Complex Queries for Heterogeneous Resources on a Structured P2P Semantic Link Network

Hai Zhuge, Xue Chen and Xiaoping Sun

China Knowledge Grid Research Group, Key Lab of intelligent Information Processing

Institute of Computing Technology, Chinese Academy of Science, Beijing, 100190, China

zhuge@ict.ac.cn

{chenxue, sunxp}@kg.ict.ac.cn

***Abstract*—This paper investigates the issue of realizing complex queries for heterogeneous resources on dynamic and large-scale decentralized networks. We build a distributed index on a structured P2P network HRing to represent semantic relations between resources to support complex query, and establish semantic links among nodes of P2P network to realize efficient routing of queries. Incorporating distributed index, semantic links and HRing forms a structured P2P Semantic Link Network (SemHRing). Current search engines are limited in ability to support relational queries, which are often required in real applications. SemHRing can support keyword queries and relational queries while guaranteeing high performance and low maintenance cost as well as high robustness. SemHRing can be a feasible solution to the distributed storage system for next-generation search engines.**

I. INTRODUCTION

How to effectively organize heterogeneous resources on decentralized network and provide efficient complex query services is a challenge. Current search engines such as Google and Yahoo! mainly offer keyword queries. Demands on searching for or by relations are increasing.

Research on complex queries in structured P2P networks concerns two problems: routing performance and query types. The structured P2P networks have the potential to be scalable, efficient and robust. However, current structured P2P networks only support limited query types due to its topology construction method.

A typical query-answer process of a structured P2P network consists of two conversion sub-processes before routing (see Fig.1): the conversion from keywords into the IDs of the desired data objects and the conversion from data object IDs into the desired node IDs. The query types a P2P network can support is determined by the first conversion. Keywords represent certain semantics, while the only way to know data objects in a network is their IDs. The topology construction method like DHT (Distributed Hash Table) maps keywords into a uniform binary ID space, which destroys the semantics of data objects and thus incur high cost for complex queries [23][25][27][29]. To support complex queries, the topology construction method should enable data object IDs to preserve the semantics of data objects. One intuitive way is to directly use keywords as IDs, which requires the network to be able to support string ID space. The second conversion illustrates the mapping between data object IDs and the node IDs (they belong to the same ID space). Thus, before a query is routed in the network, the source node needs to determine the IDs of the target nodes that are responsible for the desired data objects. Then, searching for a query is transformed into the routing from the source node to target nodes with specific IDs.

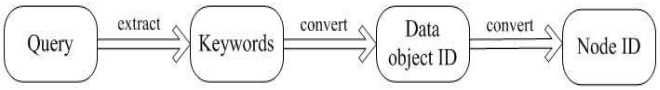

Fig.1. Query analysis process before routing.

To support complex queries, a structured P2P network should satisfy two requirements: First, the network topology should be efficient, scalable and robust. Second and more importantly, it should be able to preserve the semantics of data objects.

A ring-structured P2P network HRing based on Harmonic Series is proposed in [33]. It can achieve both high performance and low maintenance cost, while guaranteeing remarkable robustness. Further, the construction of HRing topology is entirely independent of the ID space. It does not rely on data types and data management method. Thus, HRing can serve as the underlying P2P overlay for managing decentralized heterogeneous resources.

Our solution is to build semantic relations over the HRing to support both keyword queries and relational queries with high performance and low maintenance cost.

II. RELATED WORK

A. P2P Topologies and Relevant Complex Queries

P2P topology design plays a fundamental role in the design of routing algorithm and query types. Different topologies need different routing algorithms and support different query types, thus, having different performance and cost. Many structured P2P topologies have been proposed. The representative methods are DHT, balanced tree/trie, small-world and Skip list.

DHT method maps keywords of data objects and node identifiers (like IP addresses) into a uniform *m*-bit binary space, where nodes and data objects obtain binary IDs. Data objects on DHT networks are organized by their ID order. Each node is responsible for a specific range of the ID space. DHT networks can efficiently support exact-match query, but they cannot preserve the semantics and locality of keywords of data objects. It is hard for them to directly support complex

queries such as range query and multi-attribute query. Typical DHT networks include CAN [23], Chord [26], Pastry [25], and Tapestry [29].

To support complex query on DHT networks, the key is to make DHT adaptive to string IDs. Much work has been done to build distributed index on DHT networks. Aberer et al. design a distributed indexing tree structure P-Grid to support range query [1]. Ramabhadran et al. build a binary trie called prefix hash trees (PHT) over DHT to support range query and load balance [22]. Crainiceanu et al. provide a B+-tree-like hierarchical index structure to support range query based on the assumption of one data object per node [6]. Zhuge et al. build a distributed trie index on Chord to support range query as a scalable Knowledge Grid platform [32]. Cai et al. extend Chord to support range query and multi-attribute query using a uniform locality preserving hash function to map data into the Chord identifier space [5]. Zhu et al. exploit the locality sensitive hash functions on DHT overlays to realize that semantically close files are clustered into the same peers with high probability [30].

Tree-structured P2P networks use the property of the balanced tree structure to achieve high search efficiency. Such structures support range query by directly using keywords of data objects as their IDs. BATON is a balanced binary tree structured P2P overlay by building vertical and horizontal links in each node [15]. Kothari et al. present a balanced binary tree-like structure to support range query [19]. Zatloukal and Harvey introduce the family tree, an ordered and distributed dictionary data structure with each node having constant pointers [28]. Query routing costs $O(\log(n))$ in expectation and $O(\log^2(n))$ with high probability.

Skip-List-based overlays such as SkipNet [11] and Skip Graph [3] support range queries by using random numeric IDs to construct routing tables and using name IDs to locate data objects. They can achieve $O(\log(n))$ routing hops in expectation with $O(\log(n))$ routing table size. Harvey and Munro provide a deterministic SkipNet that ensures deterministic $O(\log(n))$ bound for routing hops [12]. But node insertion and departure require $O(\log^2(n))$ time. Aspnes et al. propose a bucket-based Skip Graph that reduces the space complexity of a Skip Graph from $O(m\log(m))$ to $O(n\log(n))$, where m is the number of data objects, and n is the number of nodes in the system [2].

Small-world phenomena can be exploited to build P2P overlays. Kleinberg provides a method to model a small-world in a two-dimensional grid [17][18]. Symphony uses Kleinberg model to build a ring-structured P2P network [20]. Mercury supports multi-attribute range query by building a Symphony-based ring [4]. Semantic small world (SSW) supports semantic-based query by organizing data objects according to their semantics. Through dimension reduction on semantic space, SSW self-organizes into a linear small world.

DHT can only support exact-match, the other structures such as tree-based, skip-list-based and small-world-based structures can support complex query. In DHT networks, the conversion from keywords to data objects ID uses the consistent hash function, which makes data object IDs lose semantics and locality of keywords, thus incurring high cost for complex query. Namely, the conversion from keywords to data object ID indicates the query types that can be supported by P2P network. The other three topology methods allow keywords of data objects to be their IDs. The data object ID space is just their keyword space. So they can directly support complex keyword queries including range query, prefix query and multi-attribute query.

Two observations can be obtained. First, one topology construction method only supports one query type. This is because that the topology construction method determines the conversion from keywords to data object IDs (see Fig.1), thus determining the data organization method. A certain data management method is only appropriate for one query type. To support more query types, an effective way is to build an index to extend the data organization method since the index can organize pointers of data objects in a way different from the data organization on the topology.

Second, most structured P2P solutions can hardly support relational queries. Taking relations between two objects as a two-dimensional data space, a two-dimensional distributed indexing structure can be used to support relational queries. HRing allows the co-existence of multiple ID spaces and can directly support multi-dimensional index without using dimension reduction.

B. Semantic Link Network

A Semantic Link Network (SLN) is a directed network consisting of semantic nodes and semantic links [31]. Semantic nodes contain resources of various types. A semantic link between two nodes is a link with a tag directed from one node to another. The tag indicates the directed relation between two nodes. The tag can be of different types according to specific applications.

