2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks

SARIDS: A Self-Adaptive Resource Index and Discovery System

Yi-Hsiang Lin[†] Wu-Chun Chung[‡] Kuan-Chou Lai[¥] Kuan-Ching Li[§] Yeh-Ching Chung[‡]

[†] Institute of Information Systems and Applications, National Tsing Hua University, Hsinchu, Taiwan, R.O.C.
[‡] Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, R.O.C.
[¥] Department of Computer and Information Science, National Taichung University, Taichung, Taiwan, R.O.C.
[§] Department of Computer Science and Information Engineering, Providence University, Taichung, Taiwan, R.O.C.
Email: {[†]yslin, [‡]wcchung}@sslab.cs.nthu.edu.tw, [¥]kclai@ntcu.edu.tw, [§]kuancli@pu.edu.tw, [‡]ychung@cs.nthu.edu.tw

***Abstract*—Recently, the resource sharing systems apply the P2P technique to provide scalable multi-attribute range queries. However, due to the heterogeneity of resources and the variation of sharing policies in different providers, current P2P-based resource discovery systems may suffer the load imbalance problem in a large scale distributed system. In this paper, we propose a self-adaptive resource index and discovery system (SARIDS) to achieve load balancing. SARIDS adopts a two-tier architecture based on the structured P2P overlay. The intra-overlay is constructed by normal peers with the same attribute via the locality preserving hash function; and, the inter-overlay is constructed by super-peers with classified attributes in different intra-overlays. SARIDS supports not only the multi-attribute range queries but also the self-adaptive mechanisms for load balancing in the intra-overlay and among the intra-overlays. The simulation results show that SARIDS is scalable and efficient for load balancing even in the non-uniform peer range environment.**

Keywords: Grid, Resource discovery, Multi-attribute query, Range query, Load balancing

I. INTRODUCTION

The large scale distributed resource sharing systems such as the Grid or Peer-to-Peer (P2P) computing system provide mechanisms for sharing and accessing mass heterogeneous resources [6]. To discover resources satisfying user requirements efficiently is a key service in large scale distributed resource sharing systems.

Traditional approaches use a centralized server or a set of hierarchical servers to index resource information. Through these servers, users can find resources matching their requirements. However, the centralized or hierarchical approach has some limitations in terms of scalability, fault tolerance, and self-organizing. To overcome these drawbacks, the decentralized model [12, 17] is proposed for indexing and discovering resource information, e.g., the Peer-to-Peer manner.

The P2P strategy can be classified as unstructured and structured. The unstructured P2P networks, such as Gnutella [4] and KaZaA [9], usually search resources by flooding messages with Time-To-Live (TTL). However, the message flooding results in a large volume of unnecessary network traffic. On the other hand, the structured P2P networks, e.g., Chord [7], CAN [16], Pastry [14] and Tapestry [3], use a distributed indexing service which is based on the Distributed Hash Table (DHT). The structured P2P systems are more scalable than the unstructured ones; and, the structured approach guarantees to find the requested resource in logarithmic bounds if the resource exists. Because the DHT looks up a resource exactly matching the given key, supporting complex queries (e.g., multi-attribute range queries) is the weakness of the structured approach.

In order to support complex queries in the structured P2P network, some resource discovery systems [1, 10, 11, 15] were proposed. However, they don't consider the impact on the load imbalance. The causes of the load imbalance include: a) the various attribute categories which caused by the different sharing policies or the resource heterogeneities, and b) the distribution of an attribute value isn't uniform because some values of an attribute are common and popular. Therefore, it is a challenge to provide a distributed resource sharing system for load balancing.

In this paper, we propose a self-adaptive resource index and discovery system, named SARIDS, to support multi-attribute range queries and to achieve the load balancing. The SARIDS is a two-tier architecture based on the structured P2P overlay. The intra-overlay is constructed by normal peers while the inter-overlay is constructed by super-peers. Normal peers in the intra-overlay index the values of the same attribute. Each super-peer acts as an entry point of an intra-overlay for a specific attribute. When a query or publishing information is issued, the query or publishing information routes in the inter-overlay according to the attribute, and then locates the specific range in the intra-overlay depending on the attribute value.

