World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:7, 2007

A Framework for Scalable Autonomous P2P Resource Discovery for the Grid Implementation

Hesham A. Ali, Mofreh M. Salem, and Ahmed A. Hamza

***Abstract*—**Recently, there have been considerable efforts towards the convergence between P2P and Grid computing in order to reach a solution that takes the best of both worlds by exploiting the advantages that each offers. Augmenting the peer-to-peer model to the services of the Grid promises to eliminate bottlenecks and ensure greater scalability, availability, and fault-tolerance. The Grid Information Service (GIS) directly influences quality of service for grid platforms. Most of the proposed solutions for decentralizing the GIS are based on completely flat overlays. The main contributions for this paper are: the investigation of a novel resource discovery framework for Grid implementations based on a hierarchy of structured peer-to-peer overlay networks, and introducing a discovery algorithm utilizing the proposed framework. Validation of the framework's performance is done via simulation. Experimental results show that the proposed organization has the advantage of being scalable while providing fault-isolation, effective bandwidth utilization, and hierarchical access control. In addition, it will lead to a reliable, guaranteed sub-linear search which returns results within a bounded interval of time and with a smaller amount of generated traffic within each domain.

***Keywords*—**Grid Computing, Grid Information Service, P2P, Resource Discovery.

I. Introduction

THE Grid is a type of parallel and distributed system that "enables coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" [1]. Resources comprising the Grid are heterogeneous (e.g. computers, data sources, applications, etc.) and the main goal of the Grid is to efficiently enable users to access and integrate these resources through a single interface. The main subsystem responsible for achieving this is the Grid Resource Management System (GRMS).

In order to facilitate resource sharing, it would first be necessary that the GRMS finds the location of desired resources in an efficient and timely fashion. This problem is known as resource discovery. Resource discovery, a basic service of any GRMS, is concerned with the problem of matching a query for resources to a set that meets the query's conditions. These conditions (constraints) describe required characteristics (or attributes) of the resources (e.g., having certain CPU architecture).

Current approaches to resource discovery on Grids are based on centralized or hierarchical client/server models. In such models, there exist one or more index servers which act as a centralized resource discovery directory. These servers maintain index information about available resources, and accept requests from resource consumers. As Grid systems increase in scale, they begin to require solutions to issues of self-configuration, fault-tolerance, and scalability, for which peer-to-peer (P2P) research has much to offer.

It is expected that there will be strong convergence between peer-to-peer and Grid computing [2]. The techniques used in each of these two different types of systems will result in mutual benefits. And the outcome of such convergence would be a new class of technologies which would address scalability, self-adaptation, and failure recovery, while, at the same time, providing a persistent and standardized infrastructure for interoperability.

In this paper, we present SAPRDG (Scalable Autonomous Resource Discovery for the Grid), a framework for scalable resource discovery on the Grid based on structured P2P overlay networks. The framework attempts to solve many of the difficulties encountered in P2P resource discovery while, at the same time, remaining scalable to a large number of distributed nodes. In particular, the contributions of the proposed framework are: (i) the ability to perform multi-attribute exact match and range queries to discover resources matching a given criteria, (ii) the ability to scale to a large number of nodes and organizations scattered over geographically distributed locations, and (iii) maintaining administrative control over the local resources and their states.

The remainder of this paper is organized as follows. Section 2 gives an overview on the most recent efforts in utilizing structured P2P overlay networks for decentralizing the Grid Information Service (GIS). A description of the shortcomings of the current approaches to decentralized resource discovery and the suggested approach to overcome them is given in Section 3. Section 4 formally introduces the proposed resource discovery framework and provides a detailed

Mofreh M. Salem is with the Computers and Systems Engineering Department, Faculty of Engineering, Mansoura University, Mansoura, Egypt.

Hesham A. Ali is with the Computers and Systems Engineering Department, Faculty of Engineering, Mansoura University, Mansoura, Egypt (e-mail: h_arafat_ali@mans.edu.eg).

Ahmed A. Hamza is with the Computers and Systems Engineering Department, Faculty of Engineering, Mansoura University, Mansoura, Egypt (e-mail: ahmed.hmz@gmail.com).

description of its architectural layers and interacting modules. The resource discovery algorithm utilizing the proposed framework is discussed in Section 5. A discussion of experimental results based on the simulation of a preliminary implementation of the framework is introduced in Section 6. Finally, Section 7 summarizes our work and points out some future directions for research.

II. Related Work

The Multi-Attribute Addressable Network (MAAN) [3] is an extension of the Chord protocol [4] to handle multi-attribute range queries. To preserve the locality of keys, MAAN uses a uniform locality-preserving hash function to map the values of the resources to Chord's *m*-bit key space. Two approaches are proposed to search candidate resources for multi-attribute range queries: *iterative* and *single attribute dominated* query resolution. The first approach iteratively searches all candidate resources for each sub-query on one attribute dimension and intersects these search results at query originator. The second approach reduces the number of routing hops needed to resolve the query by taking advantage of the fact that search results of a query must satisfy all the sub-queries on each attribute dimension and is the intersection set of all resources that satisfies each individual sub-query.

Mercury [5] is a Grid system that uses Symphony [6], a one-dimensional DHT as its underlying architecture. The *Mercury* network organizes the nodes into *attribute hubs*. Each hub is responsible for the management of a single attribute of the indexed items. Range queries in a hub are resolved by searching the node responsible of storing the index of the smallest value in the range. Then the query is forwarded to the other peers until the range of the largest value is covered. All the attribute values of a resource are registered in all the hubs. Multi-attribute queries are resolved by using only the attribute of the sub-query with the smallest range (i.e. the most selective attribute).

SWORD [7] is an information service that supports multi-attribute range queries to locate a set of machines matching user-specified constraints on both static and dynamic node characteristics. There are two kinds of nodes in SWORD: *reporting nodes*, which periodically send measurement reports on resources, and *DHT server nodes*, which receive resource information and handle queries from users. Server nodes are organized using Bamboo [8], which is a structured P2P system similar to Pastry [9]. SWORD builds its own soft-state distributed data repository on top of the key-based routing functionality of the underlying DHT and assigns attributes to different sub-regions of the common overlay by the choice of *explicit* hash functions. These functions segment the key space by encoding the attribute in the higher-order bits.

