. Establishing the RSM overlay

According to the classification attributes of the journal papers in DBLP, a two-dimensional resource space *Paper* (*Journal*, *Year*) is established. The *Journal* axis consists of 483 different coordinates, and each corresponds to a journal. The *Year* axis consists of 69 different coordinates, and each corresponds to a year. Each point in Fig. 9 represents 50 papers.

7.1.2. Establishing the Chord overlay

The Chord overlay consists of 1000 peers. As the overlay network configuration and operations are based on Chord, its maintenance cost is of the same order as in Chord. The journal papers from DBLP are hashed to Chord as follows:

- The *Title* of each paper is hashed using suffix-tree-based approach.
- The *Author* of each paper is hashed using the first name, the last name and the whole name, respectively.
- The *Journal name* is hashed as a string.
- The *Year* is hashed as a string.

7.2. Performance analysis

Fig. 10 plots the number of data nodes involved in the R-Chord and Chord approaches. The data nodes are the nodes that store data items satisfying a query, while the involved nodes are the nodes being visited to locate the required data nodes according to the Chord protocol. On average, the number of data nodes visited by the R-Chord and Chord approaches is 194 and 835. The optimum number of data nodes involved is 89. Fig. 10 shows that for answering a query, almost all the data nodes in the P2P networks are involved in Chord, while only 23.23% of all the data nodes are involved in R-Chord.

Fig. 11 plots the visiting times on data nodes in R-Chord and Chord approaches. On average, the visiting times on the data nodes are 884 and 1835, respectively. On average, the optimum visiting times on the data nodes are 97. Fig. 11 shows that for answering a query, the visiting times on data nodes in R-Chord approach are about 48.17% of the visiting times in Chord approach.

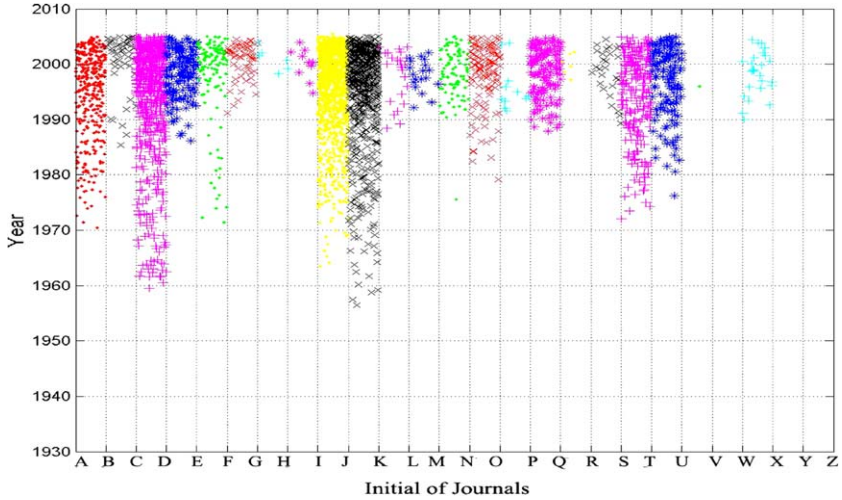


Fig. 9. Paper distributions in the resource space model.

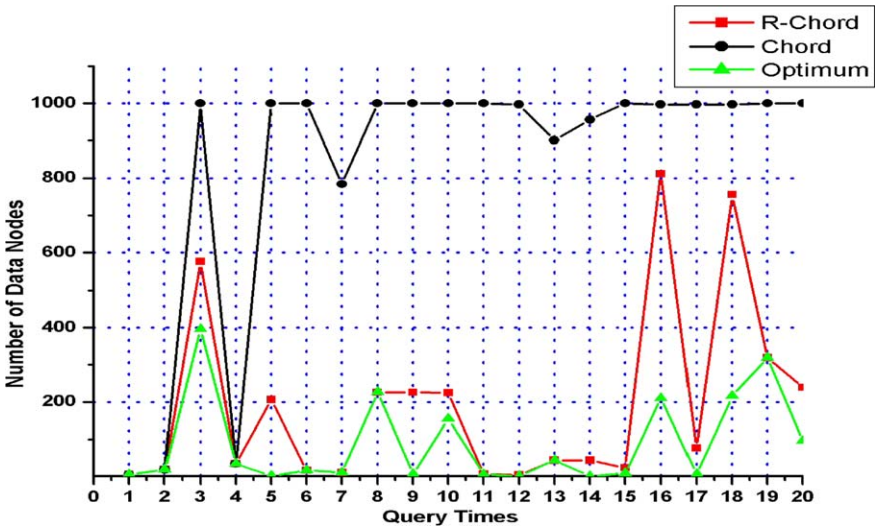


Fig. 10. Number of data nodes involved in R-Chord and Chord approaches.