To support the multi-attribute range queries, the randomizing hash function and the locality preserving hash function are adopted to map the attribute information to specific normal peers. Moreover, since the resource attribute category is arbitrarily changed, this paper introduces an adaptive mechanism for dynamically constructing and destructing the intra-overlay. To overcome the non-uniform data distribution among peers and the load imbalance, we also propose load balancing mechanisms to balance the load of normal peers in the intra-overlay and redistribute normal peers among different intra-overlays.

To verify the performance of our proposed approach, this paper evaluates the routing performance, the improvement of

978-0-7695-3908-9/09 $26.00 © 2009 IEEE
DOI 10.1109/I-SPAN.2009.21

load balance, and the impacts of various workloads on SARIDS. The experimental results prove that the SARIDS is efficient and could improve the load balancing.

The remainder of this paper is organized as follows. In Section II, we discuss related works. Then, we briefly describe the system overview in Section III and give the detailed design of the SARIDS in Section IV. The simulation results are presented in Section V. We give final conclusions in Section VI.

II. Related Work

Many P2P-based resource sharing schemes have been proposed in the literature to overcome the shortcomings of centralized or hierarchical approaches. Recently, the structured P2P network model is widely adopted for resource discovery approaches in large scale distributed systems. Many structured P2P approaches [1, 10, 11, 15] use the locality preserving hash function to support multi-attribute range queries.

In the MAAN approach [10], one DHT is constructed for each attribute and all of these DHTs are mapped onto the same overlay. However, the MAAN knows the distribution of the attribute values in advance for balancing the load among peers. In the SWORD strategy [11], each attribute is assigned a different sub-region of a common DHT. However, the load balancing mechanism in SWORD is less efficient in an environment with a non-uniform distribution of peer ranges. In the Mercury approach [1], one DHT is constructed for each attribute, and each of these DHTs is randomly mapped to a subset of peers in this system. However, the Mercury doesn't support the functions of adding and removing attributes in the system and doesn't consider the load balance among intra-overlays. In the Squid strategy [15], one DHT is constructed for all attributes and the Squid uses the dimensionality reducing indexing scheme to map the multi-dimensional information space to physical peers while preserving the lexical locality and supporting complex queries. However, when the dimensionality of the information space is greater than five, the locality preserving property of Spacing Filling Curve (SFC) begins to deteriorate and the querying response time increases.

Compared to the aforementioned, SARIDS can dynamically adapt the load balancing according to the real situation of system load. Besides, SARIDS can balance the load even in the environment with a non-uniform distribution of peer ranges, support the functions of adding and removing attributes in the system, and consider the load balance among intra-overlays.

In this paper, we propose two load balancing mechanisms for our two-tier architecture in the large scale resource sharing system with different sharing policies and heterogeneous resources. The intra-overlay load balancing mechanism adopts a leave-rejoin-style protocol which is similar to that in [8, 13]. However, this protocol only works well when it can contact each peer randomly. When the distribution of peer ranges is highly-skewed in an overlay, this approach is far from trivial. The previous study [8] proposes two solutions to solve the above problem. One is to enable every peer to maintain a virtual peer without storing any data item and without changing its position. Then, the proposed mechanism [8] can select a random peer by a random ID and route this random ID to the corresponding virtual peer. The other is to use the random skip list as an alternate routing infrastructure, such as Skip Graphs [2] or SkipNet [5]. However, these approaches need to maintain an additional overlay with extra cost. In this paper, we propose a leave-rejoin-style protocol based on a random-walk for the intra-overlay load balancing. This mechanism works well even in an environment with highly-skewed peer range distribution, and brings no extra overhead.