NodeWiz [10] is a self-organizing, distributed aggregate directory that specializes in multi-attribute range queries. NodeWiz is not DHT-based, it builds a distributed decision tree through routing tables created on the participating nodes. The number of tracked attributes is predetermined and, similar to SWORD, a soft-state approach is used for storing resource information for the dynamic attributes.

In [11], the authors present a Grid resource indexing algorithm based on the Pastry [9] DHT. A Resource ID is generated from the resource state by combining a 32-bit encoding for the dynamic attributes values and a 128-bit hash of a 32-bit encoding for the static attributes values to obtain a 160-bit key. Consequently, the dynamic part of the Resource ID is represented by overlapping arcs on the overlay. Three different heuristics for resolving range queries are proposed: *single-shot searching*, *recursive searching*, and *parallel searching*.

Table I summarizes the main features of the Grid resource discovery systems based on structured P2P architectures described above in terms of architecture, protocol, resource registration, query resolution, and load balancing.

III. Problem Statement

Surveying the current literature on Grid systems based on P2P resource discovery techniques [12], [13] it is clear that utilizing such techniques in general, and *structured* P2P techniques in particular, for decentralizing the Grid Information Service (GIS) faces a number of challenges. The three main challenges are:

1. *Nature of the query* (number of dimensions and constraint type): Because they use hashing to distribute keys uniformly in order to achieve load balance, DHTs only directly support exact match queries. Extending DHTs to support range queries is a complex problem by itself and for supporting multidimensional range queries the problem is complicated even more.
2. *Type of the resource attribute:* In contrast to static attributes, dynamic attributes change very rapidly within short periods of time. Resource state information inserted into the GIS containing such attributes must be periodically refreshed to reflect the current state of the resource. This, however, may lead to high network traffic since a large number of messages get routed between the overlay's nodes.
3. *Preserving autonomy:* Utilizing completely flat P2P overlays to decentralize the GIS does not take into account the autonomy of the participating organizations and their, possibly, conflicting interests. Thus, *content locality* and *path locality* cannot be ensured, thereby raising security and accountability concerns.

These challenges and resource discovery mechanisms attempting to address them directly affect two main performance factors: response time and amount of routed traffic. For example, locality greatly affects the response time. If the nodes containing the information being queried are in close proximity of each other, lookup messages will travel shorter distances resulting in less time to resolve queries. Also, solutions that address the range queries problem usually

TABLE I
GRID RESOURCE DISCOVERY SYSTEMS BASED ON STRUCTURED P2P ARCHITECTURES

System	Architecture	Routing Protocol	Resource Registration	Resource Query Resolution	Load Balancing
AdeepGrid (2005) [11]	Resources are represented as potentially overlapping arcs on the DHT ring	Pastry	Hashing a customized encoding of attributes values. Normal DHT insertion.	Calculate resource ID based on protocol's encoding and hash functions	Relies on underlying DHT
NodeWiz (2005) [10]	Divide attribute space using a structure represented by a kd-tree	Distributed decisio tree-like	Utilize routing tables to locate responsible node.	Utilize routing tables to locate responsible node.	Divide attribute space during join based on workload according to propagated load information. (Top-K algorithm)
MAAN (2004) [3]	One DHT per attribute	Chord	Register each attribute in appropriate DHT	Query each attribute separately then intersect results at querying node	Uniform, locality preserving hash (LPH) function. Value distribution known in advance
Mercury (2004) [5]	One DHT per attribute	Symphony	All attributes registered in every DHT	Lookup on DHT of attribute with smallest range	Periodic network sampling to find load imbalances (leave-rejoin protocol)
SWORD (2004) [7]	Assign attributes to different sub-regions of a common DHT	Bamboo	Route report containing values to sub-regions of key attributes	Send query to sub-region of most selective attribute, or one that is randomly chosen	Leave-rejoin protocol. Customized hash functions

resort to some sort of indexing technique which, in a distributed environment would usually require the exchange of maintenance messages to keep the index in a correct state. A good P2P resource discovery solution would attempt to address these challenges while striking a balance between these two performance factors.

A. Plan of Solution

In order to reduce the response time, recent approaches to resolve resource discovery queries in a decentralized fashion attempt to cluster the data being inserted into a flat overlay network so that points close together are mapped to nodes that are close together in the Id space. In contrast, we propose a framework that clusters the overlay nodes (not the data) depending on the autonomous system (AS) they are in. By doing so, latencies between the nodes lying within the same AS and, consequently, the response time for a query issued within that domain are reduced significantly.

Queries are resolved at the target cluster. And for global queries, those spanning all the domains participating in the system, we parallelize the search process. In addition, due the difference in nature between static attributes and dynamic attributes, different techniques are used to resolve queries on dynamic attributes that take into account the volatile nature of their measured values.

IV. PROPOSED FRAMEWORK

Before describing the framework, some terminology is in order. Each organization participating in a Grid system has its own set of routers to which the nodes in the system are attached. These routers form what is referred to in the field of computer networks as an autonomous system (AS). Therefore, within the context of this paper, the terms *organization* and *domain* will be used interchangeably.

A fixed system-wide attribute schema of resources is set in advance. The schema defines the following parameters for each attribute: *value type*, *name*, *minimum value*, *maximum value*, *interval width*, and *attribute type*. An example schema is shown in Table II. The interval width determines how the full value range of the attribute is discretized when performing resource registration or query resolution. The attribute's nature determines whether the attribute is static or dynamic. This will be of interest when we discuss the discovery algorithm used in the framework.

TABLE II
SIMPLE EXAMPLE OF AN ATTRIBUTE SCHEMA

Type	Name	Min. Value	Max. Value	Interval Width	Nature
Integer	Memory Total	0	1024	1	Static
Integer	Disk Total	0	1024	1	Static
Integer	CPU Idle	0	100	10	Dynamic
Integer	Free Memory	0	100	10	Dynamic

A. System Organization

In the SAPRDG framework, the nodes of the system are organized in two levels of structured P2P overlays, as shown in Figure-1. A structured P2P overlay network (e.g. Chord [4], Pastry [9], Kademlia [14]) is created between various nodes within each organization to distribute the load and eliminate single points of failure.