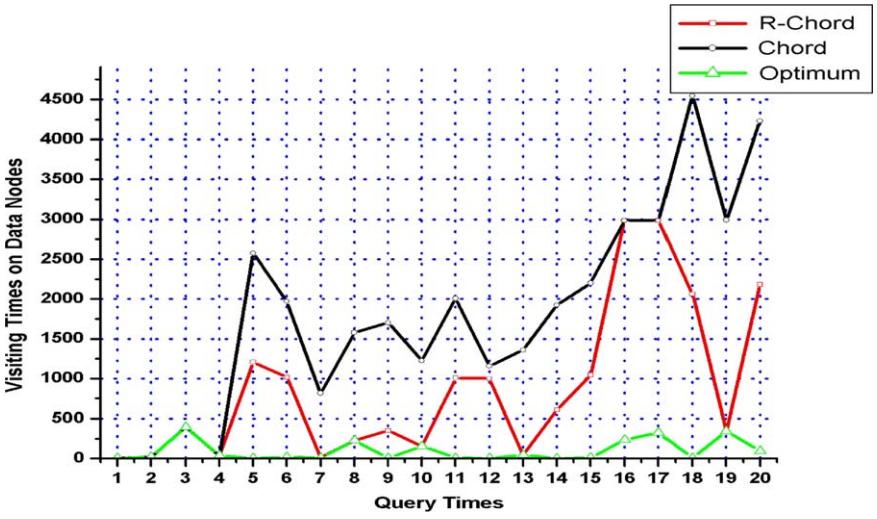


Fig. 11. Visiting times on data nodes in R-Chord and Chord approaches.

Fig. 12 plots the total visiting times of nodes involved for answering each query. On average, the total visiting times of nodes involved in R-Chord and Chord approaches are 21,277 and 75,102. By comparing Fig. 11 with Fig. 12, the following conclusions can be drawn:

(1) The visiting times on data nodes are quite less than the total involved times of all the nodes.
(2) To answer a query, the total involved times of nodes in the R-Chord approach are only about 28.33% of that of the Chord approach. This is because the R-Chord approach uses the classification attribute (*Journal*, *Year*) to organize and retrieve data, which can reduce the search times exponentially, while on average the latter requires $O(\log N)$ steps to locate each required data item.

7.3. Comparison

Compared with previous semantic-based routing strategies in structured P2P networks, the R-Chord approach is mostly like the *Space-Filling-Curves* (SFCs) approach, which solves the problem of complex queries in structured P2P networks by mapping multidimensional information space to physical peers (Schmidt and Parashar, 2004). An *SFC* is a continuous mapping from *d*-dimensional space to 1-dimensional space. The *d*-dimensional space is regarded as a *d*-dimensional cube with the *SFC* passing once through each point in the cube's volume, entering and exiting the cube only once. Using this mapping, a point in the cube can be described by its spatial coordinates or by the length along the curve measured from one of its ends. The recursive, self-similar, and locality-preserving properties of

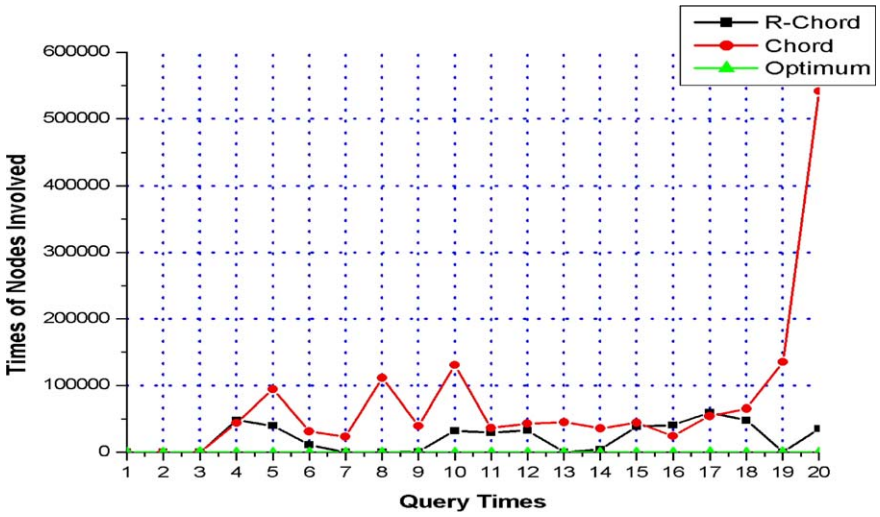


Fig. 12. Visiting times on nodes in R-Chord and Chord approaches.