III. SARIDS Overview

This section presents an overview about the SARIDS. We begin with a briefly introduction of the system architecture before describing the detailed mechanisms.

A. System Architecture

The SARIDS is constructed by a two-tier architecture, as shown in Fig. 1. The inter-overlay is constructed by super-peers while the intra-overlay is constructed by normal peers. A super-peer represents an entry for one intra-overlay. When a query or publishing information is issued, it starts to route in the inter-overlay for finding the super-peer with the specified attribute. Then, the request is routed to the dedicated intra-overlay through that super-peer and locates the normal peers with the matched values.

In SARIDS, each resource is described by a set of attribute-value pairs. Each pair is a form of (*attribute name*, *value*, *resource location*). For example, a pair (CPU_Speed, 2GHz, 192.168.1.11) represents the resource with 2GHz CPU speed. This pair describes the values of a resource attribute and the provider. Although we use the IP address as the resource location in this example, this field could be extended to present the other forms for the resource location.

A query is also described by a set of attribute-value pairs, such as (*attribute name*, *min value*, *max value*). Besides, each query has an additional field which is the number of results satisfying the user's requirement. This approach can also easily support the operators as follows: $\leqq$, $\geqq$, $<$, $>$ and $=$.

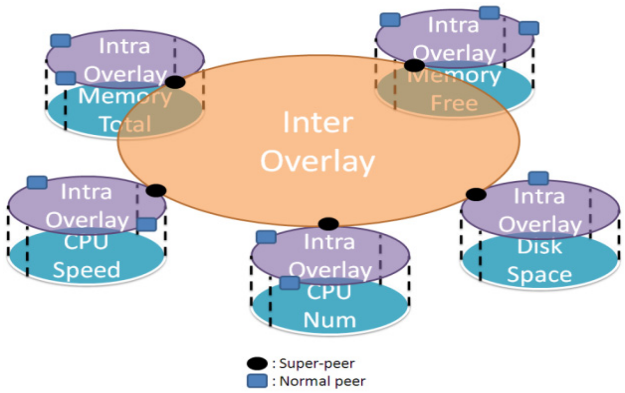

Figure 1. Overview of SARIDS overlay architecture.

B. Hash Function

In SARIDS, the ID of a peer or attribute information has three parts, as shown in Fig. 2. The first part represents the key of an attribute name applied in the inter-overlay. The second part represents the key of an attribute value. The third part is organized by random bits for distributing the popular values of an attribute to achieve load balance. The second part and the third part are applied in the intra-overlay.

The SARIDS applies the randomizing hash function and the locality preserving hash function. The randomizing hash function generates a key for the attribute name, such as SHA-1 (That is, every attribute name only has a unique hash key). Besides, the locality preserving hash function can generate keys of continuously attribute values along the intra-overlay to support range queries. Finally, SARIDS appends random bits to the tail of the key. While a peer maintains a more popular value of an attribute, this peer may result in the load imbalance. The proposed mechanism enables a popular attribute value to be queried and published among different peers in a load balancing fashion.

The locality preserving hash function in the SARIDS is similar to that in the MAAN [10]. However, when the distribution of attribute values is not uniform, it would produce a non-uniform distribution of hashing values, and lead to load imbalance. To address this problem, MAAN proposes a uniform locality preserving hashing function that can always produces a uniform distribution of hashing values. However, it only works if the distribution of input attribute values is known in advance. In SARIDS, we propose a random-walking-based leave-rejoin-style protocol for load balancing of the intra-overlay.

IV. Self-Adaptive Resource Index and Discovery

The discovery operation in SARIDS can be classified as the inter-overlay routing and the intra-overlay routing. For the explanation of our routing mechanism, let $Attr_Q$ denote the set of attributes in a query Q, $Attr_I$ denote the set of attributes in resource information I, and AO_a denote the intra-overlay for the attribute a. The query Q would be passed to one intra-overlay AO_i, where $i \in Attr_Q$. The attribute i is randomly chosen from $Attr_Q$. The resource information I would be published to all intra-overlays AO_j, where $j \in Attr_I$, to locate all the relevant resources in any of the intra-overlays.