A semantic node can be a simple concept or a complex system. As illustrated in Fig.2, node A represents an e-learning system, and node G represents a student information system of a university. The e-learning system needs to regularly query student information to update student records. Then, node A interconnects node G with a *reference* semantic link. The original SLN model contains a set of basic semantic links and a set of reasoning rules on semantic links [31]. The distinguished advantage of SLN is that it is a self-organized data model and supports relational reasoning.

Our idea is to add tags on links to indicate the relations between nodes in P2P network. The relations between nodes are determined by the data objects stored on nodes. Since query messages are forwarded among nodes, discovering and building semantic links could improve search efficiency.

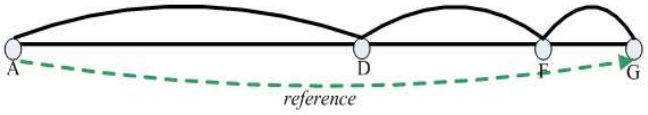

Fig.2. Semantic relation between two semantic nodes.

III. Semantic Data Model and Queries Of SemHRing

C. Semantic Data Model and the Supported Queries

As a semantic data model, Semantic Link Network SLN consists of a set of semantic links and a set of rules for relation reasoning on semantic links, represented as <{d—p→v}, *Rules*>, where d represents any data object named by URI, p represents an attribute or a relation on a subject, and v represents the value of p. If p is an attribute of d, then v is an attribute value, and if p is a relation, then v is a data object that has p relation with d. If d and p are default, v represents keyword.

Above data model supports the following types of queries.

(1) *Multiple keyword query*: input v_1 AND v_2 … AND v_k to obtain the data objects that contain these keywords.
(2) *Range query*: input $p \in [v, v']$ to obtain the data objects whose attribute p's value is within the interval.
(3) *Multiple attribute query*: input attribute p_1 AND p_2… AND p_k to obtain the data objects that commonly have these attributes and return the corresponding values.
(4) *Relational query type* 1: input a relation p to obtain the pairs of data objects (d_i, v_i) that have p relation.
(5) *Relational query type* 2: input two sets of attributes and their values $p_1 = v_1$ AND $p_2 = v_2$ …AND $p_k = v_k$; $p_1' = v_1'$ AND $p_2' = v_2'$ … AND $p_m' = v_m'$ to obtain the relations between the data objects with attributes $p_1 = v_1$ AND $p_2 = v_2$ …AND $p_k = v_k$ and the data objects with attributes $p_1' = v_1'$ AND $p_2' = v_2'$ …AND $p_m' = v_m'$.
(6) *Relational query type* 3: input a set of attributes and their values $p_1 = v_1$ AND $p_2 = v_2$ …AND $p_k = v_k$ and a relation p to obtain the data objects related with relation p and with these attributes and values.

B. Implementation Solution

We construct semantic links at two levels: among data objects and among P2P nodes.

A semantic link d—p→v can also be represented as a function $p(d) = v$ or a relational triple (d, p, v). SemHRing adopts RDF (Resource Description Framework) [24] to implement semantic link as relational triple, maps the RDF triples into a 2-dimensional distributed index, and then construct the index on HRing topology.

We assume that relations between data objects already exist. That is, triples have been extracted from data objects using existing techniques or are explicitly published by users. The task of SemHRing is to manage these triples in a uniform manner to facilitate complex queries. The key work is the design of the index structure and its deployment method on HRing.

Relations between nodes in P2P network are determined by data objects stored on them. SemHRing discovers and builds semantic links to indicate relations between nodes on HRing topology to improve search efficiency. However, HRing topology will be changed when adding semantic links, which will possibly increase the routing cost. Thus, the key is to add semantic links to the network with the guarantee of logarithmic routing hops and routing table size.

IV. The Underlying Topology

The underlying topology of SemHRing adopts the HRing [33]. The basic idea of HRing comes from the Harmonic Series [13]. The sum of the first n terms of the Harmonic Series $H_n = \sum_{k=1}^{n} \frac{1}{k}$ approximates to ln(n) + 0.5772156649... and the larger n is, the closer it gets to ln(n).

In HRing, the routing table of each node contains two short links pointing to its predecessor and successor, and O(ln(n)) long links pointing to its remote neighbors. We define the position distance *Dist*(A, B) between two nodes A and B as the number of "ring steps" from node A to node B in clockwise. As shown in Fig.3, *Dist*(A, E) = 4 indicates that node A reaches node E in four steps along the ring. The long link construction between two nodes is in reverse proportional to their position distance. When a new node C joins HRing, it visits the existing nodes along the ring clockwise and remote neighbors are added with the probability in inverse proportion to its traversal steps. So, based on the Harmonic Series, the routing table size of each node scales with $ln(n)$. HRing's ANEW process for long link construction guarantees that each node adds $O(ln(n))$ remote neighbors within $ln(n)$ steps [33].

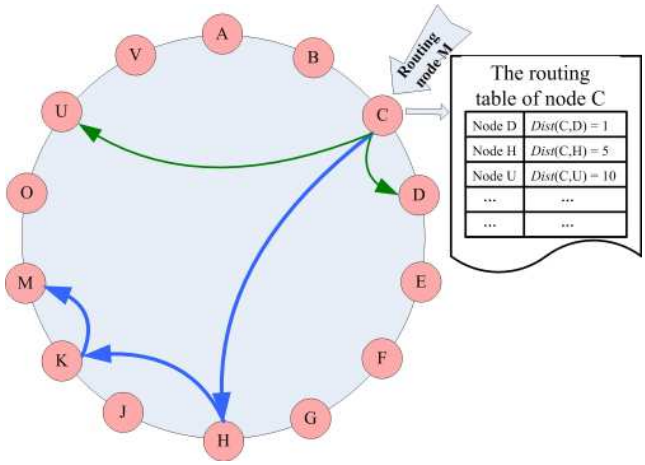

Fig. 3. Basic structure of HRing.

The prominent difference between HRing and the existing structured networks is that the long link construction in HRing is entirely independent of the ID space. In the existing structured networks, the node ID space is in charge of two tasks — building long links and performing routing, while in HRing, the ID space is responsible for only one task — performing routing. Building long links in the ID space will restrict the ID types, thus limiting the semantics of data objects in the network. For example, the long links in Chord are constructed in ID space, which requires the ID space to be computable and uniform [26]. Except for the numeric ID space, the other types of ID space cannot guarantee both computability and uniformity. But numeric ID space cannot express the semantics of data objects, which makes Chord difficult in supporting complex queries. In contrast, the long link construction in HRing does not rely on the ID space,

which enables the data object IDs and node IDs to be of any type, such as number, string, date, and IP address.

As a distributed platform to manage heterogeneous resources and support complex queries, HRing has the following four advantages:

1. Topology. HRing topology can achieve high performance and low maintenance cost while guaranteeing remarkable robustness.
2. Data management. The topology construction of HRing does not rely on data type and data management method. HRing does not aim at specific data types, specific data management methods, or specific query types. It can support the coexistence of heterogeneous resources.
3. Routing. The greedy routing algorithm in HRing guarantees logarithmic search performance. And, the semantic link in HRing can further improve search efficiency.
4. Complex query. Node IDs and data object IDs in HRing preserve the semantics of data objects. Moreover, HRing allows the co-existence of multiple ID spaces. So, it can support multiple types of complex queries.

V. 2-DIMENSIONAL DISTRIBUTED INDEX (2DDI)

This section introduces a 2-dimensional distributed index (2DDI) that supports both keyword queries and relational queries on HRing topology with high performance, scalability and robustness.

