Evaluation of P2P Resource Discovery Architectures Using Real-Life Multi-Attribute Resource and Query Characteristics

H. M. N. Dilum Bandara and Anura P. Jayasumana
Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523, USA.
dilumb@engr.ColoState.edu, Anura.Jayasumana@ColoState.edu

***Abstract*—Emerging collaborative Peer-to-Peer (P2P) applications rely on resource discovery solutions to aggregate groups of heterogeneous, multi-attribute, and dynamic resources that are distributed. In the absence of data and understanding of real-life resource and query characteristics, design and evaluation of existing solutions have relied on many simplifying assumptions. We first present a summary of resource and query characteristics from PlanetLab. These characteristics are then used to evaluate fundamental design choices for multi-attribute resource discovery based on the cost of advertising/querying resources, index size, and load balancing. Simulation-based analysis indicates that the cost of advertising dynamic attributes is significant and increases with the number of attributes. Compared to uniform queries, real-world queries are relatively easier to resolve using unstructured, superpeer, and single-attribute dominated query based structured P2P solutions. However, they cause significant load balancing issues in all the designs where a few nodes are mainly involved in answering majority of queries and/or indexing resources. Moreover, cost of resource discovery in structured P2P systems is effectively *O*(*N*) as most range queries are less specific. Thus, many existing design choices are applicable only under specific conditions and their performances tend to degrade under realistic workloads.**

Keywords- Multi-attribute queries; peer-to-peer; resource discovery; simulation

I. INTRODUCTION

Emerging collaborative Peer-to-Peer (P2P) applications thrive on interactions among groups of diverse and distributed resources to accomplish greater tasks beyond conventional file and processor cycle sharing. These systems utilize a variety of *resources* such as processor cycles, storage capacity, network bandwidth, sensors, data, and scientific algorithms to not only consume a variety of contents but also to generate, modify, and manage those contents. Collaborative Adaptive Sensing of the Atmosphere (CASA) [1], for example, is an emerging network of weather radars that collaborate in real time to detect hazardous atmospheric conditions. Collaborative P2P data fusion provides an attractive implementation choice for CASA real-time radar data fusion as data is constantly being generated, processed, and pushed and pulled among groups of radars, processing nodes and storage nodes. Community cloud computing [2], based on underutilized computing resources in homes/businesses, targets issues such as centralized data, proprietary applications, and cascading failures in modern cloud computing systems. Certain applications also benefit from a mixture of dedicated and voluntary resources that are spread across multiple cloud sites [3]. A collaborative P2P system is at the heart of such a multi-site or a community-cloud system that interconnects dedicated/voluntary resources. We refer to such systems as *P2P clouds*. A variety of applications related to CASA and P2P clouds require some form of resource collaboration. They need to aggregate complex resources as and when needed to meet the performance and Quality of Service (QoS) requirements. Timely aggregation of complex resources is becoming increasingly necessary even in conventional grid, desktop grid, and cloud computing due to the proliferation of parallel applications that utilize multiple resources. P2P-based distributed resource discovery is a natural fit to collaborative applications and further enhances their scalability and robustness. Yet, it is nontrivial to discover and utilize heterogeneous, multi-attribute, dynamic, and distributed resources.

Significant progress has been made in multi-attribute resource discovery in grid computing [4-5] and P2P [6-9]. However, compared to single-attribute P2P systems such as file sharing systems, formal characterization of real world, multi-attribute resources and queries has received attention only recently [10-11]. In the absence of data and understanding of characteristics, designs and evaluations of resource discovery solutions have relied on many simplifying assumptions such as independent and identically distributed (*i.i.d.*) attributes, uniform or Zipf's distribution of all the resources/queries [4, 8-9], and queries specifying a large number of attributes and a small range of attribute values [4-5]. Moreover, cost of updating dynamic attributes is ignored [4-9]. Such assumptions affect both the designs and performance analysis, and consequently the applicability of solutions under real workloads. For example, performance analysis in [9] is limited to point queries in structured P2P systems and extensively rely on aforementioned assumptions. Therefore, a formal comparison of fundamental design choices in P2P resource discovery with respect to behavior learned from actual systems is needed. Such an analysis will not only provide a better understanding of the applicability of existing solutions but also help identify best practices for resource aggregation in emerging collaborative P2P systems.