Each normal peer within an intra-overlay AO_a maintains a continuous range r_a of attribute values. The function, f_a, produces the value or the range of a particular attribute a in a query or publishing. A normal peer in charge of the range r_a resolves a query Q for which $f_a(Q) \cap r_a \neq \phi$, and this peer stores resource information I for which $f_a(I) \cap r_a$. The intra-overlay routing is done as follows. The resource information I would be routed to the peer with the value $f_a(I)$. In the case of a query Q, $f_a(Q)$ produces values in a range. Hence, for routing queries, SARIDS routes Q to the random peer in $f_a(Q)$. Then, SARIDS uses the bidirectional searching method to resolve range queries.

The bidirectional searching method is a sequential searching solution to resolve the range queries in the intra-overlay and to balance the routing load among normal peers. SARIDS routes a random key of the requested range to the corresponding normal peer. Then, the query continuously lookups next value in the right-hand side of the corresponding normal peer in the intra-overlay. This discovery procedure continues until it meets the normal peer's ID which is larger than the hashing key with the maximum value in the requested range. If the number of requested results is not enough after routing the right-hand side, the query lookups the values in the left-hand side of the corresponding normal peer. This procedure stops when the number of requested results is sufficient or meets the next normal peer's ID which is less than the hashing key with the minimum value in the requested range. The number of requested results is defined by users for the bidirectional searching method to avoid visiting too many peers. Thus, this approach can discover the candidate resources in a sequential searching resolution with a limitation.

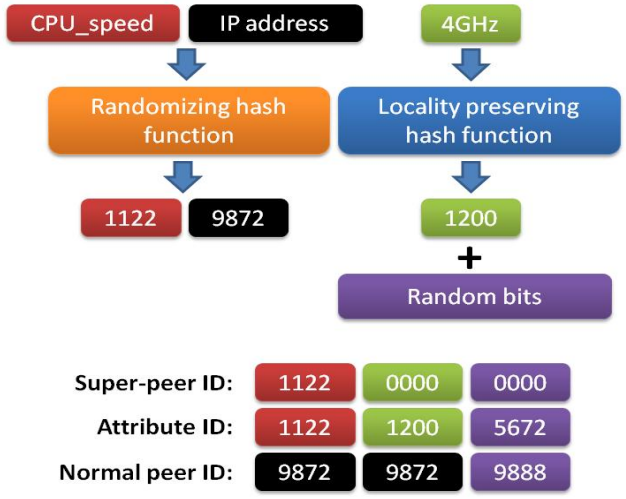

Figure 2. Example of super-peer ID, attribute ID and normal peer ID.

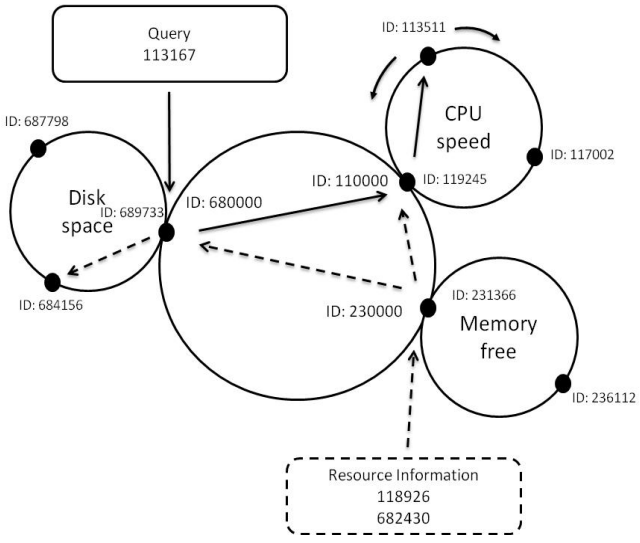

Figure 3. Example of query routing and information publishing.