Even though the nodes of any given organization are usually within a proximate range of each other and are connected via a high speed connection with large bandwidth. Constructing a structured P2P overlay between these nodes will allow the organization to build an incrementally scalable distributed discovery service using cheap commodity components [15].

From the nodes participating in the local P2P overlay of an organization, a small number of well connected and relatively

powerful nodes are chosen to perform inter-domain routing. These nodes then become gateways for any local node wishing to perform a search for resources existing in other domains. The gateway nodes (GNs) of each organization (or domain) participate in another structured P2P overlay at a higher level. This overlay would then enable routing messages between the different organizations.

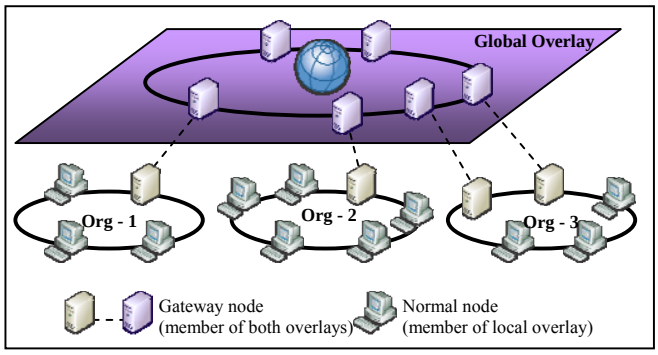


Fig. 1 System organization for the proposed framework

This organization of the system's nodes is inspired by and similar to that discussed in [16]. However, unlike [16], it will not be required that all connectable nodes must participate in the global overlay. The small subset mentioned earlier would suffice. But the robustness of the system and its degree of fault-tolerance would increase by increasing the size of the subset, because this will decrease the probability of domain isolation in case all gateway nodes fail.

To enable inter-domain routing, a globally unique domainId is assigned for each organizational overlay. All members of a given overlay know its domainId, and the global overlay has a well-known domainId (e.g. all zeros). Consequently, each node in the system can be identified by an Id obtained by concatenating the domainId of the overlay in which the node participates and the Id of the node within the overlay. It should be noted that a globally accessible naming service (e.g. a service similar to DNS or the Contact Service mentioned in [17]) can be used to map more readable and meaningful domain names to domainIds. Such a service could also be implemented using a DHT to prevent it from being a bottleneck to the system.

To facilitate global search, each newly connected GN in an organization broadcasts the domain's Id to all other members in the global overlay. This way, all the GNs connected to the global overlay will be aware of all reachable domains in the system. A node wishing to perform a global search can do so by either first contacting a GN of its domain to obtain a list of all reachable domains and then issue a domain-specific search request for each one, or delegate this whole process to a GN within its domain.

B. Architectural Layers and Modules

The framework exploits a layered architecture in order to leverage existing technologies and techniques. This has the advantage of distributing responsibilities among different layers and provides the flexibility of changing implementations while maintaining the same interface between the various layers. Fig. 2 depicts the different layers that exist within each node of the system.

The information service (IS) layer contains the main modules of the framework and acts as the coordinator (or orchestrator) which guides the resource discovery process. Thus, it is in this layer where most of this proposal's implementation resides. The modules within this layer interact with each other and with the underlying layers to successfully process a given query for resources and return appropriate matches. The two main layers that the IS layer directly interacts with are the distributed indexing layer and the publish/subscribe layer. The former builds a distributed indexing data structure on top of an underlying DHT to efficiently resolve range queries on static attributes. This is necessary to overcome the breakage of data locality caused by the randomizing cryptographic hash functions utilized for load balancing in most DHTs. The publish/subscribe layer will be used to resolve queries on dynamic attributes as discussed in the next section.

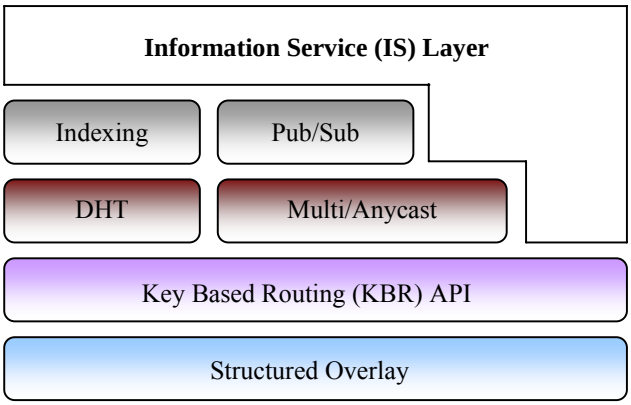


Fig. 2 Layered Architecture

Within the IS layer of each participating node, there are four modules: the parsing module, the routing module, the query processing module, and the advertisement module. The first three modules cooperate to answer a given query for suitable resources and the fourth is responsible for advertising the current state of the node.

Fig. 3 extends Fig. 2 to demonstrate the modules of the information service layer and the interaction between these modules and the underlying layers. Following is a brief description of the responsibilities of each module:

Advertisement Module periodically monitors the state of the resources available on the node by measuring the values of the attributes specified in the schema and performs resource registration via the underlying publish/subscribe and distributed indexing infrastructures.

Parsing Module receives incoming queries and parses them according to defined schema to extract the search criteria and search level.