SFCs support complex queries such as partial keywords, wildcards and ranges in P2P networks. The differences between the R-Chord and SFC-based routing approaches are in three aspects:

1. *Number of the Coordinates* – Both the SFC and the R-Chord approaches organize resources in *d*-dimensional spaces; however, the SFC approach requires that the number of coordinates on each axis is the same, while the RSM approach does not have that limitation so it is more flexible for resource organization.
2. *Integrity Constraint* – The SFC-based approach does not consider integrity constraints, while the normal forms of RSM provide designers with guidelines for guaranteeing the correctness of operations on RSM. The 1NF avoids explicit coordinate duplication, the 2NF prevents one coordinate from semantically depending on another, and the 3NF ensures that resources are properly classified.
3. *Multiple Spaces vs. One Space* – Only one *d*-dimensional space is involved in the SFC-based approach, while in R-Chord approach, semantic links between resource spaces are established to denote the semantic relationship between multiple resource spaces for query routing.

Table 2
Advantages of R-Chord

Application requirements	Solutions	Provided by
Semantic relationship between peers' schemas	Semantic link	P2PSLN
Semantic relationship between resource spaces	Semantic link	P2PSLN
Semantic-based P2P model	Classification semantics	RSM
	Relational semantics	P2PSLN
Semantic-based query routing	Axis- and coordinate-based routing	RSM
	Semantic-link-based routing	P2PSLN
	Complex query in structured P2P networks	Chord (Suffix Tree)

Compared with previous work in P2P resource organization, the R-Chord approach can meet more application requirements listed in Table 2.

8. Conclusion

This paper proposes R-Chord, a new semantic-based peer data management model, by incorporating the Resource Space Model, the P2P Semantic Link Network Model and the DHT Chord protocol. It incorporates the advantages of the structured and unstructured P2P networks. The Chord protocol ensures the efficiency of query routing in large-scale P2P networks, while the RSM and the P2PSLN routing strategies incorporate the classification semantics and the relational semantics. Combination of these models forms an efficient solution for P2P resource management. The efficiency of R-Chord is measured by three criteria: (1) the number of data nodes involved for answering queries; (2) the visiting times on data nodes; and (3) the visiting times on all the involved nodes. The R-Chord approach provides a scalable semantic overlay for managing distributed resources in the Knowledge Grid (Zhuge, 2004b).

Ongoing work includes two aspects: (1) supporting advanced relational operations, such as join, top-K ranking, in P2P networks; and, (2) incorporating query optimization techniques into R-Chord to improve its effectiveness and efficiency.

Acknowledgements

The research work was supported by the National Science Foundation of China (No. 60503047) and the National Basic Research Program of China (973 Project No. 2003CB317000).

References

Aberer, K., Cudre-Mauroux, P., 2005. Semantic overlay networks. In: Proceedings of VLDB; 2005. p. 1367.

Bernstein, P., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., Zaihrayeu, I., 2002. Data management for peer-to-peer computing: a vision. In: Proceedings of WebDB; 2002. pp. 89–94.

Boncz, P., Treijtel, C., 2003. AmbientDB: relational query processing in a P2P network. In: Proceedings of DBISP2P 2003, pp. 153–168.

Cai, M., Frank, M., 2004. RDFPeers: A scalable distributed RDF repository based on a structured peer-to-peer network. In: Proceedings of WWW; 2004. pp. 650–657.

Giunchiglia, F., Zaihrayeu, I., 2002. Making peer databases interact – a vision for an architecture supporting data coordination. In: Proceedings of CIA; 2002. pp. 18–35.

Halevy, A., Ives, Z., Mork, P., Tatarinov, I., 2003. Piazza: Data management infrastructure for semantic web applications. In: Proceedings of WWW; 2003. pp. 556–567.

Hellerstein, J., 2004. Architectures and algorithms for Internet-scale (P2P) data management. In: Proceedings of VLDB; 2004. p. 1244.

Huebsch, R., Chun, B., Hellerstein, J., Loo, B., Maniatis, P., Roscoe, T., Shenker, S., Stoica, I., Yumerefendi, A., 2005. The architecture of PIER: an Internet-scale query processor. In: Proceedings of CIDR; 2005. pp. 28–43.

Kementsietsidis, A., Arenas, M., Miller, R., 2003. Mapping data in peer-to-peer systems: semantics and algorithmic issues. In: Proceedings of SIGMOD; 2003. pp. 325–336.

Koloniari, G., Pitoura, E., 2005. Peer-to-peer management of XML data: issues and research challenges. ACM SIGMOD Record 34 (2), 6–17.

Koloniari, G., Petrakis, Y., Pitoura, E., 2003. Content-based overlay networks of XML peers based on multi-level bloom filters. In: Proceedings of DBISP2P; 2003. pp. 232–247.

Lenzerini, M., 2004. Principles of P2P data integration. In: Proceedings of DIWeb; 2004. pp. 7–21.

Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I., Loser, A., 2003. Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In: Proceedings of WWW; 2003. pp. 536–543.