Our goal is to evaluate the fundamental design choices for multi-attribute resource discovery under real workloads. This is a continuation of our prior work in [11-12]. In [11], we analyzed real world, multi-attribute resource and query characteristic, and use those characteristics to qualitatively evaluate existing P2P resource discovery solutions. A tool that generates synthetic traces of multi-attribute resources while preserving statistical properties of real-world systems is presented in [12]. In this paper, we utilize the characteristic learned in [11-12] to

This research is supported in part by the Engineering Research Center program of the National Science Foundation under award number 0313747.

quantitatively evaluate multi-attribute P2P resource discovery solutions. A representative subset of designs based on centralized, unstructured, superpeer, and structured P2P architectures is considered. Design choices are evaluated based on the cost of advertising/querying resources, load balancing, scalability, etc. First, multi-attribute resource and query characteristic of PlanetLab [13] nodes are presented. The findings show that resource and query popularity is skewed, and queries are less specific where they specify only a few attributes and a large range of attribute values. Based on these observations, a set of simulators is then developed to evaluate fundamental design choices using real-life workloads. Analysis indicates that the cost of advertising dynamic attributes is significant and increases with the number of attributes, and hence should not be ignored in performance studies. Cost of resource discovery in ring-based structured P2P systems is effectively $O(N)$, as most range queries are less specific. Compared to uniform queries, real-world queries are relatively easier to resolve using unstructured, superpeer, and single attribute dominated query based structured P2P architectures. However, real-world queries introduce significant overhead in structured P2P solutions that utilize sub-queries. Furthermore, all the designs are prone to significant load balancing issues where few nodes are involved in answering majority of the queries and indexing resources. Thus, many existing design choices are applicable only under specific conditions and perform poorly under realistic workloads. To our knowledge, this is the first evaluation of diverse design choices for resource discovery using real-life workloads.

Section II reviews resource and query characteristics. Sections III and IV present the simulation setup and performance analysis, respectively. Concluding remarks are presented in Section V.

II. Resource and Query Characteristics

PlanetLab [13] is a global research network that provides a platform for testing networking applications, protocols, etc. by aggregating a globally distributed set of nodes. It reflects many characteristics of Internet-based distributed systems such as heterogeneity, multiple end-users/applications, dynamic nodes, and global presence, and therefore is being used to evaluate many preliminary P2P protocols and applications. Resource characteristics are analyzed using data from CoMon [14], which is a node and slice monitoring system for PlanetLab. SWORD [6], a multi-attribute resource discovery tool for PlanetLab, data is used to analyze the query characteristics. Here we present a summary of findings relevant to the following discussion; further details can be found in [11-12].

PlanetLab resources are represented using 46 attributes. 12 of those attributes are static (e.g., CPU speed and total memory) and the rest are dynamic (e.g., free CPU and transmission rate). Most static attributes follow a Gaussian distribution (e.g., CPU speed, total memory, and disk capacity). Dynamic attributes such as CPU load and transmission/receive rate and their rate of change fit a Generalized Pareto distribution. 32% of the nodes significantly changed their dynamic attribute values 100 times or more within a 24-hour period (based on 288 samples taken at five-minute intervals). 3% of the nodes changed their attribute values over 200 times within the same period. Thus, there is a wide variation in what attributes and how frequently the attribute values change. 16% of the nodes had identical static attributes such as CPU speed, memory, and disk capacity. Most PlanetLab sites also have several identical hosts. This is the case even in grid and cloud computing where sites may simultaneously deploy or upgrade to a similar set of machines. Furthermore, some of the resource attributes were highly skewed (e.g., free CPU). Several categorical attributes had only a few valid attribute values (e.g., CPU architecture and operating system). Distribution of those attributes is extremely skewed, and does not fit a standard distribution. Attributes were somewhat correlated, e.g., total memory, disk capacity, and free CPU positively correlated with the number of CPU cores [12]. However, correlation among dynamic attributes was lower as their attribute values depend on specific applications that are using those nodes.

SWORD enables users to query for groups of resources that are characterized by multiple-attributes. 80.5% of the queries specified only one or two attributes while seven was the maximum. This could be due to the nature of applications run in PlanetLab, which may not require fine-grained resource selection or users' inability to identify detailed resource requirements. 29% of the queries requested ten resources, 42% requested 50 or more resources, and 8% requested all the resources in the system. Thus, there is a tendency to request a large number of resources. Among the attributes specified in a query, dynamic attributes were predominant. Several combinations were frequently requested together, e.g., free CPU and disk space combination was requested 44% of the time. It was further observed that the range of attribute values specified in most of the queries is relatively large. For example, 89% of the queries requested free CPU value of 40-100% and 70% queries requested free disk space of 5-1000 GB. Thus, most queries are less specific. Furthermore, popularity distribution of queries is skewed; however, it does not follow a Zipf's-like distribution.