A. 2DDI Structure

Fig.4 illustrates the semantic data model for several heterogeneous data objects with their attributes, keywords and relations. A_1, A_2, A_3 and A_4 can be XML documents or Web pages of authors. Each author has two attributes: *Name* and *Affiliation*. The *Name*'s values are *Jack*, *Mary* and *Alice*. A_2 and A_4 have the same name. The *Affiliations*' values of A_1 and A_2 are *Tsinghua*, and the *Affiliations*' values of A_3 and A_4 are *ICT*. *Colleague* relation exists between A_1 and A_2, and between A_3 and A_4. *Teacher-student* relation exists between A_2 and A_3. P_1, P_2 and P_3 are three papers, their *Title*'s values are *Chord*, *Tapestry* and *Pastry*. They have the same attribute *Keyword P2P*. A_1 is the *Author* of P_1, and A_2 is the *Editor* of P_1. A_2 is the *Author* of P_2, and A_3 is the *Editor* of P_2. A_3 and A_4 are the *Co-authors* of P_3. P_1 is published in conference C_1 whose *Name* is *SIGMOD*. P_2 and P_3 are published in conference C_2, *WWW*. Observe that A_i (i = 1, 2, 3, 4), P_j (j = 1, 2, 3), and C_k (k = 1, 2) are semi-structured, unstructured and structured data objects respectively.

Users' input may be keywords, attributes and relations. Fig.5 shows a 3-dimensional resource space corresponding to the triple of our semantic data model, where dimension *v* denotes the linear keyword space, dimension *p* denotes the linear space of relations and attributes, and dimension *d* denotes the data object space. By using keywords, attributes or relations, users can locate one set of desired resources. Since there is no mapping between attributes and relations, it does not necessarily to use another two dimensions for attribute and relation respectively. They can be merged into one dimension. So, the semantic data model can be put into a simplified 2-dimensional index (2DDI). Table 1 is the 2DDI structure corresponding to the 3-dimensional resource space. All keywords in the column, and all attributes and relations in the row are linearized. The linear space formed by attributes and relations is a kind of relation space since attributes can be taken as the relations between data objects and keywords. The linear space formed by keywords is a kind of keyword space.

In table 1, attribute r_1's value of data objects d_1 and d_2 is v_2. (d_u/r_k, d_v) corresponds to r_s and v_k, which means that d_u and d_v has r_s relation, and the attribute r_k's value of d_u is v_k. Attributes, values and relations of data objects can be of any type such as string, number, date and IP address. HRing supports the co-existence of different types of the relation space and the keyword space as long as they can be linearized by a predefined order.

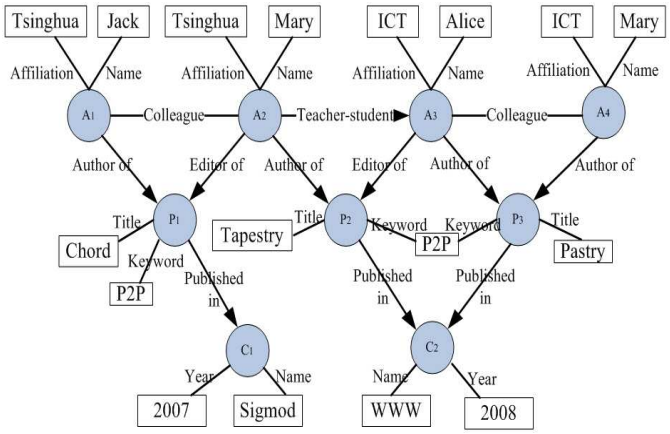

Fig.4. An example of the semantic link network.

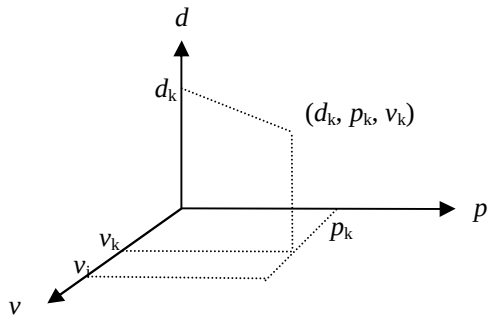

Fig. 5 The 3-dimensional resource space that supports complex queries

Table 1. The structure of the 2-dimensional distributed Index (2DDI).

relation space / keyword space	r_1	r_2	...	r_s	...	r_m	...
v_1							
v_2	d_1, d_2						
...							
v_k		(d_3/r_1, d_5)		(d_u/r_k, d_v)			
...							
v_n						(d_i/r_i, d_5)	
...							

Table 2. An example of 2DDI.

relation space / keyword space	Affiliation	AuthorOf	Colleague	EditorOf	Keyword	Name	Published in	Teacher-student	Title	Year
2007										C_1
2008										C_2
Alice		(A_3/Name, P_3)	(A_3/Name, A_4)	(A_3/Name, P_2)		A_3				
Beijing										
Chord							(P_1/Title, C_1)			
ICT	A_3, A_4								P_1	
Jack		(A_1/Name, P_1)	(A_1/Name, A_2)			A_1				
Mary		(A_2/Name, P_2), (A_4/Name, P_3)	(A_2/name, A_1), (A_4/name, A_3)	(A_2/Name, P_1)		A_2, A_4		(A_2/Name, A_3)		
P2P					P_1, P_2, P_3					
Pastry						C_1	(P_3/Title, C_2)			
Sigmod									P_3	
Tapestry							(P_2/Title, C_2)			
Tsinghua	A_1, A_2								P_2	
WWW						C_2				

Table 2 is the 2DDI structure built according to the semantic data model in Fig. 4. The data objects whose affiliation value is ICT are A_3 and A_4. The pairs of data objects that have AuthorOf relation are (A_1, P_1), (A_2, P_2), (A_3, P_3) and (A_4, P_3). The form of (A_3/Name, P_3) has two meanings: A_3 and P_3 have the AuthorOf relation; and A_3's Name is Alice. So, 2DDI can clearly express the mapping between attributes and attribute values, as well as data objects and relations.

2DDI allows search along both spaces separately or simultaneously according to specific query types. Search solely along the keyword space can answer multiple keyword queries. Search solely along the relation space can answer multiple attribute queries. Search along both spaces can answer relational queries.

Query Examples

(1) Multiple keyword query.
Input *Sigmod AND 2007* in keyword space
Output C_1 containing the two keywords.

(2) Range query.
Input *Year* $\in$ [*2007, 2008*] in keyword space and relation space
Output C_1's and C_2's *Year* is within that interval.

(3) Multiple attribute query.
Input *Name and Affiliation* in the relation space
Output A_1, A_2, A_3 and A_4 having *Name* and *Affiliation*, where A_1's *Name* is *Jack*, A_3's *Name* is *Alice*, and A_2 and A_4 have the same *Name Mary*, the *Affiliation* of A_1 and A_2 is *Tsinghua*, and the *Affiliation* of A_2 and A_4 is *ICT*.

(4) Relational query type 1.
Input *Teacher-student* in relation space
Output A_2 and A_3 have this relation, and A_2's *Name* is *Mary*.

(5) Relational query type 2.
Input $Name_1$ = *Jack* AND $Name_2$ = *Mary* in keyword space and relation space
Output A_1's *Name* is *Jack*, and A_2's and A_4's *Name* is *Mary*. A_1 and A_2 is *Colleague*, and there is no relation between A_1 and A_4.

(6) *Relational query type* 3.
Input *Name* = *Mary* AND *Affiliation* = *ICT* and a relation *Colleague* in keyword space and relation space.
Output A_4's *Name* is *Mary*, *Affiliation* is *ICT*, A_3 has *Colleague* relation with A_4, and A_3's *Name* is *Alice*.

Observe that 2DDI is able to differentiate the relations of data objects that have the same attributes and attribute values.

Compared with the Dataspace index in P2P context [7], 2DDI has the following two advantages:

1. The Dataspace index is designed for centralized personal information service. Data objects are linearized into the row, while the column is keywords with its corresponding attributes and relations. Such a structure is not appropriate for deployment and search in distributed systems. In contrast, 2DDI is specially designed for distributed networks. Data objects are stored in nodes of their own, and 2DDI only maintains the mappings of attributes, values and relations of data objects.
2. The row in Dataspace index is data objects, so search is only performed in the column. But the column only consists of keywords and its corresponding attributes and relations, so users cannot independently use attributes or relations to search data objects like relational query type 1-3. In contrast, 2DDI can perform

search process in both dimensions, and support keyword and relational queries.

In distributed systems, the index maintenance is more complex than that in centralized systems due to the operations of node's join, departure and failure, as well as data object update and backup. The following will introduce the construction method and the maintenance method of 2DDI on SemHRing.