Nelson, M., 1996. Fast string searching with suffix trees. Dr. Dobb's Journal. Available from: <http://www.dogma.net/markn/articles/suffixt/suffixt.htm>.

Ng, W., Ooi, B., Tan, K., Zhou, A., 2003. PeerDB: A P2P-based system for distributed data sharing. In: Proceedings of ICDE; 2003. pp. 633–644.

Ooi, B., Tan, K., 2004. Guest editors' introduction: special section on peer-to-peer-based data management. IEEE Transactions on Knowledge and Data Engineering 16 (7), 785–786.

Schmidt, C., Parashar, M., 2004. Enabling flexible queries with guarantees in P2P systems. IEEE Internet Computing 8 (3), 19–26.

Shen, H., Shu, Y., Yu, B., 2004. Efficient semantic-based content search in P2P network. IEEE Transactions on Knowledge and Data Engineering 16 (7), 813–826.

Stoica, I., Morris, R., Karger, D., Kaashoek, M., Balakrishnan, H., 2001. Chord: a scalable peer-to-peer lookup service for Internet applications. In: Proceedings of SIGCOMM; 2001. pp. 149–160.

Tatarinov, I., Halevy, A., 2004. Efficient query reformulation in peer data management systems. In: Proceedings of ACM SIGMOD; 2004. pp. 539–550.

Ukkonen, E., 1995. On-line construction of suffix trees. Algorithmica 14 (3), 249–260.

Zhuge, H., 2004a. Resource space grid: model, method and platform. Concurrency and Computation: Practice and Experience 16 (14), 1385–1413.

Zhuge, H., 2004b. The Knowledge Grid. World Scientific Publishing Co., Singapore.

Zhuge, H., Liu, J., Feng, L., Sun, X., He, C., 2005a. Query routing in a peer-to-peer semantic link network. Computational Intelligence 21 (2), 197–216.

Zhuge, H., Sun, X., Liu, J., Yao, E., Chen, X., 2005b. A scalable P2P platform for the knowledge grid. IEEE Transactions on Knowledge and Data Engineering 17 (12), 1721–1736.

Jie Liu is an assistant professor at the Institute of Computing Technology, Chinese Academy of Sciences. Her research interests are Semantic Link Network and Peer-to-Peer Data Management. She has published papers in leading international journals such as *Computational Intelligence* and *IEEE TKDE*. She received the *Special Prize of President Scholarship* of the *Chinese Academy of Sciences* in 2004, and the *Microsoft Fellow Award* in 2003. She is serving on the editorial board of the *International Journal of Knowledge and Learning*, and was the Program Co-Chair of the *First International Conference on Semantics, Knowledge and Grid* and the *Second International Workshop on Knowledge Grid and Grid Intelligence*. She is a member of IEEE and a member of ACM.

Hai Zhuge is a professor and the director of the Key Lab of Intelligent Information Processing in Chinese Academy of Sciences. He is the founder of the China Knowledge Grid Research Group (http://kg.ict.ac.cn), which employs 30 young researchers. Currently, he is the chief scientist of the China National Semantic Grid Project, a five-year project of the National Basic Research Program of China. He was the keynote speaker at the 2nd International Conference on Grid and Cooperative Computing (GCC2003), the 5th and 6th International Conference on Web-Age Information Management (WAIM2004 and WAIM2005), the 1st International Workshop on Massive Multi-Agent Systems (MMAS2004), the 1st International Workshop on Autonomous Intelligent Systems – Agents and Data Mining (AIS-ADM2005), the 8th International Conference on Electronic Commerce (ICEC2006), and the 1st Asia Semantic Web Conference (ASWC2006). He was the Chair of the 2nd International Workshop on Knowledge Grid and Grid Intelligence, the Program Co-Chair of the 4th International Conference on Grid and Cooperative Computing (GCC2005) and the Chair of the 1st International Conference on Semantics, Knowledge and Grid (SKG2005). He has organized several international journal special issues on Semantic Grid and Knowledge Grid. He is serving as the Area Editor of the *Journal of Systems and Software* and the Associate Editor of *Future Generation Computer Systems*. His current research interest is the model and theory on the future interconnection environment and applications in China. His monograph *The Knowledge Grid* is the first book in the area. He is the author of over 100 papers appeared mainly in leading international conferences and journals such as *Communications of the ACM*; *IEEE Computer*; *IEEE Transactions on Knowledge and Data Engineering*; *IEEE Intelligent Systems*; *IEEE Transactions on Systems, Man, and Cybernetics*; *Computational Intelligence*; *Information and Management*; *Decision Support Systems*; and *Journal of Systems and Software*. He is among the top scholars in the systems and software engineering field during 1999–2003 and 2000–2004 according to the assessment report published in *Journal of Systems and Software* in 2004. He is a Senior Member of IEEE and a member of ACM.