In summary, resource attributes are somewhat correlated and follow a mixture of probability distributions. Categorical attributes are highly skewed. Rate of change in dynamic attributes differ from one attribute/node to another, and some of the attributes changed frequently. These factors contribute to a large resource advertising cost that should not be ignored. Though resources are represented using tens of attributes, they are queried using only a few attributes. Queries are less specific as they specify 1-7 attributes, large range of attribute values, and request for a large number of resources. We believe that this trend will remain even in collaborative P2P applications where users may not be informed enough to issue very specific queries. These findings invalidate commonly used assumptions such as *i.i.d.* attributes, uniform/Zipf's distribution of all the attribute values, and queries requesting a large number of attributes and a small range of attribute values [4-5, 7-9].

III. Simulation Setup

We simulated seven representative architectures for resource discovery against the same set of resources and queries derived from PlanetLab. Use of realistic data preserves the complex distribution of attributes, dynamic and correlated changes in attribute values, and users' interest in resources. Table 1 summarizes the seven architectures. Specific details of architectures will be introduced in Section IV. To simplify performance analysis and eliminate any bias due to node failure, we replayed a trace with only the PlanetLab nodes that were continuously available for three days starting 2010/11/08. There were 527 such nodes. We were able to collect only 441 SWORD queries (collected between 2010/03/12 and 2011/04/17) due to the light usage of the system and server failure. Therefore, large number of synthetic queries was generated using the empirical distributions derived from the number of attributes in a query, popularity of attributes, range of attribute values, and number of resources requested by a query. To capture the correlation among attributes and attribute value ranges, conditional probabilities of attribute occurrences are also taken into account. Queries were issued only after the network was stabilized. Performance of ring-based structured P2P solutions were also evaluated for different numbers of nodes in range 250 to 1,000. A large number of resources/nodes, beyond 527 used for measurements, was generated using our correlation persevering multi-attribute resource generation tool [12].

Both the unstructured and superpeer based networks are generated using the B-A scale-free network generator [15] with a minimum node degree of two. Number of superpeers is set to 20. Maximum number of hops to forward a random walk (i.e., TTL) is set to 100 and 10 hops in unstructured and superpeer architectures, respectively. Those TTL values were sufficient to achieve ~70% query hit rate with different numbers of attributes. Multi-ring, partitioned-ring, and overlapped-ring architectures are based on Chord overlay [16]. Based on [8], the number of cell levels is set to three. Minimum update interval for resource attributes is five minutes, which is the sampling interval of CoMon. Each node issued queries based on a Poisson distribution with a mean inter-arrival time of 2.5 minutes (i.e., two queries per sampling interval per node). Results are based on eight samples with different random seeds.

IV. Performance Analysis

Fig. 1 shows the total cost of advertising and querying resources using only the 12 static attributes. This enables us to validate performance of different architectures against their prior performance studies (when such a study exists) that did

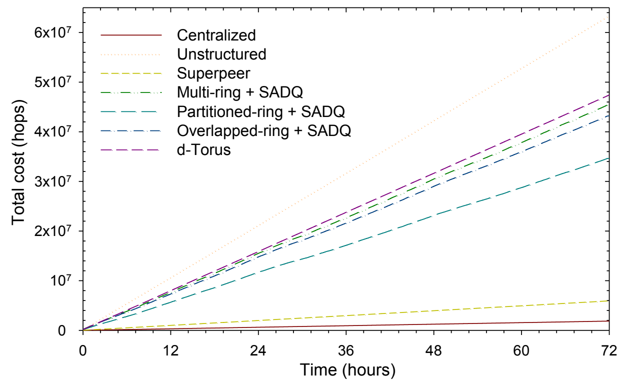

Figure 1. Total cost of advertising and querying static attributes.

not consider the cost of advertising dynamic attributes. As expected, centralized architecture has the lowest overall cost while unstructured P2P architecture has the highest cost. Indexing resources (i.e., storing attribute values of resources) at a central node is the simplest approach to resource discovery. Latency and cost of resource advertising and querying are minimum as nodes can directly communicate with the central node using one overlay hop regardless of the number of attributes of a resource or complexity of a multi-attribute range query. However, this solution is susceptible to single point of failure.