B. Construction and Maintenance of 2DDI on SemHRing

Besides holding its own resources, each SemHRing node is responsible for a range of index pointing to the resources on other nodes. Since keyword queries and relational queries are routed along the keyword space and relation space, so 2DDI should be divided along the two dimensions and distributed among SemHRing nodes. Thus, SemHRing needs two index ID spaces: the keyword ID space and the relation ID space. Correspondingly, each SemHRing node has two index IDs. One indicates the keyword range it manages, and the other indicates the relation range it manages. To preserve semantics and support heterogeneity of data objects, SemHRing allows index IDs to be in any type and of any length as long as both the ID spaces can be linearized by a predefined order.

Additionally, to build SemHRing on HRing topology, each node needs a network ID to organize nodes. Since P2P networks are an overlay of TCP/IP, the network IDs can be IP addresses or URIs. Nodes are arranged into a ring structure in order of network IDs. Note that network IDs are only used for new nodes to find their locations in SemHRing. The long link construction in SemHRing is independent of the network ID space and two index ID spaces.

Below we introduce the operations of the new node's join, load balance and backup processes. The "load" indicates the "index range" that each node manages. When a new node finds its predecessor and successor according to its network ID, the successor will transfer its half load of two index spaces to the new node. In order to enhance network robustness and improve search efficiency, neighboring nodes backup their two indexes with each other. Thus even if two neighbors fail simultaneously, HRing can still guarantee the integrity of index spaces. The two index IDs of each node are the maximum values of the managed index ranges. For example in Fig.6, the keyword index space and the relation index space are denoted as two lines respectively, where the dots denote the specific locations of certain keywords, attributes or relations. Node D, F and G are neighbors. Node D manages keyword range (k_1, k_3] and index range (r_1, r_3], while backups the keyword range (k_3, k_7] and index range (r_3, r_7] of node F. F manages ranges (k_3, k_7] and (r_3, r_7], and backups ranges (k_1, k_3] and (r_1, r_3] of node D as well as ranges (k_7, k_9] and (r_7, r_9] of node G. G manages ranges (k_7, k_9] and (r_7, r_9], and backups (k_3, k_7] and (r_3, r_7] of F. The keyword ID and relation ID of D is k_3 and r_3 respectively. Similarly, the two index IDs of F are k_7 and r_7, and the two index IDs of G are k_9 and r_9. When the new node E joins in between D and F, F gives its half index ranges (k_3, k_5] and (r_3, r_5] as well as the backup indexes of D to E. Meanwhile, F informs D and G to update their index backups. In this process, the index IDs of D, F and G are not changed, and E is given two index IDs k_5 and r_5 according to its managed index ranges. Observe that D, E, F and G are network IDs.

VI. SEMANTIC LINKS ON SEMHRING

The relations between data objects are managed by 2DDI, and building 2DDI on SemHRing can support complex queries. The relations between nodes are determined by the data objects they managed. Since query messages are forwarded among nodes, discovering and building semantic links among SemHRing nodes will further improve search efficiency. For example in Fig.2, when node A issues a query for the data on node G, A can directly visits G without through D and F.

A. Design of Semantic Links

In an HRing topology of size n, each node has $\ln(n)$ position intervals [33]:

$$I_i = \begin{cases} e^0 = 1 & i = 0 \\ \left(e^i, e^{i+1}\right] & i = 1, 2, ..., \lceil \ln(n) \rceil - 1 \end{cases}$$

$e^0 = 1$ indicates that the successor of a node is its direct neighbor. For a given node A, its ith position interval denotes a set of nodes that are at the ring-distance from node A larger than e^i but smaller than e^{i+1}. We have proved that as long as each node randomly selects a neighbor in each of its position intervals, then HRing topology can achieve the logarithmic routing table size and routing hops [33].

Based on Theorem 2 in [33], we can build semantic links among SemHRing nodes using the following method. Each node searches for the remote neighbors with certain relations in its $\ln(n)$ position intervals and build its semantic routing table. Taking Fig.2 as an example, node A can determine which position interval node G lies in according to their ring distance. Thus, A can build a semantic link to G and use it to substitute the existing routing table link in that interval. In this way, node A builds semantic long links without changing its routing table size, and the search efficiency between nodes can also be kept.

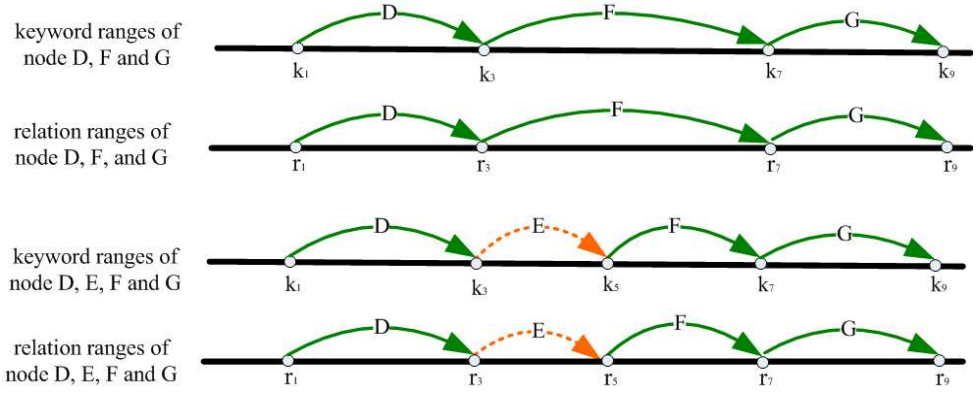

Fig. 6. An illustration of a new node's join process.

In dynamic networks where nodes continuously join, depart and fail, there are two occasions for nodes to build semantic long links. In the first occasion when a new node finds its location and joins, it visits its consecutive $\ln(n)$ nodes as well as their routing tables along the ring in clockwise, selecting $(\ln(n) - 1)$ nodes as its remote neighbors among those $\ln^2(n)$ nodes. Since the neighbors of those consecutive $\ln(n)$ nodes strictly follow the distribution of the position intervals, the distance between the new node and the neighbors of its kth consecutive node is $(k + e^i, k + e^{i+1}]$ where $k = 1, \ldots, (\ln(n) - 1)$ and $i = 1, \ldots, (\ln(n) - 1)$. As shown in the following equations, given k and i, the node at the distance within $(k + e^i, k + e^{i+1}]$ from the new node will be located in either of two neighboring intervals of the new node, thus it can be chosen as the neighbor in either of the two intervals. For example, when $k + e^i \le e^{i+1}$, the node at distance within $(k + e^i, k+e^{i+1}]$ from the new node will be in the $(i+1)$th or $(i+2)$th intervals of the new node, so the nodes in that two intervals can be added as the $(i+1)$th or $(i+2)$th remote neighbors. The following function can illustrate the above observations.

$$k + e^i \le e^{i+1} \Rightarrow k \le e^{i+1} - e^i$$
$$\Rightarrow \left(k + e^i, k + e^{i+1}\right] \subset \left(e^i, e^{i+1}\right] \cup \left(e^{i+1}, e^{i+2}\right]$$
$$e^{i+1} < k + e^i \le e^{i+2} \Rightarrow e^{i+1} - e^i < k \le e^{i+2} - e^i$$
$$\Rightarrow \left(k + e^i, k + e^{i+1}\right] \subset \left(e^{i+1}, e^{i+2}\right] \cup \left(e^{i+2}, e^{i+3}\right]$$
$$e^{i+2} < k + e^i \le e^{i+2} \Rightarrow e^{i+2} - e^i < k \le e^{i+3} - e^i$$
$$\Rightarrow \left(k + e^i, k + e^{i+1}\right] \subset \left(e^{i+2}, e^{i+3}\right] \cup \left(e^{i+3}, e^{i+4}\right]$$
$$\ldots$$
$$e^{i+\ln n-2} < k + e^i \le e^{i+\ln n-1} \Rightarrow e^{i+\ln n-2} - e^i < k \le e^{i+\ln n-1} - e^i$$
$$\Rightarrow \left(k + e^i, k + e^{i+1}\right] \subset \left(e^{i+\ln n-2}, e^{i+\ln n-1}\right] \cup \left(e^{i+\ln n-1}, e^{i+\ln n}\right]$$

For a given new node, $\ln^2(n)$ remote neighbors of its $\ln(n)$ consecutive successors are distributed in its $(\ln(n) - 1)$ position intervals. Each position interval may contain more than one remote node. When updating the remote neighbor in each interval, the new node will preferentially select the semantic node and build a semantic link. If no such semantic nodes exist, the new node will select and update neighbors by the inverse of the distance based on the Harmonic Series. According to Theorem 2 in [33], the necessary and sufficient condition to guarantee the logarithmic routing table size and routing hops is that each position interval of SemHRing nodes maintains one and the only one remote neighbor. Thus, once the new node finds a semantic neighbor in a certain position interval, it will stop finding the other semantic nodes even if there are other semantic nodes in that interval. The semantic long link construction algorithm ASNEW (Adds Semantic long links on New nodes) is illustrated in Fig.7. The major difference between ASNEW and the original HRing algorithm ANEW is that ASNEW will preferentially select semantic nodes as neighbors. If no semantic nodes exist, it will use ANEW to select neighbors.