Fig. 3 shows an example for our routing mechanism. The query with the key 113167 routes to the super-peer which is correspond to the CPU speed with ID = 110000 at first. Then, the query would be passed to the normal peer with ID = 113511. Then, the peer discovers its right-hand side and left-hand side for the requested range by the bidirectional searching method. On the other hand, when a peer tries to provide resource information, it would publish the information to all corresponding intra-overlays according to its providing information. In this case, the resource information would be passed to the super-peers with ID = 680000 and 110000, and is stored at the corresponding normal peers ID = 684156 and 119245, respectively.

A. Intra-overlay Construction and Destruction

Because the categories of attributes are different in distinct sharing policies, we propose an adaptive mechanism for constructing and destructing the intra-overlay. In general, the resource information is published to the corresponding intra-overlay. In the case of an attribute is only published or provided by very few providers, there is no corresponding intra-overlay. Because the load of the attribute is very light, this attribute is only stored temporality on a super-peer with a following attribute key. Therefore, this paper defines the temporary attribute as an attribute with a light load, and the primary attribute as an attribute for a dedicated intra-overlay.

To accommodate the dynamic construction and destruction of the overlay, each super-peer estimates the global average load ρ which denotes the number of data items and the parameter β which denotes the threshold of deciding the current load status of the intra-overlay. If the number of resource information of a temporary attribute in a super-peer is greater than $(1+\beta)\rho$, where $\beta < 1$, this attribute would be a primary attribute. Then, the super-peer would select a new super-peer from an under-loaded intra-overlay. This new super-peer maintains the new primary attribute and given a new ID by hashing this primary attribute name. The other loads of temporary attributes which IDs are less than this new primary attribute would also be transferred to this new super-peer. Then, the new selected super-peer joins the inter-overlay resulting in a new intra-overlay. When the load of partial temporary attributes is greater than $(1+\beta)\rho$, the super-peer only chooses a new super-peer as its predecessor and transfers these loads of temporary attributes to that predecessor.

In some cases, SARIDS would destruct the corresponding intra-overlay when the load of a primary attribute is vanished. If the total load of this peer is less than $(1-\beta)\rho$, the intra-overlay would be destructed. The load of this peer includes the load of the primary attribute and the load of temporary attributes. Then, this primary attribute would be degraded to a temporary attribute. All resource information in this peer would be transmitted to its successor in the inter-overlay. After the transferring operation uses the underlying Chord infrastructure, this peer rejoins the system.

B. Load Balancing

The SARIDS proposes an inter-overlay load balancing mechanism to redistribute peers among intra-overlays and also enhances the leave-rejoin-style mechanism to balance the load in the intra-overlay with a non-uniform distribution of peer ranges. We introduce the collection approaches for the load information in the SARIDS before describing the detailed load balancing mechanisms.

1) Clockwise Information Collecting Method: Each super-peer uses a clockwise information collecting method to gather the load information in its intra-overlay. This load information includes the total number of peers and the total number of resource information.

To gather the load information, a message is issued from a super-peer and then continuously passed to the successor in the intra-overlay until this message is sent back to the requested super-peer. This message computes the total number of peers and aggregates the total number of resource information in the intra-overlay. According to the collected information, super-peers can calculate the local average load and the global average load. Besides, this method is compatible with the stabilization process or the predecessor checking process of the Chord without extra overhead. When a peer joins or leaves the intra-overlay, it would notify its super-peer for precisely gathering the local average load. Then, the super-peer re-computes the local average load.

2) Inter-overlay Load Balancing Mechanism: The load balancing mechanism in the inter-over has two operations for generating the global average load by random sampling, and for redistributing peers in different intra-overlays according to the average local/global load.