Superpeer-based architecture has the second lowest cost. It extends unstructured P2P systems to a two-layer overlay where resource rich nodes, namely *superpeers*, form a separate overlay while acting as proxies for rest of the nodes. Superpeers index resources of nodes directly connected to them and resolve queries on their behalf. When local resources are insufficient, superpeers forward queries to other superpeers using random walks. Cost of resource discovery is relatively lower as nodes can directly communicate with their superpeer, and only superpeers are involved in resource indexing and querying. Moreover, less specific queries make it easier to find resources by visiting few superpeers. For example, though random walks cannot guarantee to find all the resources, superpeer-based architecture was able to resolve 96% of the queries. Therefore, most of the random walks can terminate within few hops while

TABLE I – SUMMARY OF DIFFERENT RESOURCE DISCOVERY ARCHITECTURES USED IN SIMULATIONS.

Architecture	Routing Mechanism	Advertising		Querying		Indexing	
		Mechanism	Cost*	Mechanism	Cost*	Mechanism	Cost*
1. *Centralized*	Direct	To central node	1	Query central node	1	At central node	$O(AR)$
2. *Unstructured* – Random overlay	Random walk	Optional	–	Query random nodes	$O(TTL)$	Index locally	$O(A)$
3. *Superpeer* – Random overlay among superpeers	Random walk among superpeers	To superpeer	1	Query superpeers	$O(TTL)$	At superpeer	$O(AR)$
4. *Multi-ring* [7] – Separate ring-like overlay for each attribute type	Chord [16]	To relevant ring(s) based on attributes	$O(\log N)$	1. Multiple sub-queries 2. Single attribute dominated query	$O(\log N)$	1. At relevant rings 2. At all rings	1. $O(AR)$ 2. $O(A^2R)$
5. *Partitioned-ring* [4, 6] – Each attribute type is assigned a different segment of address space (overlay ring)	Chord [16] or Cycloid [4]	To relevant partition(s) based on attributes	$O(\log N)$	1. Multiple sub-queries 2. Single attribute dominated query	$O(\log N)$	1. At relevant partitions 2. At all partitions	1. $O(AR)$ 2. $O(A^2R)$
6. *Overlapped-ring* [5] – All attribute types are mapped to same ring-like overlay	Chord [16]	To relevant nodes in the ring	$O(\log N)$	Single attribute dominated query	$O(\log N)$	At relevant nodes	$O(A^2R)$
7. *d-Torus* [8] – *d*-torus partitioned into set of cells	Depth first search using neighbor cells	Not required	–	Visit cells that overlap with query region	$O\left(\frac{\ln 2^d}{\ln d}\right)$	Index locally	$O(A)$

* N – Number of nodes, A – Number of attributes, R – Number of resources, d – Number of dimensions, TTL – maximum hop count for random walk

reducing the overall cost.

Partitioned-ring-based architecture [4, 6] has the third lowest cost. This approach partitions the Chord ring into several segments such that each segment corresponds to a different attribute. Prefix bits of the overlay *key* is used to represent the attribute type and suffix bits represent the attribute values. Locality of attribute values needs to be preserved to support range queries. Thus, suffix bits are assigned using a Locality Preserving Hash (LPH) function [4-6]. Each resource advertises either each attribute value to the corresponding partition or all the attribute values to all the partitions that are used to index a resource. In the former case, a multi-attribute range query q is first split into set of *sub-queries* where each sub-query searches for one of the attributes specified in q. Sub-queries are then simultaneously forwarded to relevant partitions in the ring and are resolved by forwarding to all the nodes responsible for indexing the specified range of attribute values. Query results have to be then combined at the query originator using a *join* operation like in databases. Therefore, cost of resolving q is:

$$COST_q = \sum_{i \in q_A} \left(h_i + \left\lceil \frac{r_q^i}{r_{\max}^i} \times \frac{N}{A} \right\rceil - 1 \right) \quad (1)$$

where q_A is the set of attributes specified in q, h_i is the number of overlay hops to reach the node that indexes the minimum required value of the i-th attribute in q, and A is the number of attributes in the system. Typically, $h_i = O(\log N)$ [16]. After reaching the first node, queries that specify a range of attribute values need to be forwarded further. Therefore, additional number of nodes visited by a range query is proportional to the range of the i-th attribute value r_q^i and number of nodes N/A in a partition (assuming uniform node distribution), and inversely proportional to the maximum range of attribute value r_{max}^i. Given that range queries in real-world systems tends to be less specific (i.e., large r_q^i), query cost is effectively $O(N/A)$.