The second occasion to build semantic links is during the search processes. Taking the queries from node A to node G in Fig.2 as an example, we present the construction process for semantic links. Through comparisons and validation in a query process, node A finds that only node G can exactly return its desired student information, thus inferring that A and G has a *reference* relation. By measuring the distance from A to G, node A determines the interval I_i that node G belongs to. After that, node A replaces the existing neighbor in I_i with node G, thus building the semantic link from A to G. Note that each interval is only permitted to have one neighbor regardless of the number of semantic neighbors in that interval. Semantic links are dynamic and updated according to query demands, obeying the least-recently-used algorithm (LRU). Fig.8 illustrates the semantic link constriction algorithm, in which the span of a link is a value v such that the length of the link is larger than e^v and less than e^{v+1}.

Theorem. In a SemHRing of size n, the expected average hops between any two nodes is $O(\ln(n))$.

Proof: we consider a SemHRing as a directed line of length n. The ith position interval of a given node denotes a set of nodes whose position distance to it is within $(e^{i-1}, e^i]$ for $i = 1, 2, \ldots, \ln(n)$, and e^0 is the first interval including only one node, i.e., the successor of node A. Each node has only one neighbor in each of its interval. Since the distance between any two nodes is less than n, we take the worst case where two nodes A and B with distance $(n - 1)$ as an example to compute the routing hops. Let n_0 be $(n - 1)$. Since node A has one link at each of its interval, the length of the kth jump

is larger than 1/*e* times the current distance n_{k-1} (k = 1, 2,…, ln(n)). After the kth jump, the remaining distance is $n_k < (1 - 1/2)\ n_{k-1}$. Then, by iteration, we can obtain $O(\ln(n))$ average routing hops.

```
/* Input: Given a new node A, CurrentVisitNumber denotes the number
   of visited nodes, VisitNumber denotes the total number of nodes that
   should be visited, Currentn means the estimated number of nodes in
   SemHRing. */
/* Output: node A's routing table */
Set a large value to the initial Currentn;
steps = 0;
CurrentVisitNumber = 0;
do{//visiting nodes along the ring in clockwise;
    steps ++;
    Visiting node Ni whose position distance from A is steps;
    CurrentVisitNumber ++;
    Currentn = min{Currentn, Currentn from Ni};
    VisitNumber = ln(Currentn);
    succ_add = AddNeighborBySteps(steps, Ni);
    If (succ_add == 0){//visiting the routing table of Ni
        for each neighbor Rij of Ni{
            Dist = position distance from A to Rij;
            v = ln(Dist); //Rij can be the vth neighbor of A
            //if A already has a neighbor Sv in Iv, existDist is the length
            //of the out-link
            if(existDist == 0){
                existDist = Dist;
                add Rij as the vth neighbor of A;
            }//end if
            else{
                if(Rij has a semantic relation with A){
                    replacing Sv in Iv with Rij;
                    existDist = Dist;
                }//end if
                else{//no relationship between Rij and A
                    //A choose Rij as the vth neighbor with the probability
                    //Adding_prob
                    Adding_prob = (1/Dist)/((1/existDist) + (1/Dist));
                    generating a random decimal x from [0,1];
                    if (x < Adding_prob){//choosing Rij
                        replacing Sv in Iv with Rij;
                        existDist = Dist;
                    }//end if
                }//end else
            }//end else
        }//end for
    }//end if
}while(VisitNumber ≥ CurrentVisitNumber)
```

Fig.7. Semantic long link construction on new nodes.

```
/* a node A issues queries for building semantic links, node B replies to A,
   A then adds B as a neighbor at a proper interval*/
/* Input: Given a node A, and its routing table RA, */
/* Output: node A's routing table RA*/
Dist = position distance from A to B;
v = ln(Dist); //B can be the vth neighbor of A
if( A has no neighbor in the vth interval Iv){
    add B as the vth neighbor of A;
}//end if
else{// if A already has one neighbor Nv in the vth interval Iv
    if( Nv has no semantic relationship with A){
        replacing Nv with B;
    }//end if
    else if (Nv has semantic relationship with A, but currently A needs the
            answers from A){
        replacing Nv with B;
    }//end else
}//end else
```

Fig.8. Semantic long link construction during search process.

B. Node Join, Departure and Failure

In a dynamic SemHRing, nodes continuously join, depart and fail. Nodes keep track of their neighbors by periodically probing them to guarantee the routing performance in SemHRing.

When a new node is to join a SemHRing, it should contact at least one existing node as its bootstrapping node, which helps it to find its position in SemHRing. A new node join procedure is equivalent to a query routing for the node ID, thus taking $O(\ln(n))$ hops. After the new node locates its own position, it begins to fix its routing tables. It preferentially adds the semantic nodes as its neighbors in each of its interval. The neighbor selection process is described in Fig. 7.

When a node A is to depart, it needs to transfer all resources, including the data objects, the ranges of keyword index and relation index, as well as its in-links and out-links, to other proper nodes in order to guarantee the integrity of the two index spaces and the efficiency of the routing performance and routing table size. The departing node A will consider preferentially transferring its resources to its successor C. Two situations will be considered.

```
/*Given a departing node A and its successor C, A transfers all its data as
well as its in-links and out-links to C. Then the in-link routing table size
and the out-link routing table size of node C have become 2ln(n).*/
/*Input: node C's in-link routing table Rin = {N1,N2,…,Nt} and out-link
routing table Rout = {M1,M2,...,Mt}, t = 2ln(n); Iin and Iout are two character
arrays with size ln(n) and initial values all being C */
/*Output: after the long link section, node C's in-link routing table
Rin = {N1,N2,…,Ns} and out-link routing table Rout = {M1,M2,...,Ms},
s = ln(n) */
//rebuilding in-link routing table of node C, i.e., the selection of ln(n)
in-links from 2ln(n) in-links;
for i = 1 to t{
    spanC(Ni) = in-link span from in-neighbor Ni to C;
    if (Iin(spanC(Ni)) == C)
        Iin(spanC(Ni)) = Ni;
    else
        if( Iin(spanC(Ni)) has no relationship with Ni)
            Iin(spanC(Ni)) = Ni;
}//for in-link selection
//rebuilding out-link routing table of node C, i.e., the selection of ln(n)
//out-links from 2ln(n) out-links;
for i= 1 to t{
    spanC(Mi) = out-link span from C to out-neighbor Mi;
    if (Iout(spanC(Mi)) == C)
        Iout(spanC(Mi)) = Mi;
    else
        if (Iout(spanC(Mi)) has no relationship with Mi)
            Iout(spanC(Mi)) = Mi;
}//for out-link selection
for i = 1 to s{
    Rin(i) = Iin(i);
    Rout(i) = Iout(i);
}
```

Fig.9. The long link update algorithm.