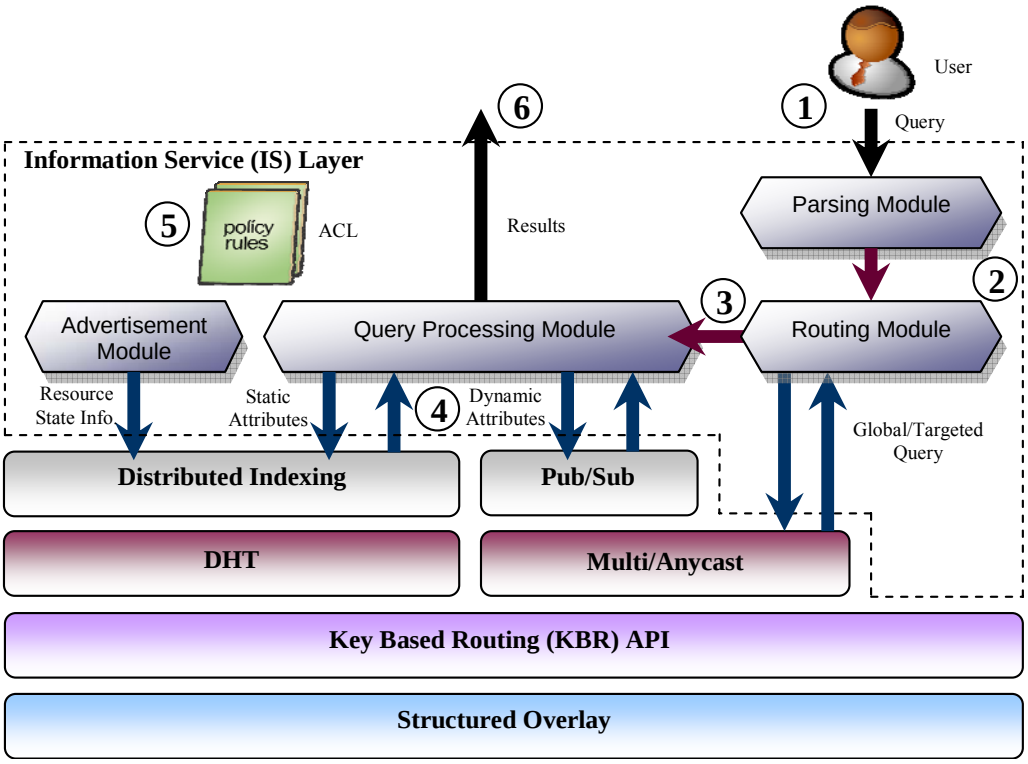


Fig. 3 Information service layer modules and their interaction model

Query Processing Module responsible for obtaining suitable matching resources, enforcing the domain's policy rules, and answering the query at hand by interacting with the distributed indexing and publish/subscribe layers. On GNs, this module is also responsible for checking the results against the organization's defined policies and performing post-query filtering.

Routing Module responsible for routing search requests to the appropriate overlay(s) at which they will be processed by member nodes.

In addition to the four modules that run on each node, gateway nodes also maintain an Access Control List (ACL) that defines the capability limits allowed to be exported to certain external domains. This is achieved via a mapping between the organization's domainId and a number of corresponding restrictions on the allowable results.

V. Discovery Algorithm

In this section, we discuss a resource discovery algorithm for SAPRDG. Resource discovery is initiated upon reception of a resource lookup request via the information service layer of any node within the system. This request consists of two parts: (i) a query containing a number of predicates, and (ii) the level at which the query will be executed. The request's structure and organization can be formatted in XML for example. It is first handed to the parsing module to extract the search criteria and convert it to a form usable by the system.

After extracting the necessary information from the request, the parsing module then passes this information to the routing module to determine where it should be routed in order to be processed. This decision is based on the search level indicated in the lookup request. At this point, the mechanism can proceed in one of three possible directions, according to the possible alternatives of the search level, as shown in Fig. 4. If the search level is the *local* level, no external routing is necessary and the current node, the one receiving the request, will be responsible for processing the query. If the search level is the *targeted* (domain-specific) level, the query needs to be routed to a member GN of the target domain's overlay via the global overlay. The last alternative is the *global* level, in which case the query gets broadcasted to all the domains of the system (also using the global overlay).

Query processing is done locally within each domain by the local query processing module of either any node within the overlay of the organization (if the query originated at that node) or one of the gateway nodes of the organization's overlay (if the query originated at a node in another organization). In other words, if the origin of the query is a foreign node, the task of processing it is delegated to the nearest encountered GN in the target overlay. A flow chart demonstrating how the query is resolved by the local query processing module is shown in Fig. 5.

The module first determines the types of the attributes present in the query's constraints then proceeds in two parallel paths: one for the static attributes and one for the dynamic ones. For static attributes, the distributed indexing layer is used after applying a dimensionality reduction mechanism

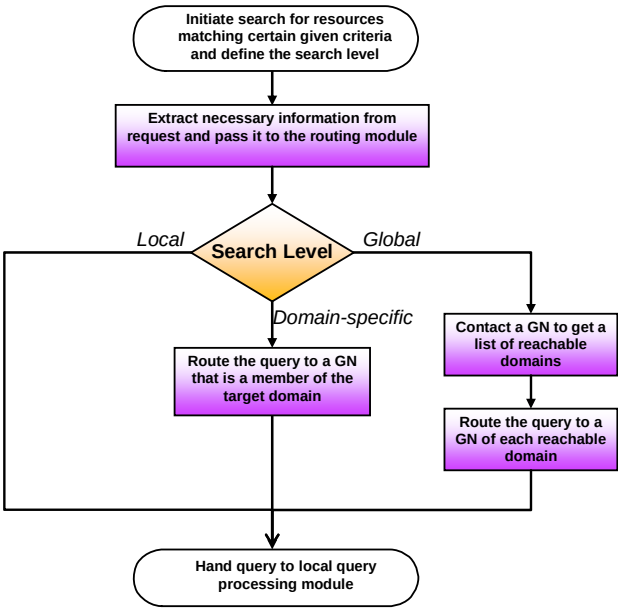


Fig. 4 Routing alternatives based on the search level

(e.g. Space-Filling Curves (SFCs) [18]) on the measured static attributes values. Taking a Prefix Hash Tree (PHT) [19] as an example for a distributed indexing layer, the minimums of the query are linearized and the binary string representation is obtained. Similarly, we obtain a binary string representation of the linearized maximums. The common prefix of both binary bit strings is then determined and the logical distributed data structure used for indexing is used to look up any matches. This is achieved by determining the longest common prefix of the linearized maximums and minimums, looking up the corresponding PHT node, and performing parallel traversal of its sub-tree [19].

Dynamic attributes are treated differently. Instead of using DHT lookup, we revert to a publish/subscribe mechanism, more specifically a content-based publish/subscribe mechanism [20]. When processing a query, the query processing module determines the dynamic attributes present in the query and expresses its interest in publications with contents matching the desired resource attribute values or value ranges. This is achieved by subscriptions with predicates over these attributes. These subscriptions are time-bound (i.e. the node remains subscribed for a predefined period of time t_{sub} after which the subscription is removed).