In the latter case, q can be resolved using only one of the partitions as each partition is aware of all the attribute values of a resource. Furthermore, query forwarding can be terminated as soon as the desired number of resources is found. Query resolution cost can be further reduced by forwarding q to the partition that corresponds to the most selective attribute, i.e., attribute with the lowest r_q^i/r_{max}^i ratio. Lower r_q^i/r_{max}^i is desirable as it reduces the number of hops that a query is forwarded (see Eq. 1). This approach is called *Single-Attribute Dominated Querying* (SADQ) and is used in [5-7]. Query resolution cost is still $O(N/A)$, if the smallest r_q^i is relatively large.

Another alternative is to maintain a separate overlay ring for each attribute type [7]. Similar to the partitioned-ring, depending on how the resource attributes are advertised, multiple-rings can also utilize either sub-queries or SADQ to resolve range queries. However, a separate overlay routing table needs to be maintained for each ring. Alternatively, it is possible to map/overlap all the attribute values to the same address space using a separate uniform LPH function for each attribute type [5]. Both multi-ring and overlapped-ring architectures have the same cost as they have the same address space and number of nodes. However, their cost of resolving a range query is $O(N)$, as the range of attribute values is mapped to all the nodes in the ring. For this reason, resource discovery cost of multi-ring and overlapped-ring-based designs are higher compared to the partitioned-ring.

d-dimensional torus (d-torus) [8] has the second largest overall cost. Each dimension of the torus represents an attribute type. Resources are mapped to the torus according to their attribute values. A multi-attribute range query q is represented as a hyper-box on the torus. This approach is used in [8] where dimensions of the torus reflect only the static attributes. For routing, d-torus is logically partitioned into hierarchy of levels and each level is further partitioned along each dimension d. A node keeps a set of pointers to all the other nodes with identical attribute values and to a node in each partition at level l and dimension d. In contrast to prior architectures, resources are not explicitly advertised. Instead, a gossip scheme is used to identify a random node in each (l , d) pair. These pointers are used to route q to the desired region on the torus using depth-first search. This routing scheme is inefficient when queries are less specific, which significantly increases the volume of the hyper-box to query. This scheme is not considered for rest of the discussion, as it cannot route queries with only the dynamic attributes, which are the most popular types of attributes defined in real-world multi-attribute queries.

Unstructured P2P topologies are attractive as nodes self index resources (or distribute across many nodes) while providing resilience and load balancing. These systems typically use agent-based random walks to disseminate resource specifications and/or resolve queries. Agent lifetime is controlled using a Time To Live (*TTL*) value to balance the cost of advertising/querying and query success rate. Though advertising resources to other nodes speeds up the query resolution, state of the selected resources may be stale. Therefore, we consider only the query agents. A random walk is not guaranteed to find a resource; hence, only 73% of the queries were resolved.

Next, we analyze ring-based architectures in detail, as they are applicable in many large-scale applications due to scalability and some guarantees on performance. Fig. 2 shows the per node advertising cost of ring-based architectures under varying number of attributes. Advertising cost increases as dynamic attributes are introduced (first 12 attributes are static). Resources need to be re-advertised whenever their attribute values change and previous indexes need to be removed (to maintain consistency), if old and new attribute values do not map to the same overlay node. It is typically assumed that distributed hash table entries will expire after a predefined timeout. However, PlanetLab data suggest that it is nontrivial to determine an appropriate timeout given the diversity of attributes and their rate of change. Hence, we also considered the cost of removing old indexes as part of the advertising cost. Both advertise and remove messages can be delivered within $O(\log N)$ as they are sent to specific nodes. SADQ requires all the attributes of a resource to be advertised to each ring/partition corresponding to the attribute set of a resource. Therefore, resources need to be re-advertised to all the rings/partitions even when a single attribute is changed. Fig. 2 confirms this behavior where advertising cost of architectures that utilize SADQ is significantly higher and increases somewhat linearly with the number of attributes. We introduced dynamic attributes according to their popularity where attribute 13 is the most popular, 14 is the second most popular, and so on. Attributes 13-15 that corres-