In the first situation, if receiving the resources of node A will not lead to the overload on its successor C, then C will receive all the resources of A. Thus, the number of in-links and out-links in C will increase to 2ln(n). Doubling the number of links does not benefit the routing scalability; instead, it will decrease the scalability of the routing table size. Thus, in this situation, we should consider how to

reduce the 2ln(n) in-links and out-links to ln(n). Moreover, C should choose only one link in each of its interval and it will preferentially choose the semantic links. The long link update algorithm for node C is illustrated in Fig.9.

In the second situation, if receiving the resources of node A will overload A's successor C, then A will globally select a lightly loaded node B, and transfer all its resources to B. The selected node B needs to leave the network and then rejoin as the successor of A by a new ID. In this way, node B can replace the position of A to manage all its resources and preserve the semantic integrity of nodes and data objects, thus guaranteeing the search efficiency. In the leave-and-rejoin process, B will transfer all the resources to its successor. So the precondition for B to leave and rejoin is that receiving the resources of B will not lead to the overload on its successor, which will avoid the recursion in the leave-and-rejoin process.

SemHRing uses the same strategy as HRing to deal with node failure problem [33] .

C. Semantic Link Maintenance.

In SemHRing, each node should have ln(n) out-links and ln(n) in-links in its ln(n) intervals. Only in this way, the long link distribution can be balanced. The long link construction process may make each node equipped with ln(n) out-links, but it cannot guarantee to equip each node with ln(n) in-links. In fact, as discussed in [33], when a new node joins, its long link adding probabilities for all existing nodes are not uniform. Thus, the long link construction for new nodes will incurs in-link skew among existing nodes. To balance the in-link distribution among nodes, an algorithm in SemHRing is proposed based on the in-link transfer in HRing. Each node will compare its number of in-links with its successor's. If it has more in-links than its successor, it will ask the successor to share some with it. Note that we do not transfer the semantic links in the in-link balancing process. The in-link transfer algorithm in SemHRing is described in Fig.10.

```
/*Given a node A and its successor S, as well as A's in-link routing table
  R_in, computing each in-link span to A and to S respectively, then
  selecting the in-neighbors that do not have semantic relationships
  to node A and transferring them to S*/
/*Input: node A, node S, and node A's in-link routing table R_in =
  {N_1,N_2,...,N_t} including t in-neighbors */
/*Output: the candidate in-neighbor set C_in = {N_k} that can
  be transferred from A to S */
for i = 1 to t{
  span_A(N_i) = in-link span from in-neighbor N_i to A;
  span_B(N_i) = in-link span from in-neighbor N_i to S;
}
j = 0;//a counter to record the size of the candidate in-neighbor set
for j = 1 to t{
  if(span_A(N_i) == span_s(N_i) && N_i has no semantic relation with A)
  C_in(j++) = N_i;
}
```

Fig.10. In-link transfer algorithm.

VII. LOAD BALANCE ON SEMHRING

SemHRing inherits the merit of HRing to support leave-and-rejoin load balancing without incurring uneven long link distribution. Thus, performing load balance in SemHRing will not affect its structure and the routing performance. Since semantic links exist between nodes, the load balance strategy in SemHRing should reserve the semantic relations as much as possible and reduce the transfer of semantic links. The leave-and-rejoin method will incur topology changes when nodes leave and rejoin and rebuild their routing tables. In contrast, neighboring load balance only needs to rearrange links between neighbors. Thus, the number of links transferred by neighboring load balance is fewer than that by leave-and-rejoin load balance.

The load balance algorithm in SemHRing combined the advantages of above two methods. It contains two stages. First, when a node B is overloaded, it will first connect its predecessor A and successor E to judge if they will be also overloaded after node B gives its half load to node A or node E. If one of them is not overloaded, then A will give its half load to it. If neither of A and E is able to receive B's load, then we adopt the leave-and-rejoin method to globally select a lightly-loaded node D. Node D first needs to leave the SemHRing with its links transferred to its predecessor and successor, then it rejoins the network and locates itself after node B using a new ID C. After that, B gives half load to C, and the semantic links pointing to the moved load should also transferred to C. So, B and C should rearrange its in-links and out-links to guarantee the logarithmic routing hops. Finally, C needs to construct its routing table as a new node does, which costs O(ln(n)).

VIII. SIMULATIONS AND ANALYSIS

This section uses simulations to analyze the influence of the semantic links on the routing table size, the routing hops as well as the long link distribution. We will compare the routing table size and the routing hops on HRing and SemHRing of the same size under both static and dynamic environments. We will first build a HRing topology, and then add semantic links on it and evolves it into SemHRing.

In a static network, the network size is fixed. Nodes do not depart or fail after they have completely joined and form a perfect network topology. The goal of simulating the static HRing and SemHRing is to study their efficiency and scalability in an ideal environment. We first build a group of static HRings of size n from 10^3 to 10^4, and use 26 English characters to randomly generate node IDs of fixed length 8. Nodes are organized into a ring according to the string order of node IDs. Each node adds long links to its routing table following the routing table construction method of HRing. For each node, we assume that there are 0 to ln(n) semantic nodes in HRing. We do not simulate the process of discovering semantic nodes since the discovery process relates to a specific application. Here, we only simulate the process of building semantic links after discovering the semantic nodes. Such a process is illustrated in Fig.7. Fig.11 shows that the average routing table sizes in HRing and SemHRing are almost the same, but the average hops in SemHRing are apparently less than that in HRings, thus indicating that building semantic links will improve search efficiency.

To simulate the real dynamic environments, we design a growing HRing and build long links on it by ANEW algorithm. Initially, there is only one node. When a new node joins, it will contact an existing node for bootstrapping. After locating itself, it adds long links by travelling the whole ring clockwise. Once finishing flooding, its routing table is established. After that, it will not consider subsequent new nodes and do not add new long links to them. For each node, we randomly select 0 to ln(n) nodes as its semantic nodes in HRing. We simulate the process of replacing the exiting links with the semantic links (see Fig.8).

Fig.12(a) shows that in the growing HRing and SemHRing of the same size, the average routing table size are almost the same. But under the circumstance of the same cost for the long link construction, the average routing hops in SemHRing is clearly less than that in HRing. Thus, it can be concluded that semantic links play a positive role on routing performance.

Below we analyze the out-link distribution and in-link distribution in a growing SemHRing. Fig.13 shows that the out-links exhibit poisson distribution, indicating that nodes have almost the same number of out-links, while the in-link distribution is skewed, indicating that most nodes only have a small number of in-links but a few nodes have large number of in-links. The skewed in-link distribution will incur the unbalanced workload in SemHRing such that a few of in-link-rich nodes will be much busier in handling queries because more out-neighbors pointing to them than to others. Fig.14 shows that after applying the in-link transfer method in SemHRing, the in-link distribution can be balanced. Fig.15 shows the comparisons of the average routing table sizes and the average routing hops in SemHRing before and after using in-link transfer algorithm. The simulation results show that the in-link transfer algorithm does not influence the scalability and efficiency of SemHRing.

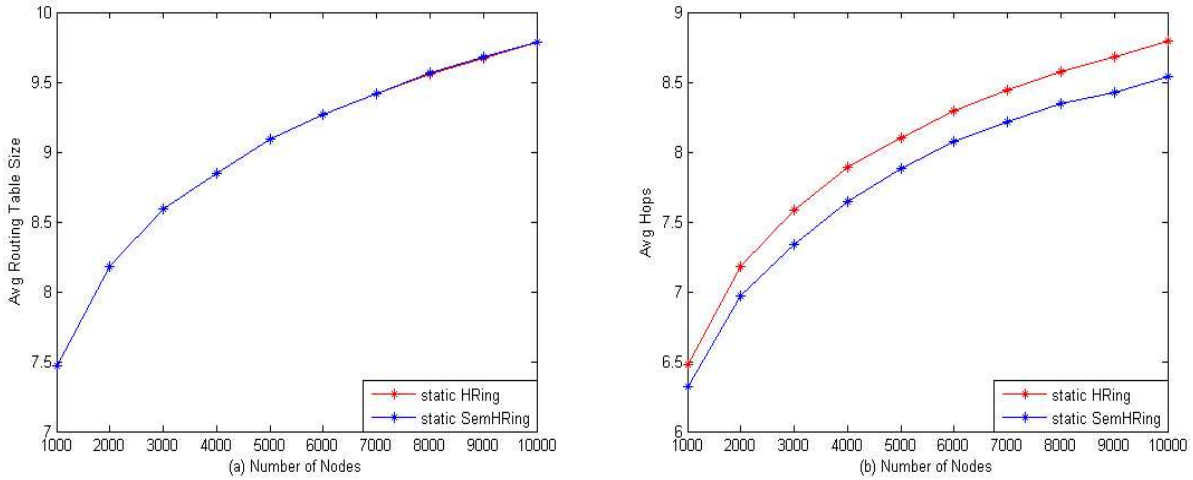

Fig.11. The average routing table size and the average hops in the static HRings and the static SemHRings.