In the first operation, each super-peer generates random keys and routes these keys to the corresponding super-peers. Then, these super-peers reply the load information of its intra-overlay. The experimental results show that the inter-overlay load balancing mechanism works well by sampling $\log N$ peers randomly. Our mechanism adopts the similar method in [1] to collect most recent $\log N$ global average load, where N is the size of inter-overlay. The global average load equals the total number of resource information divided by the total number of peers in the most recent samples.

In the second operation, SARIDS defines a bound β as the threshold to decide the load status of an intra-overlay. The β can help the system to bear dynamic peer joining and leaving, and to avoid unnecessarily redistributing a peer for resource information publishing and vanishing. To simply describe the status of an intra-overlay, let f_{LOAD} denote as the following:

$$f_{LOAD} = |LocalAvgLoad - GlobalAvgLoad| / GlobalAvgLoad \cdot$$

When f_{LOAD} is equal to or less than β, the load status of intra-overlay is balancing; when f_{LOAD} is larger than β and the local average load is less than the global average load, the load status of the intra-overlay is light; when f_{LOAD} is larger than β and the local average load is also larger than the global average load, the load status of the intra-overlay is heavy.

Each super-peer decides the load status in its intra-overlay. Then, the super-peer with the heavily loaded intra-

overlay would generate a random key and route this key to the corresponded super-peer in the inter-overlay. If this corresponding super-peer belongs to the lightly loaded or the balanced intra-overlay, then this super-peer would randomly choose a peer from its overlay to rejoin the heavy one to balance their load. Only super-peers with heavily loaded intra-overlays would periodically and randomly probe other super-peers.

3) Intra-overlay Load Balancing Mechanism: To balance the load in the intra-overlay, we enhance the leave-rejoin-style protocol to enable SARIDS to work well even in an environment with a non-uniform distribution of peer ranges. The SARIDS contacts a random-selected peer by a random-walking-based routing protocol. Each normal peer balances its load in an intra-overlay by this enhanced load balancing protocol. This approach routes the probing message with TTL by a successor list and a finger table of the Chord.

For describing our approach, let L_{local} denote the load on the requested peer, L_i denote the load of peer i that is met along the routing path and $C \times L_i$ denote the degree of load balancing where C is a constant coefficient. If $C \times L_i$ is equal to or larger than L_{local}, the requested peer leaves its current location to be a neighboring peer i. Then, the load of peer i is partitioned into two peers. Each intra-overlay has a good load balancing property by this enhanced leave-rejoin-style protocol for adapting to any resource distribution.

V. Experimental Results

To verify the performance of our system, we implement a simulator in JAVA and adopt the Chord to be our overlay structure. We evaluate the effectiveness of the routing mechanism and the load balancing mechanisms with different workload in the inter-overlay and the intra-overlay.

A. Routing Performance

Since the SARIDS applies the Chord as a routing infrastructure, the number of routing hops for finding the super-peer with the specified attribute in the inter-overlay is $\log N_1$, where N_1 is the size of an inter-overlay. On the other hand, the number of routing hops for locating the normal peer with the matched value is $\log N_2$, where N_2 is the size of the specified intra-overlay. Hence, the number of routing hops to publish an attribute is $\log N_1 + \log N_2$, and the number of routing hops to search resources is $\log N_1 + \log N_2 + k$, where k is the number of routing hops of the bidirectional searching method. The variable k is calculated as the number of requested results divided by the average load of a peer.

B. Sampling Analysis

In our system, each super-peer randomly probes other super-peers as samples and obtains their resource information. This information is gathered for balancing the load in the inter-overlay. In this experimental environment, we randomly assign peers and loads to each intra-overlay in a uniform distribution. Then, we generate 100 global average loads by different sample numbers to compute the standard deviation. Fig. 4 shows that when the number of samples is greater than $\log N$, where N is the number of intra-overlays, the sampling accuracy would not be influenced obviously.