During the subscription period, publications by any resource having measurements matching the criteria defined in the query for the dynamic attributes will arrive at the node on which the query is being processed. This mechanism is more efficient than that defined in [21] and similar approaches in which dynamic attributes measurements are broadcasted to all nodes within the system. This is mainly because such approaches generate high network traffic and deliver unnecessary information to uninterested nodes.

After obtaining a list of results matching the desired static attributes values L_{static} and the results list matching the desired dynamic attribute values $L_{dynamic}$, the next step for the query processing module is to filter these results for resources present in both lists (i.e. those matching both types of attributes at the same time) and obtain a list of candidate resources $L_{candidates}$.

Depending on whether the query by a GN is on behalf of an external node or not, a final step is performed before returning the final matching results list $L_{matching}$ to the node issuing the query. This step is to check the GN's ACL for any restrictions placed on the results returned to the domain of the originating node. In the case of local level lookups, however, $L_{matching}$ will be $L_{candidates}$ and this step is not necessary.

From above, it is clear that the final response time is greater than the maximum response time of both the static attributes sub-query and the dynamic attributes sub-query. If the chosen t_{sub} is greater than the response time for the static attributes sub-query, the total response time will be greater than t_{sub}. Otherwise, if the response time for the static attributes sub-query is greater, then the total response time will be greater than that of the static attribute's sub-query.

VI. Implementation and Experimental Results

The SAPRDG framework was implemented in Java on top of FreePastry[1], an open-source implementation of the Pastry [9] P2P routing substrate. For the purpose of performance evaluation, a customized discrete-event simulator was built. We used Brite [22] topology generator to create a two-level top-down hierarchical topology of 20 ASs each containing 10 routers for a total of 200 router nodes. Fig. 6 illustrates the topology's graph as visualized by Otter [23], a network visualization tool. The nodes in the graph represent routers to which a number of hosts can be attached and the edges represent communication links between the various routers. Routers belonging to the same AS are represented with the same color.

The Dijkstra shortest-path algorithm [24] was then used to generate a latency matrix whose elements designate the delays between the various routers of the network. The simulator uses the generated router-level topology latency matrix and end-host nodes are then assigned to the different domains by attaching them randomly to one of the domains routers. The latencies between end-host nodes connected to the same router are assumed to be constant and equal to 2 milliseconds. For simplicity, and to enable simulation of large networks, the simulator does not model link bandwidth.

All experiments were run on a Dell PowerEdge 1800 server machine with dual 3.0 GHz Intel Xeon processors, 2 GB of RAM, and running CentOS 5 [25] with a 64-bit version on the Linux 2.6.18-8.1.8.el5 kernel and Java run time environment version 1.5.0_12. We used the schema given in Table-2 to evaluate the framework. The metrics used for evaluation are the number of routed messages per domain and the query response time. Each experiment was repeated 5 times and the average number of messages routed per domain and average

[1] FreePastry 2.1 alpha (http://freepastry.rice.edu/)

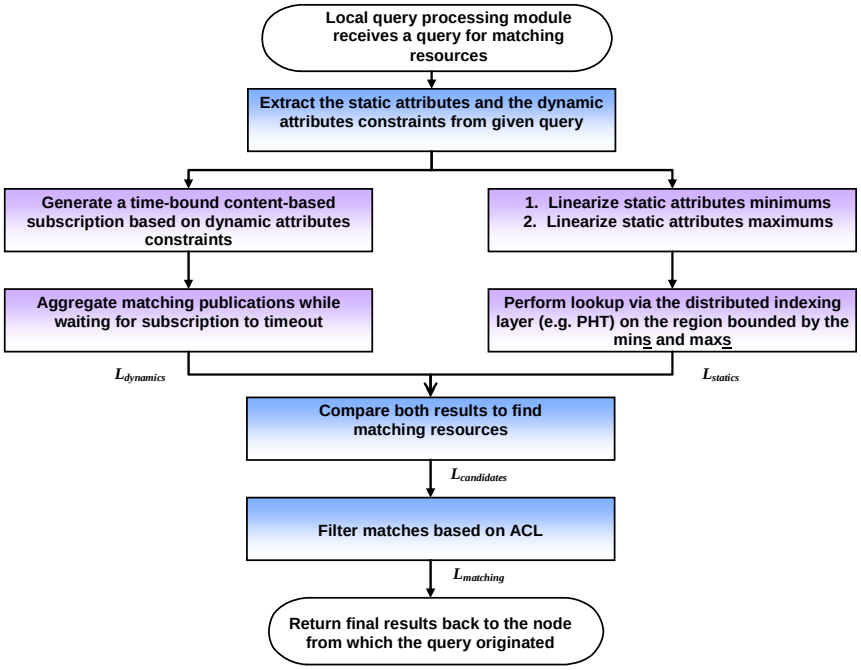


Fig. 5 Flow chart of the query resolution process performed by the query processing module

query response time were calculated.

First, the nodes are created according to the layout discussed in section IV. Each node is then assigned, with uniform probability, random values for the static attributes defined in the attribute schema and these values are inserted into the overlay via the distributed indexing data structure (in our case the PHT). After that, each node periodically publishes random values for their dynamic attributes with uniform distribution. We then pick 10 nodes at random from each domain and concurrently issue global queries from these nodes. To stress the system, the range constraints defined in the query for each attribute cover the attribute's full range. The number of routed messages in each domain is measured and the average number of routed messages per domain is calculated. In addition, we also calculate the average query response time.

The case studies used for evaluating the performance of SAPRDG are listed in Table III. In each case, the number of GNs for each domain was 3 nodes and the PHT block size was set to 10. For the DHT layer, no replication was present. The publication period for the dynamic attributes was chosen to be 30 seconds. These are, of course, system parameters that could be adjusted to give the best performance according to the system's needs in a real-world implementation.

Fig. 7 shows how the average number of routed messages per domain increases when the number of nodes per domain increases from 100 to 500 nodes at various numbers of participating organizations. It is clear that, given a certain number of participating organizations, the average number of messages routed per organization increases almost linearly with the increase of number of nodes per organization. The slope of this increment increases as the number of participating organizations gets larger. This can be traced back to the fact that increasing the number of organizations in the experiment would increase the number of nodes concurrently performing queries. However, due to the organization of the system, such traffic will not be of very big concern since it will only reside within the boundaries of the organization which will usually have high-speed LANs connecting its nodes.