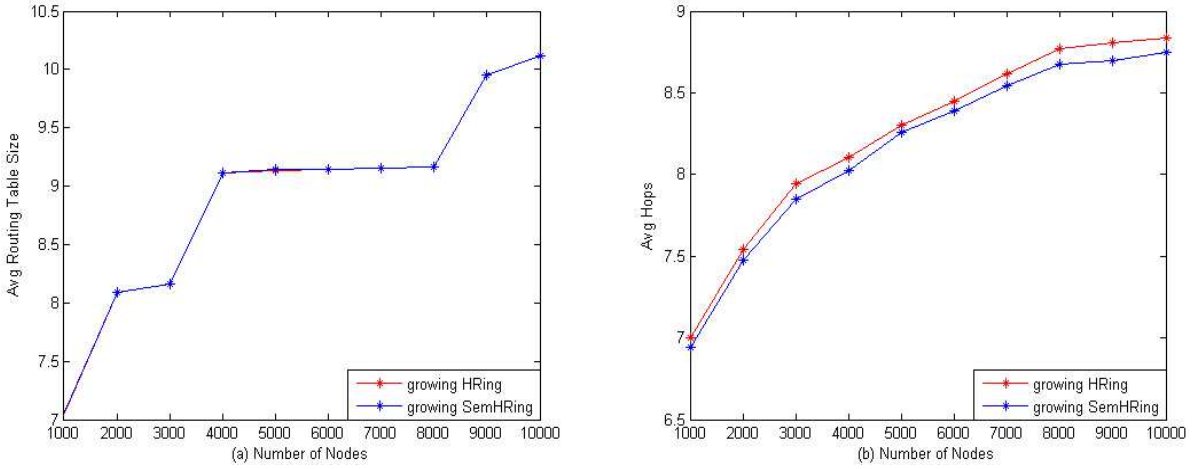

Fig.12. The average routing table size and the average hops in the growing HRings and the growing SemHRings.

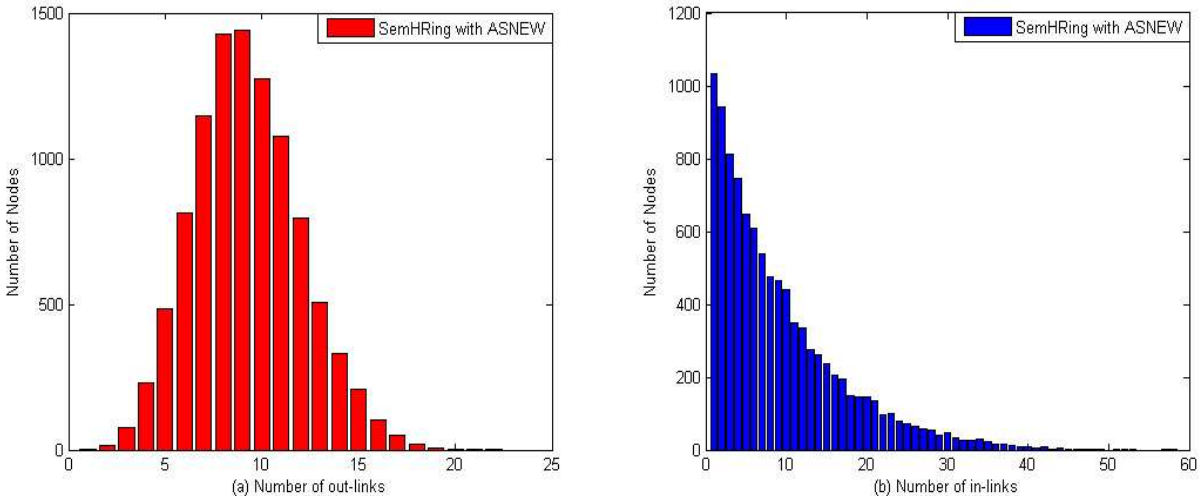

Fig.13. Out-link and in-link distributions in a growing SemHRing of size 10^4.

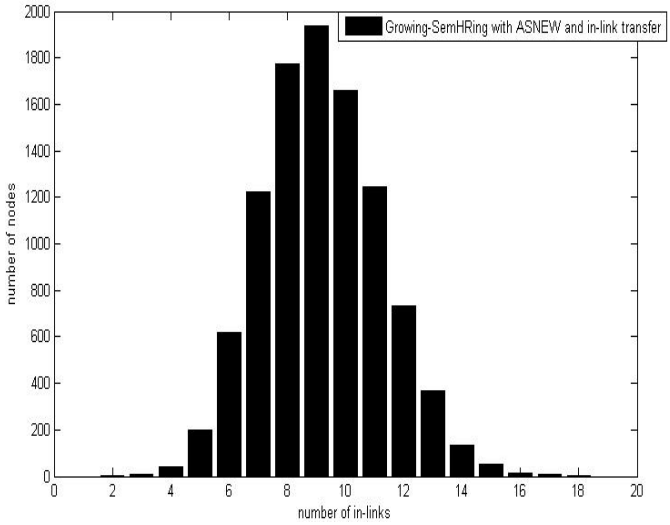

Fig.14. In-link distribution after in-link transfer in a growing SemHRing of size 10^4.

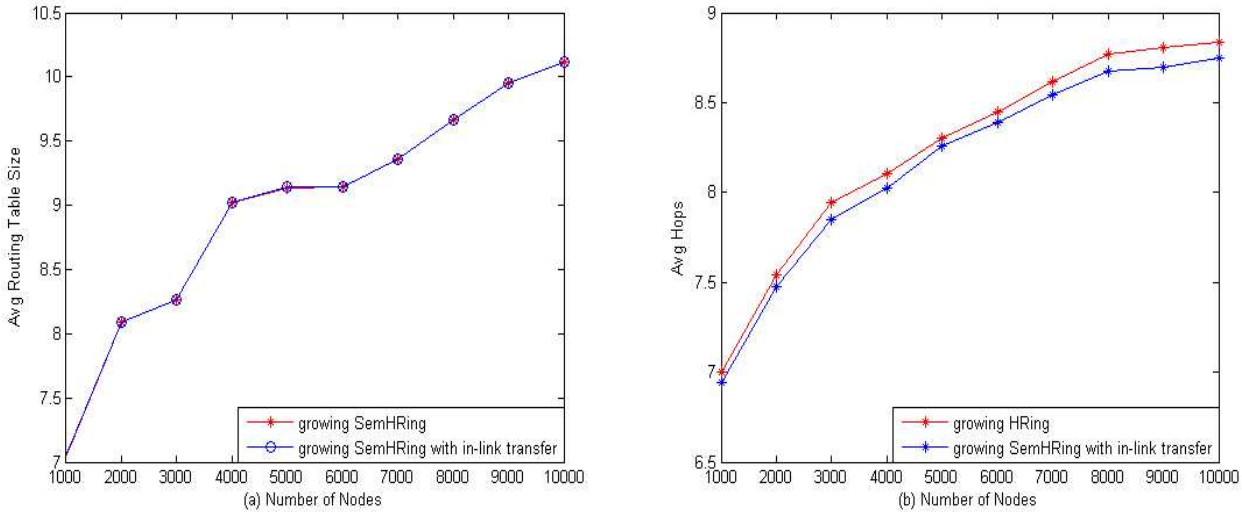

Fig.15. The average routing table size and the average routing hops in a growing SemHRing before and after in-link transfer.