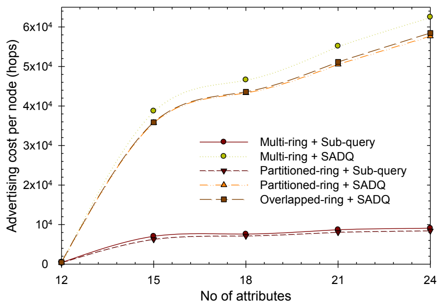

Figure 2. Advertising cost of ring-based architectures ($N = 527$).

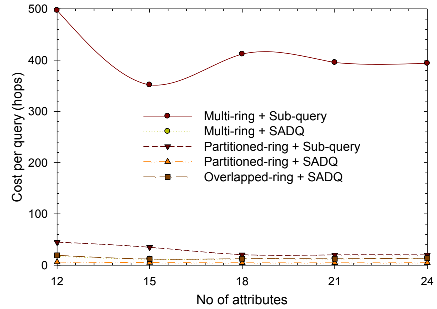

Figure 3. Query cost of ring-based architectures ($N = 527$).

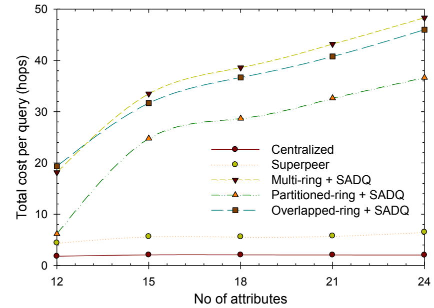

Figure 4. Total cost vs. number of attributes ($N = 527$).

ponds to response time of a node, its CPU load and free CPU utilization are updated frequently (minor variations are ignored by applying a fixed threshold [11]). This is the reason for the significant increase in advertising cost between attributes 12 and 15. Rate of update was relatively lower of rest of the dynamic attributes.

Fig. 3 illustrates the cost per query. Architectures based on SADQ have much lower query cost as they use only the most selective attribute. Moreover, queries can be terminated as soon as the desired number of resources is found. However, overall cost (advertising and querying) of SADQ-based architectures will be acceptable only if queries are more frequent than advertisements. Alternatively, sub-queries need to span multiple rings/partitions and have to search the entire range of attribute values specified in each sub-query, and hence have a higher cost. Queries that specified attributes 13-14 (response time and CPU load) tend to be more specific (i.e., small r_q^i), and therefore can be resolved by forwarding to a lesser number of nodes. Furthermore, these two most popular attributes appeared in many queries consequently reducing the overall query cost. Query distribution and range of attribute values get balanced as rest of the dynamic attributes are introduced. Consequently, query cost tends to stabilize. Though new attributes were introduced, SWORD queries specified only 1-7 attributes (one or two attributes were specified 80% of the time [11]). This explains why the query cost seems to be independent of the number of attributes in the system (Fig. 3).

Best design choice from each architecture is compared in Fig. 4. Unstructured-P2P-based architecture, which has the highest cost, is not shown to simplify the diagram. Increase in advertising cost of centralized and superpeer architectures due to dynamic attributes is insignificant, as their query cost is independent of the number of attributes and range of attribute values. Therefore, both architectures are competitive particularly when applied to small or moderate-scale P2P systems with many dynamic attributes and less specific queries. Next, in Table 2, we compare the three best ring-based designs with varying number of nodes N. Both the average cost per query and maximum query cost tend to increase linearly with N confirming Eq. 1. Therefore, cost of resource discovery is bounded by $O(N)$. Partitioned-ring-based approach has the lowest cost as its query cost is bounded by $O(N/A)$. These findings indicate that structured-P2P-based solutions effectively have a higher cost, compared to the well-known $O(\log N)$ bound, when range-queries are less specific.

TABLE II – QUERY COST UNDER VARYING NUMBER OF NODES ($A = 24$).