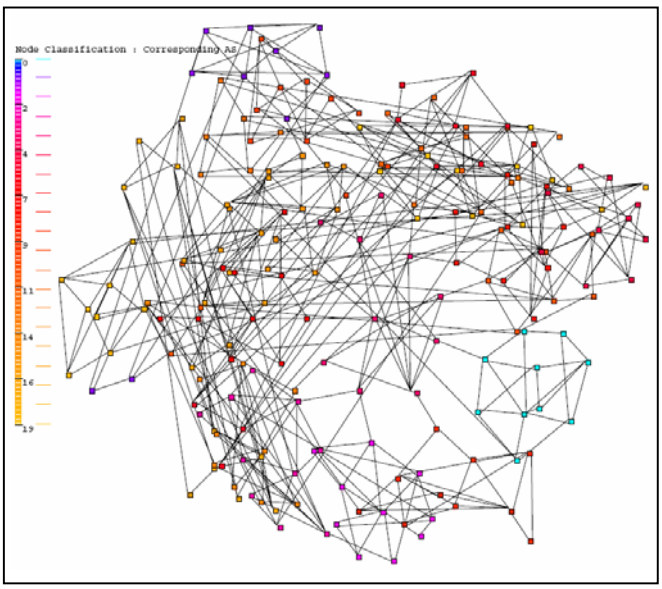


Fig. 6 Visualization of the generated router-level topology

TABLE III
CASE STUDIES USED FOR FRAMEWORK PERFORMANCE EVALUATION

Case	Number of Orgs.	Number of Nodes Per Organization		Total Number of Nodes		Total Number of Queries
		From	To	From	To	
C1	2	100	500	200	1000	20
C2	4	100	500	400	2000	40
C3	6	100	500	600	3000	60
C4	8	100	500	800	4000	80
C5	10	100	500	1000	5000	100

Fig. 8 illustrates the average query response times for the same set of experiments. It can be seen from the chart that the system exhibits a small increase in response time with respect to the number of nodes per organization (approximately 1 second for each 100 node increment).

In order to demonstrate the performance gain of the SAPRDG approach over the traditional flat overlay approach, we compared the response times and number of routed messages obtained for cases C1, C2, and C3 against local search within a single overlay using the same topology generated with Brite. In this sense, the nodes of the single overlay are distributed amongst the ASs without regard to any organizational boundaries.

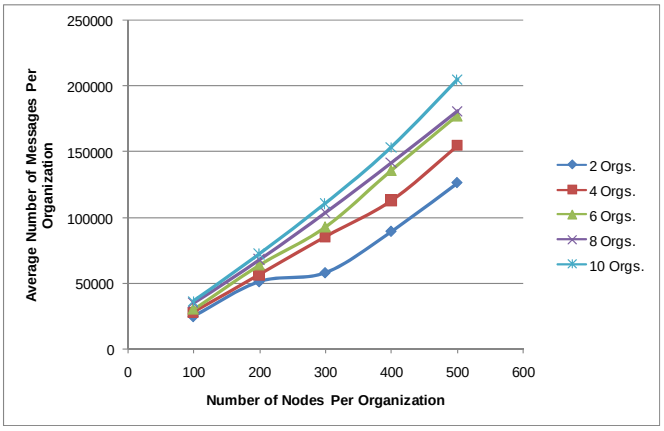


Fig. 7 Average number of routed messages vs. number of nodes per organization

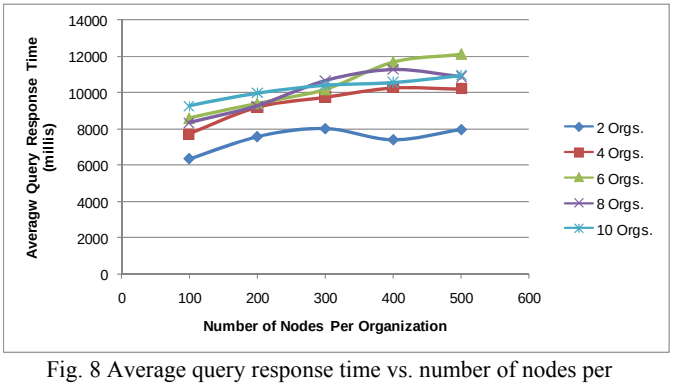


Fig. 8 Average query response time vs. number of nodes per organization

As can be seen in Fig. 9, the total number of routed messages in a system utilizing our proposed framework is significantly less than that of a system based on a single global overlay. This is because clustering the nodes will create local overlays with smaller number of nodes. Thus, the Id range defining the range of objects for which a node is responsible for increases. Having a smaller number of nodes per overlay also means that the number of forwarding nodes in the multicast tree that the publish/subscribe infrastructure will decrease. These consequences, in turn, would cause a fewer number of hops in order to resolve range queries.

It is also noted that, as shown in Fig. 10, the average query response time also decreases. Plotting the performance gain in response time for our framework over the traditional flat approach for the three studied cases using the formula:

$$g = \frac{m_1 - m_2}{m_1} \quad (1)$$

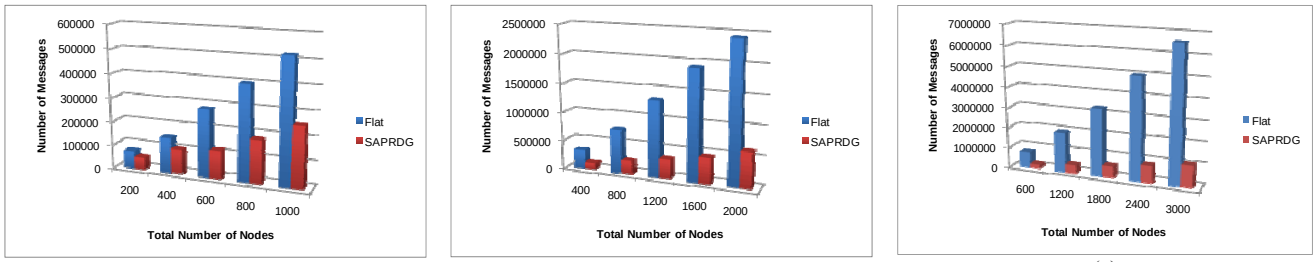
Where m_1 is the response time using a traditional flat approach to resource discovery and m_2 is the measurement using the proposed hierarchical approach, Fig. 11, it is apparent that the framework reduces the average response time for a given query by around 55%. This is largely due to the fact that we are now performing parallel search in all the local overlays in, roughly, the same time. This divide and conquer methodology is more efficient than the traditional approach in which the search is conducted over a large number of widely distributed nodes. In addition, having the nodes in close proximity form a local overlay reduces the distances the messages are required to travel and speeds up the search.