C. Load Balancing

The experiment results of load balancing include the inter-overlay effect and the intra-overlay performance. In the inter-overlay load balancing experiment, the system assumes that there are 1,000 intra-overlays and each intra-overlay initials 100 normal peers. We setup the load to each intra-overlay in the Zipf distribution with α = 0.95. Then, every super-peer in SARIDS runs the inter-overlay load balancing mechanism to redistribute peers among intra-overlays. Fig. 5 depicts that SARIDS can balance the load of each intra-overlay effectively. We set the load status of each intra-overlay by assigning different bounds β of the inter-overlay load balancing mechanism. Obviously, the smaller bound would balance the system load more evenly.

In the intra-overlay load balancing experiment, we setup an intra-overlay with 1,000 peers and assign the load to each peer from 0 to 1,000 in a uniform distribution. Then, we set the value of C to be 0.5 and evaluate the effectiveness of our mechanism under a highly-skewed distribution of peer ranges (the α *of* Zipf distribution is 0.95). As shown in Fig. 6, our leave-rejoin-style protocol based on the random-walk can balance the system load in an environment with a non-uniform distribution of peer ranges.

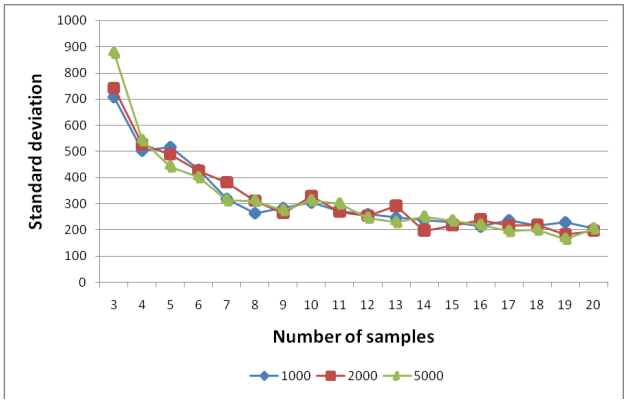

Figure 4. Sampling accuracy under different samples and network size.

β	0.5	1	1.5	Non-balanced
Std. deviation	2050.160118	2372.735969	2555.17836	20440.89531

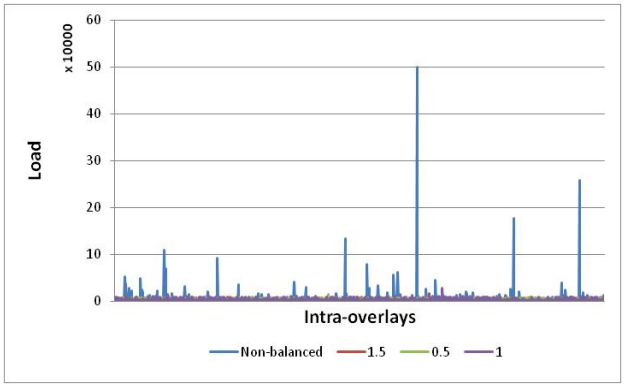

Figure 5. Effectiveness of inter-overlay load balancing under different β.

	SARIDS	Traditional	Non-balanced
Std. deviation	85.04614172	246.9494724	292.4238574

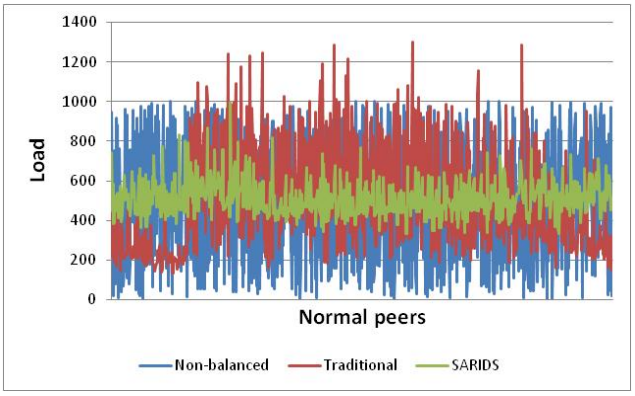

Figure 6. Intra-overlay load balancing in the non-uniform distribution of peer ranges.