IX. CONCLUSION

This paper proposes a structured P2P Semantic Link network SemHRing that uses the semantic link network as data model to support complex queries on decentralized heterogeneous resources and adopts the structured P2P topology HRing to ensure high performance and low maintenance cost. Two levels of semantic relations are considered — relations between data objects and between nodes. A two-dimensional distributed index 2DDI is built on an order-preserved structured P2P network with semantic links constructed among nodes to facilitate efficient relational queries. 2DDI enables SemHRing to organize attributes, keywords and relations on data objects in a uniform manner. Semantic links on SemHRing represent the semantic relationship between nodes, and optimize the topology of HRing to improve search efficiency. Experiments show that SemHRing is scalable, efficient and robust. SemHRing can support both relational and relational queries. It can be as a feasible solution to support efficient and decentralized semantics-rich queries.

REFERENCES

[1] K. Aberer, P. Cudre-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, R. Schmidt, "P-Grid: a self-organizing structured P2P system," In Proc. ACM SIGMOD/PODS conf., California, 2003, pp. 29-33.

[2] J. Aspnes, J. Kirsch and A. Krishnamurthy, "Load balancing and locality in range-queriable data structures," In Proc. 23rd Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), Newfoundland, Canada, Jul. 2004, pp.115-124.

[3] J. Aspnes, G. Shah, "Skip Graphs," In Proc.14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Baltimore, Maryland, Jan. 2003, pp. 384-393.

[4] A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: supporting scalable multi-attribute range queries," ACM SIGCOMM Computer Communication Review, 34(4), 2004, pp. 353-366.

[5] M. Cai, M. Frank, J. Chen, and P. Szekely, "Maan: A multi attribute addressable network for grid information services," In Proc. 4th Int. Workshop on Grid Computing (GRID), Phoenix, AZ, USA, Nov. 2003, pp. 184-191.

[6] A. Crainiceanu, P. Linga, A. Machanavajjhala, J. Gehrke, and J. Shanmugasundaram, "Querying Peer-to-Peer Networks Using P-Trees," In Proc. 7th Int. Workshop on the Web and Databases (WebDB), Paris, France, Jul. 2004, pp.25-30.

[7] X. Dong and A. Halevy, "Indexing Dataspaces", In Proc. Int. ACM SIGMOD, New York, 2007, pp.43-54.

[8] P. Ganesan, M. Bawa, and H. Garcia-Molina, "Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems," In Proc. 13th Int. Conf. on Very Large Data Bases (VLDB), Toronto, Canada, Sept. 2004, pp. 444-455.

[9] P. Ganesan, B. Yang, and H. Garcia-Molina, "One torus to rule them all: Multi-dimensional queries in P2P systems," In Proc. 7th Int. Workshop on the Web and Databases (WebDB), Paris, France, Jul. 2004, pp. 19-24.

[10] B. Gedik, L. Liu, "PeerCQ: A Decentralized and Self-Configuring Peer-to-Peer Information Monitoring System," In Proc. 23rd Int. Conf. Distributed Computing Systems (ICDCS), Providence, Rhode Island, May, 2003, pp.490 – 499.

[11] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman, "SkipNet: A scalable overlay network with practical locality properties," In Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS), Seattle, WA, Mar. 2003, pp. 113-126.

[12] N. J. A. Harvey and J. I. Munro, "Deterministic SkipNet," In Proc. 22nd ACM Symposium on Principles of Distributed Computing (PODC), pp. 152-153, 2003.

[13] J. Havil, F. Dyson, The Harmonic Series, Ch.2 in "Gamma: Exploring Euler's Constant," Princeton, NJ: Princeton University Press, 2003.

[14] M.Li, W.C. Lee, "Semantic small world: an overlay network for peer-to-peer search," In Proc. 12th IEEE Int. Conf. on Network Protocols (ICNP), Oct. 2004, pp.228-238.

[15] H. V. Jagadish, B. C. Ooi, Q. H. Vu, "BATON: A Balanced Tree Structure for Peer-to-Peer Networks," In Proc. 31st Int. Conf. on Very Large Data Bases (VLDB), Trondheim, Norway, Sept. 2005, pp. 661-672.

[16] D. Karger, M. Ruhl, "Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems," In Proc. 16th annual ACM symposium on Parallelism in algorithms and architectures (SPAA), Barcelona, Spain, Jun. 2004, pp. 36-43.

[17] J. Kleinberg, "The Small-World Phenomenon: An Algorithmic Perspective," In Proc. 32nd ACM Symposium on Theory of Computing (STOC), Portland, Oregon, USA, May, 2000, pp. 163-170.

[18] J. Kleinberg, "Navigation in a Small World," Nature 406, 2000, p.845.

[19] A. Kothari, D. Agrawal, A. Gupta, and S. Suri, "Range Addressable Network: A P2P Cache Architecture for Data Ranges," In Proc. 3rd Int. Conf. on Peer-to-Peer Computing, Sweden, Sept. 2003, pp. 14-22.

[20] G. S. Manku, M. Bawa, and P. Raghavan, "Symphony: Distributed hashing in a small world," In Proc. 4th USENIX Symposium on Internet Technologies and Systems (USITS), Seattle, Washington, USA, Mar. 2003, pp. 127- 140.

[21] C. Martel and V. Nguyen, "Analyzing Kleinberg's (and other) small-world models'," In Proc. 23th annual ACM symposium on Principles of distributed computing (PODC), Newfoundland, Canada, Jul. 2004, pp. 179-188.

[22] S. Ramabhadran, S. Ratnasamy and J. Hellerstein and S. Shenker, "Brief Announcement: Prefix Hash Tree," In Proc. 23th annual ACM symposium on Principles of distributed computing (PODC), Newfoundland, Jul. Canada, 2004, p.368.

[23] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker, "A scalable content-addressable network," ACM SIGCOMM Computer Communication Review. 31(4), 2001, pp. 161-172.

[24] RDF Primer, http://www.w3.org/TR/REC-rdf-syntax/.

[25] A. Rowstron, P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale Peer-to-Peer systems," In Proc. Int. Conf. on Distributed Systems Platforms (IFIP/ACM), Heidelberg, Germany, Nov. 2001, pp. 329-350.

[26] I. Stoica, R. Morris, D. L. Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan, "Chord: Scalable Peer-To-Peer Lookup Service for Internet Applications," In Proc. ACM SIGCOMM Conf., San Diego, Aug. 2001, pp.149-160.

[27] X. Wang, Y. Zhang, X. Li and D. Loguinov, "On Zone-Balancing of Peer-to-Peer Networks: Analysis of Random Node Join (extended version)," Technical Report, Texas A&M, June 2004.

[28] K. C. Zatloukal and N. J. A. Harvey, "Family trees: An Ordered Dictionary with Optimal Congestion, Locality, Degree, and Search time," In Proc. 15th ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, USA, Jan. 2004, pp.308-317.

[29] B. Y. Zhao, J. Kubiatowicz, A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," IEEE Journal on Selected Areas in Communications, vol.22, no.1, Jan. 2004, pp.41-53.

[30] Y. Zhu, H. Wang, and Y. Hu, "Integrating semantics-based access mechanisms with P2P file systems," In Proc. 3th Int. Conf. on P2P Computing, Sweden, Sept. 2003, pp.118-125.

[31] H. Zhuge, "The Knowledge Grid", World Scientific, 2004.

[32] H. Zhuge, X. P. Sun, J. Liu, E. L. Yao and X. Chen, "A scalable P2P platform for the Knowledge Grid", IEEE Transactions on Knowledge and Data Engineering, vol.17, no.12, 2005, pp.1721-1736.

[33] H. Zhuge, X. Chen, X. P. Sun and E. L. Yao, "HRing: a Structured P2P Overlay Based on Harmonic Series", IEEE Transactions on Parallel and Distributed Systems, vol.19, no.2, 2008, pp.145-158.

**Technical Report of Knowledge Grid Research Center*, KGRC-2008-03, 2008. www.knowledgegrid.net/TR