A. Impact of Number of Domains on Performance

To study the impact of the number of clusters (domains) on a system having a fixed number of nodes, we generated a system with 2400 nodes and initiated 24 concurrent global queries for the cases where the nodes (and query sources) were distributed evenly between two, four, six, and eight domains, respectively. As with previous experiments, we measured the average query response time and the total number of lookup messages routed within the system.

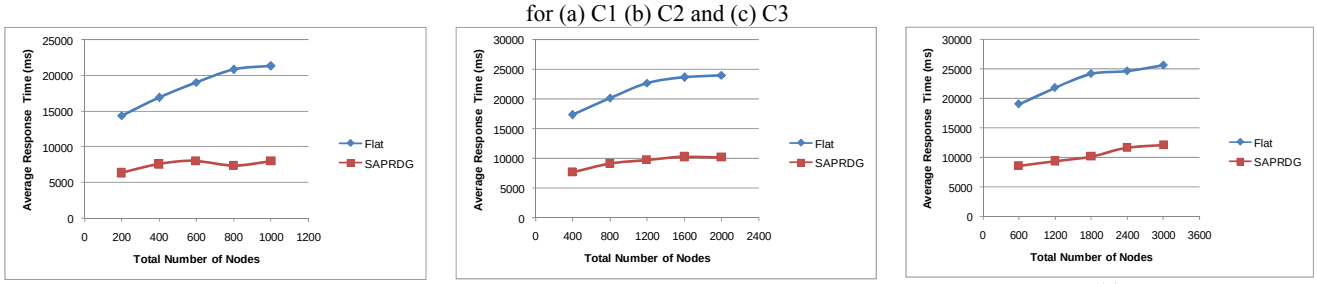
The results shown in Table IV illustrate the effect of clustering on the total number of lookup messages. The table also shows the performance gain in each metric for each of the four cases compared against a system using the flat approach with the same number of nodes and queries. As can be seen, the number of routed messages decreases when the number of domains upon which the nodes are distributed increases. The average query response time experiences similar results. The response time has dropped from 10.6 seconds in the case of two domains to less than 9 seconds for the case of eight domains.

It is expected that further performance improvement can be achieved by extending the framework to support a multi-level hierarchy organization in which the nodes within the domains are further clustered according to network proximity in a way similar to how multi-level hierarchical topologies are generated in Brite.



(a) (b) (c)

Fig. 9 Number of routed messages in proposed framework against number of routed messages in a single overlay system for (a) C1 (b) C2 and (c) C3



(a) (b) (c)

Fig. 10 Average query response time in proposed framework against average query response time in a single overlay system for (a) C1 (b) C2 and (c) C3

TABLE IV
EFFECT OF CLUSTERING GRANULARITY ON PERFORMANCE

Number of Local Overlays	Average Response Time (ms)	Perf.ormance Gain in Response Time	Number of Lookup Messages	Performance Gain in Lookup Traffic
2	10691	57%	633503	55.7%
4	10154	59.2%	482979	66.2%
6	9473	61.9%	426017	70.2%
8	8949	64%	383697	73.2%

VII. CONCLUSION AND FUTURE WORK

Peer-to-peer promises to eliminate most of the Grid's bottlenecks. Exploiting P2P techniques in current Grid services, such as the GIS, would achieve higher degrees of scalability. Structured P2P overlays guarantee bounded results and provide greater scalability than their unstructured counterparts which make them more suitable for a system as large as a global Grid. Both solutions, however, do not take locality and administrative control, a crucial consideration in Grids, into account.

In this paper we proposed SAPRDG, a framework for resource discovery in Grid environments based on a hierarchical structured peer-to-peer architecture. Simulated experimental results have shown that the framework is both scalable in terms of, both, response time and number of generated lookup traffic. The framework also preserves administrative control and autonomy because routing between various administrative domains is permitted according to the policies defined by the target domain. In addition, the framework takes the differences in nature between the dynamic and static attributes defining a resource's state into account and increases parallelism in the resource query resolution process to reduce the query response time and amount of generated traffic per domain. Although the discussed organization uses only two levels, which would normally be sufficient for building a scalable GIS, it could easily be extended to support more levels if necessary.

One of the limitations that the framework suffers from is that it does not support evolvable attribute schemas. The attribute schema has to be fixed and known in advance. This is planned to be addressed in the future. Future work also includes: an attempt to expose the framework's functionality as OGSA Grid services that can be used to augment resource discovery in Grid resource management systems (GRMSs), and studying the effect of churn on the system. Finally, in order to study the performance of the framework more accurately, we plan to perform experiments on more realistic environments. Examples of such environments include a network emulation testbed on a cluster of workstations, using ModelNet [26] for example, or a global research network like PlanetLab [27].

REFERENCES

[1] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International J. Supercomputer Applications*, 15(3), 2001.

[2] I. Foster and A. Iamnitchi, "On death, taxes, and the convergence of peer-to-peer and grid computing," *2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, 2003.

[3] M. Cai, M. Frank, J. Chen, and P. Szekely, "MAAN: A Mulit-Attribute Addressable Network for Grid Information Services," *Journal of Grid Computing*, 2(1), 2004, pp. 3-14.

[4] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashock, and H. Balakrishman, "Chord: A scalable peer-to-peer lookup protocol for internet applications," In *Proceedings of the ACM SIGCOMM*, pp. 149-160, San Diego, California, August 2001.

[5] A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: supporting scalable multi-attribute range queries," In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '04)*, Portland, Oregon, USA, 2004.