N	**Multi-ring + SADQ**			**Partitioned-ring + SADQ**			**Overlapped-ring + SADQ**		
	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max
250	0	9.2	239.1	0	3.7	19.4	0	9.1	238.4
527	0	13.7	509.0	0	4.6	27.6	0	13.5	506.0
750	0	16.2	719.1	0	4.9	36.6	0	16.5	719.9
1000	0	19.8	975.5	0	5.3	45.3	0	20.4	963.8

Table 3 presents the query cost, and per node query load and index size. For comparison, according to [5], we also generated range queries based on the uniform distribution of attributes and attribute values where each attribute in a query specifies 10% of the possible range of attribute values (i.e., $r_q^i = 0.1 r_{max}^i$). Resource attributes were not changed. Performance of resource discovery architectures is different when queries are formulated by selecting attributes and ranges of attribute values uniformly at random (irrespective of the actual resources). Not all random queries match resources in the system. Therefore, both the unstructured and superpeer architectures have to keep forwarding the queries until *TTL* expires. Consequently, overall cost of resource discovery increases. Furthermore, uniform queries cannot significantly benefit from SADQ, as the minimum range of any attribute value is 10% of the possible range (Eq. 1). Whereas in real-queries, at least a few attributes tend to be very specific (e.g., response time, CPU load, and number of CPU cores). Therefore, real-world queries can be resolved more efficiently using SADQ. Alternatively, given that range of most of the attribute values in real-world queries tend to be large, aggregated query cost significantly increases when multiple sub-queries are involved. Hence, sub-query-based solutions are less efficient while resolving real queries.

Centralized solution has to index all the resources and answer all the queries issued within the system (950,000 queries were issued during the simulation period). Unstructured and superpeer architectures have smaller index size as either nodes index themselves or superpeers index only a subset of the resources in the system. Furthermore, their query load is relatively balanced as random walks forward queries to many nodes in the system. Uniform queries are issued to multiple overlay rings/partitions with the same probability, and hence distribute the query load among many nodes. Alternatively, due to the skewed distribution of queries and attribute values in real-world systems, few rings/partitions and a subset of nodes in those rings/partitions are used to answer majority of the queries. This is the reason that number of SWORD queries answered by a node is 3.3 to 11.7 times higher than when answering uniform queries. Nevertheless, query load on a node that supports SADQ is relatively lower, as queries are resolved us-

TABLE III – QUERY COST, QUERY LOAD, AND INDEX SIZE ($N = 527$, $A = 24$).

Architecture	Total Cost per Query		Query Load				Index Size	
			Min		Max		Min	Max
	SWORD	Uniform	SWORD	Uniform	SWORD	Uniform		
Centralized	2.03	2.03	950,000	950,000	950,000	950,000	527	527
Unstructured	69.5	94.8	4,859	1,272	268,497	37,824	1	1
Superpeer	6.5	9.5	81,021	22,390	289,626	87,209	17	36
Multi-ring + SADQ	48.3	69.0	0	0	178,492	22,943	0	527
Multi-ring + Sub-queries	398.8	120.8	0	0	624,837	57,518	0	230
Partitioned-ring + SADQ	36.6	37.0	0	0	185,972	15,840	0	527
Partitioned-ring + Sub-queries	24.7	16.4	0	0	432,859	46,946	0	527
Overlapped-ring + SADQ	46.0	67.2	0	0	391,738	57,524	0	527

ing the most selective attribute. Moreover, those queries also terminate as soon as the desired number of resources is found. Hence, each node has to handle a relatively smaller number of queries. Fig. 2-4 confirm that partitioned-ring with SADQ outperforms all other design choices. However, still one of the nodes in the partitioned-ring had to answer ~20% of all the queries, which may not be acceptable when query rate is high.

Because of the correlation and skewed distribution of attributes and attribute values, resources are not uniformly spread across the possible range of attribute values and nodes. Instead, resources are indexed in only a small subset of the ring nodes while a large fraction of nodes does not index any resources at all. Multiple indexing used with SADQ and overlapped-ring also forces nodes to index many resources corresponding to different attributes. Furthermore, some of the attributes have only a few valid attribute values, e.g., CPU architecture and number of CPU cores. Such attribute values are indexed only at a few nodes and each of those nodes index large fractions of resources with the same attribute value. Any additional nodes in the ring are not useful, as they do not help to either distribute the query load or reduce the index size. Therefore, it is important to take into account the valid number of attribute values while adding nodes to a ring. Furthermore, as prefix bits were used to indicate the attribute type in partitioned-ring architecture, nodes that were mapped to less popular or unused partitions were never utilized. This issue can be overcome by partitioning the address space to support only the desired number of attributes. However, this limits the expandability of the system where addition of new attributes requires re-indexing of all the existing resources. Therefore, partitioned-ring-based SADQ suffers from significant load balancing issues, though it has a lower resource discovery cost. In general, load balancing is a critical issue in all the designs when popularity of resources/queries is skewed or queries tend to be less specific. As the existing solutions are applicable under very specific scenarios, novel resource-discovery solutions are needed to overcome the performance and QoS issues posed by real workloads. Hybrid approaches that combine the desirable features of centralized, superpeer, and ring-based architectures, while taking into account the resource and query characteristic, number of distinct attribute values, and cost of updating dynamic attributes, have the potential to provide better solutions.