VI. CONCLUSION

In this paper, we propose a self-adaptive resource index and discovery system, named SARIDS, for large scale distributed resource sharing systems with heterogeneous resources and different sharing policies. Our SARIDS not only supports the multi-attribute range queries but also balances the system load efficiently. In SARIDS, we propose a sampling-based mechanism to balance the load in the inter-overlay, and introduce a leave-rejoin-style protocol based on a random-walk for the intra-overlay load balancing. By these mechanisms, the load of each peer is almost equal in the whole system even in the environment with a highly-skewed resource distribution. Our simulation results show that SARIDS supports multi-attribute range queries with well scalability and good load balance, and works well even in a non-uniform distribution of peer ranges.

ACKNOWLEDGMENT

This work is partially supported by National Science Council of the Republic of China under contract NSC 96-2221-E-007-130-MY3 and NSC 97-3114-E-007-001. The authors would like to express their gratitude to the anonymous reviewers for their review and suggestions.

REFERENCES

[1] R. B. Ashwin, A. Mukesh, and S. Srinivasan, "Mercury: Supporting Scalable Multi-Attribute Range Queries," in *ACM SIGCOMM Computer Communication Review*. vol. 34, 2004, pp. 353-366.

[2] J. Aspnes and G. Shah, "Skip Graphs," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003, pp. 384-393.

[3] Y. Z. Ben, D. K. John, and D. J. Anthony, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," TR CSD-01-1141, University of California at Berkeley 2001.

[4] Gnutella, "http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf".

[5] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman, "SkipNet: A Scalable Overlay Network with Practical Locality Properties," in *Proceedings of the Fourth USENIX Symposium on Internet Technologies and Systems*, 2003, pp. 113-126.

[6] A. Iamnitchi and I. Foster, "On Fully Decentralized Resource Discovery in Grid Environments," in *Grid Computing — GRID 2001*, 2001, pp. 51-62.

[7] S. Ion, M. Robert, K. David, M. F. Kaashoek, and B. Hari, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," in *Proceedings of the ACM SIGCOMM*, 2001, pp. 149-160.

[8] D. R. Karger and M. Ruhl, "Simple Efficient Load-Balancing Algorithms for Peer-to-Peer Systems," in *Theory of Computing Systems*, 2006, pp. 787-804.

[9] Kazaa, "http://www.kazaa.com".

[10] C. Min, F. Martin, C. Jinbo, and S. Pedro, "MAAN: A Multi-Attribute Addressable Network for Grid Information Services," in *Proceedings of the Fourth International Workshop on Grid Computing*, 2003, pp. 184-191.

[11] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat, *Scalable Wide-Area Resource Discovery*: TR CSD-04-1334, EECS Department, University of California, Berkeley, 2004.

[12] R. Ranjan, A. Harwood, and R. Buyya, "Peer-to-Peer-Based Resource Discovery in Global Grids: A Tutorial," *IEEE Communications Surveys & Tutorials,* vol. 10, pp. 6-33, 2008.

[13] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load Balancing in Structured P2P Systems," in *Peer-to-Peer Systems II*, 2003, pp. 68-79.

[14] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Proceedings of the Middleware*, 2001, pp. 329-350.

[15] C. Schmidt and M. Parashar, "Squid: Enabling search in DHT-based systems," *Journal of Parallel and Distributed Computing,* vol. 68, pp. 962-975, 2008.

[16] R. Sylvia, F. Paul, H. Mark, K. Richard, and S. Scott, "A Scalable Content-Addressable Network," *ACM SIGCOMM Computer Communication Review,* vol. 31, pp. 161-172, 2001.

[17] P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi, "Peer-to-Peer Resource Discovery in Grids: Models and Systems," *Future Generation Computer Systems,* vol. 23, pp. 864-878, 2007.