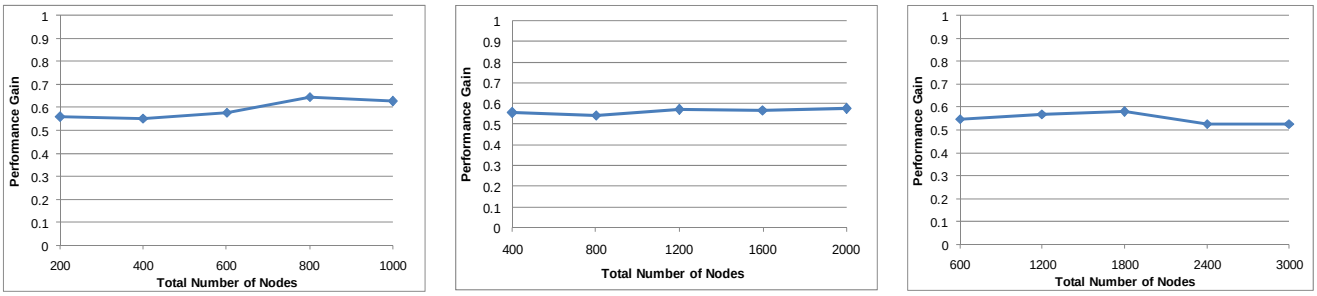


Fig. 11 Performance gain in response time over traditional single overlay approach for cases: (a) C1 (b) C2 and (c) C3

[6] G. S. Manku, M. Bawa, P. Raghavan, "Symphony: Distributed hashing in a small world," *USENIX Symposium on Internet Technologies and Systems*, 2003.
[7] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat, "Scalable wide-area resource discovery," *UC Berkeley Technical Report UCB/CSD-04-1334,* July 2004.
[8] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," *Proceedings of the USENIX Annual Technical Conference*, June, 2004.
[9] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware),* Heidelberg, Germany, pages 329-350, November, 2001.
[10] S. Basu, S. Banerjee, P. Sharma, and S. Lee, "NodeWiz: peer-to-peer resource discovery for grids," In *Proceedings of the Fifth IEEE international Symposium on Cluster Computing and the Grid (CCGrid'05)*, 2005.
[11] A. S. Cheema, M. Muhammad, and I. Gupta, "Peer-to-peer discovery of computational resources for Grid applications," *Proceedings of the IEEE/ACM Workshop on Grid Computing (GRID)*, 2005.
[12] 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, n. 7, pp. 864-878, Elsevier Science, August 2007.
[13] R. Ranjan, A. Harwood and R. Buyya, "A Study on Peer-to-Peer Based Discovery of Grid Resource Information," *Technical Report, GRIDS-TR-2006-17, P2P Networks Group and Grid Computing and Distributed Systems Laboratory,* The University of Melbourne, Australia, Nov. 10, 2006.
[14] P. Maymounkov and D. Mazieres, "Kademlia: A peerto -peer information system based on the XOR metric," In *Proceedings of IPTPS02*, Cambridge, USA, March 2002.
[15] V. Muthusamy, and H. Jacobsen, "Small scale peer-to-peer publish/subscribe," *The 2nd international workshop on P2P knowledge management (P2PKM 2005)*, San Diego, CA, USA, 2005.
[16] A. Mislove and P. Druschel, "Providing administrative control and autonomy in peer-to-peer overlays," In *3rd International Workshop on Peer-to-Peer Systems*, San Diego, CA, Feb. 2004.
[17] D. Talia and P. Trunfio, "Web Services for Peer-to-Peer Resource Discovery on the Grid," *DELOS Workshop: Digital Library Architectures,* 2004, pp. 73-84.
[18] H. Sagan, "Space-Filling Curves," *Springer,* 1994.
[19] S. Ramabhadran, S. Ratnasamy, J. M. Hellerstein, and S. Shenker, "Brief announcement: prefix hash tree," In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing (PODC '04),* St. John's, Newfoundland, Canada, 2004.
[20] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.* 35, 2 (Jun. 2003), pp. 114-131.
[21] D. Talia, P. Trunfio, and J. Zeng, "Peer-to-Peer Models for Resource Discovery in Large-scale Grids: A Scalable Architecture," *Proceedings of the 7th International Conference on High Performance Computing in Computational Sciences (Vecpar 2006)*, Rio de Janeiro, Brazil, LNCS, vol. 4395, pp. 66-78, Springer-Verlag, 2007.
[22] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An approach to universal topology generation," In *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS'01),* Cincinnati, Ohio, August 2001.
[23] B. Huffaker, E. Nemeth, and K. Claffy, "Otter: a general-purpose network visualization tool," *In Proceedings of the 9th Annual Conference of the Internet Society (INET'99),* 1999.
[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "*Introduction to Algorithms*," Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 24.3: Dijkstra's algorithm, pp.595–601.
[25] CentOS, http://www.centos.org
[26] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić, J. Chase, and D. Becker, "Scalability and accuracy in a large-scale network emulator," *Proceedings of the 5th ACM/USENIX Symposium on Operating System Design and Implementation (OSDI),* Boston, MA, December 2002.
[27] PlanetLab, http://www.planet-lab.org/

Hesham A. Ali holds a B.Sc. degree in Electrical Engineering and a M.Sc. in Automatic Control Engineering from Mansoura University, Mansoura, Egypt. He received his Ph.D. degree in Computer Engineering from the same university in 1997. Currently, he is Associate Professor at the Computers and Systems Engineering Department at the Faculty of Engineering, Mansoura University, Mansoura, Egypt. His research areas include Image Processing and Pattern Recognition, Internetworking and Grid Computing, Distributed Databases, and Mobile Agents.

Mofreh M. Salem received his B.Sc. and M.Sc. degrees in Electrical Engineering from Mansoura University, Mansoura, Egypt, in 1974 and 1979, respectively. He received his Ph.D. degree in Electrical and Electronic Engineering from the University of Strathclyde, Glasgow, UK in 1985. He is currently Professor and Head of Computers & Systems Engineering Department at the Faculty of Engineering, Mansoura University, Mansoura, Egypt. His research areas include Database Systems, Operating Systems, and Computer Communication Networks.

Ahmed A. Hamza received his B.Sc. degree in Electronic Engineering from Mansoura University, Mansoura, Egypt, in 2003. He is currently a research assistant and M.Sc. candidate at the Computers and Systems Engineering Department at the Faculty of Engineering, Mansoura University, Mansoura, Egypt. His areas of interest include Parallel and Distributed Computing, Grid Computing, Resource Management, Resource Discovery and Peer-to-Peer Networks.