V. CONCLUSIONS AND FUTURE WORK

Though real-world queries simplify query resolution in unstructured, superpeer, and SADQ-based structured P2P solutions, they introduced significant overhead in sub-query-based solutions. Query resolution cost of structured P2P systems is effectively $O(N)$ as the queries are less specific. Furthermore, all the solutions are prone to load balancing issues due to the skewed distribution of resources/queries. These findings indicate the need for more efficient and robust resource discovery solutions and the importance of accounting for the characteristics of real-world resources/queries while designing and analyzing such solutions. We are currently working on a resource aggregation solution that can overcome these limitations.

REFERENCES

[1] D. McLaughlin et al., “Short-wavelength technology and the potential for distributed networks of small radar systems,” *Bull. Amer. Meteor. Soc.*, vol. 90, Dec. 2009, pp. 1797-1817.

[2] G. Briscoe and A. Marinos, “Digital ecosystems in the clouds: towards community cloud computing,” In Proc. *3rd Int. Conf. on Digital Ecosystems and Technologies*, June 2009, pp. 103-108.

[3] A. Andrzejak, D. Kondo, and D. P. Anderson, “Exploiting non-dedicated resources for cloud computing,” In Proc. *12th Network Operations and Management Symposium*, Apr. 2010.

[4] H. Shen, A. Apon, and C. Xu, “LORM: Supporting low-overhead P2P-based range-query and multi-attribute resource management in grids,” In Proc. *13th Int. Conf. on Parallel and Distributed Systems*, Dec. 2007.

[5] M. Cai et al., “MAAN: A multi-attribute addressable network for grid information services,” *J. Grid Comput.*, Jan. 2004.

[6] J. Albrecht et al., “Design and implementation tradeoffs for wide-area resource discovery,” *ACM Trans. Internet Technology*, 8(4), Sep. 2008.

[7] A. R. Bharambe, M. Agrawal, and S. Seshan, “Mercury: Supporting scalable multi-attribute range queries,” *SIGCOMM*, Aug./Sep. 2004.

[8] P. Costa et al., “Autonomous resource selection for decentralized utility computing,” *29th Conf. on Distributed Computing Systems*, June 2009.

[9] H. Shen and C. Xu, “Performance analysis of DHT algorithms for range-query and multi-attribute resource discovery in grids,” In Proc. *Int. Conf. on Parallel Processing*, Sep. 2009. pp. 246-253.

[10] E. Heien, D. Kondo, and D. P. Anderson. “Correlated resource models of Internet end hosts,” In Proc. *31st Int. Conf. on Distributed Computing Systems (ICDCS ‘11)*, June 2011.

[11] H. M. N. D. Bandara and A. P. Jayasumana, “Characteristics of multi-attribute resources/queries and implications on P2P resource discovery,” In Proc. *9th ACS/IEEE Int. Conf. On Computer Systems And Applications (AICCSA ‘11*, Dec. 2011.

[12] H. M. N. D. Bandara and A. P. Jayasumana, “On characteristics and modeling of P2P resources with correlated static and dynamic attributes,” In Proc. *IEEE GLOBECOM ‘11*, Dec. 2011.

[13] PlanetLab, Available: http://www.planet-lab.org

[14] K. Park and V. S. Pai, “CoMon: a mostly-scalable monitoring system for PlanetLab,” *ACM SIGOPS Operating Systems Review*, 40(1), Jan 2006.

[15] M. George, “B-A scale-free network generation and visualization,” Available: www.mathworks.com/matlabcentral/fileexchange/11947

[16] I. Stoica et al., “Chord: a scalable peer-to-peer lookup service for internet applications,” In Proc. *ACM SIGCOMM ‘01*, 2001.