July 2003, Vol.18, No.4, pp.413–422 J. Comput. Sci. & Technol.

VEGA Infrastructure for Resource Discovery in Grids

GONG YiLi (龚奕利), DONG FangPeng (董方鹏), LI Wei (李 伟) and XU ZhiWei (徐志伟)

Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100080, P.R. China

E-mail: gongyili@ict.ac.cn

Received January 26, 2003; revised May 28, 2003.

Abstract Grids enable users to share and access large collections and various types of resources in wide areas, and how to locate resources in such dynamic, heterogeneous and autonomous distributed environments is a key and challenging issue. In this paper, a three-level decentralized and dynamic VEGA Infrastructure for Resource Discovery (VIRD) is proposed. In this architecture, every Border Grid Resource Name Server (BGRNS) or Grid Resource Name Server (GRNS) has its own local policies, governing information organization, management and searching. Changes in resource information are propagated dynamically among GRNS servers according to a link-state-like algorithm. A client can query its designated GRNS either recursively or iteratively. Optimizing techniques, such as shortcut, are adopted to make the dynamic framework more flexible and efficient. A simulator called SimVIRD is developed to verify the proposed architecture and algorithms. Experiment results indicate that this architecture could deliver good scalability and performance for grid resource discovery.

Keywords grid architecture, resource discovery, resource information propagation, Vega Grid

1 Introduction

A grid environment contains a large collection of different types of resourse distributed in a wide area. These resources are owned and operated by various organizations with heterogeneous administrative policies. Resources can join and leave a grid at any time, and their status may change dynamically. Resource discovery, the problem of locating resources that satisfy users' requests efficiently and optimally, is an important and challenging issue in such a grid environment. A solution to the resource discovery problem must efficiently deal with scalability, dynamic changes, heterogeneity, and autonomous administration. In this paper, we will focus on the issues of scalability and dynamic change.

The Institute of Computing Technology is developing a number of integrated technologies and systems, jointly called VEGA Grid[1,2], to provide *Versatile Service*, *Enabling Intelligence*, *Global Uniformity* and *Autonomous Control*. This paper presents an approach to the grid resource discovery issue, called VEGA Infrastructure for Resource Discovery (VIRD). Our approach comprises a three-level architecture, a URN-based resource naming scheme, a two-level algorithm for propagating resource information changes, and two schemes for locating a resource recursively or iteratively.

The VIRD architecture is a three-level hierarchy: the top level is a backbone consisting of Border Grid Resource Name Servers (BGRNS); the second level consists of several domains and each domain consists of Grid Resource Name Servers (GRNS); the third level consists of all clients and resource providers. There is no central control in the VIRD. The backbone is responsible for inter-domain resource discovery. Within a domain, resource providers register themselves to GRNS servers, and clients acquire required resources through GRNS servers.

Some mechanisms are also proposed to adapt the VIRD to the dynamic characteristic of the grid. Resources are named as objects, i.e., attribution-value pairs. A link-state-like algorithm is used to update and propagate resource information quickly. A shortcut mechanism allows the topology of the GRNS servers to change dynamically.

We have developed a configurable, event-driven simulator called SimVIRD to verify our architecture and resource information propagation algorithm. Experiment results show that the viability of the VIRD approach depends largely on a metric called *resource frequency*, while being relatively insensitive to the size of a grid.

This work is supported in part by the National Natural Science Foundation of China (Grant No.69925205), the National High-Tech R&D 863 Program of China (Grant No.2002AA104310), and the Chinese Academy of Sciences Overseas Distinguished Scholars Fund (Grant No.20014010).

The rest of this paper is organized as follows. Section 2 reviews existing work on resource discovery. Section 3 outlines the VIRD approach and explains its components and mechanisms in detail. Experiments with the SimVIRD are presented in Section 4. Section 5 offers the conclusions.

2 Related Work

There has been much work done in the field of resource discovery in grid environments.

The information service in Globus Project is MDS[3,4]. First, the data representation in the directory service is globally uniform. In fact, every resource discovery and allocation system can have its own policies, and hence an identical information denotation would bring it some limitation and inconvenience. With the permission of a system-specific information organization, the system can optimize its database to improve convenience and achieve best storing and searching performance. Second, in the MDS architecture, information, along with information servers, is organized in the strictly tree-like topology with a comparatively fixed relationship; while the resource distribution may change dynamically. Third, generally, requests should be answered by remote servers storing the required resource information. If the information is locally available, the performance would be improved. Fourth, the directory service used is LDAP[5], which is designed for reading rather than writing, while in grid environments frequent changing may become a problem for this architecture.

The authors of [6] combine peer-to-peer technologies with grids and apply them to resource discovery. The P2P architecture is fully distributed and all the nodes are equivalent. But all request-forwarding algorithms they have proposed: random, experience-based plus random, best neighbor or experience-based plus best-neighbor, cannot change the plight that every node has little knowledge about the distribution of resources within the gird and their status. So, generally, they are less effective. In particular, when the types of requested resources are vast and the work set is very big, the miss rate increases because the past experience does not work.

[7] presents a grid resource discovery model based on the routing transferring method, in which all the routers are au pair and have no difference structurally, and the adopted algorithm is RIP-like distance vector routing algorithm. The problems of the algorithm will be discussed in detail in Subsection 3.2.2.1.

Next section will present our resource discovery framework in three main aspects: resource naming, resource information propagation and resource discovery, where several approaches that have been used by other distributed systems, e.g., IP routing systems, DNS, etc., are referenced.

3 The VIRD Approach

VIRD adopts a three-level hierarchy as shown in Fig.1: the top level is a backbone consisting of Border Grid Resource Name Servers (BGRNS); the second level has several domains and each domain consists of Grid Resource Name Servers (GRNS); and the third level consists of leaves which include all clients and resource providers.

• A Border Grid Resource Name Server (BGRNS) has connections to both the backbone and one or several domains, and exchanges information with them. When locating resources, BGRNS servers forward requests among domains and help to find out qualified resources.

• A Grid Resource Name Server (GRNS) dynamically collects information about resources registered to it, and spreads information to other GRNS servers. A GRNS receives requests from

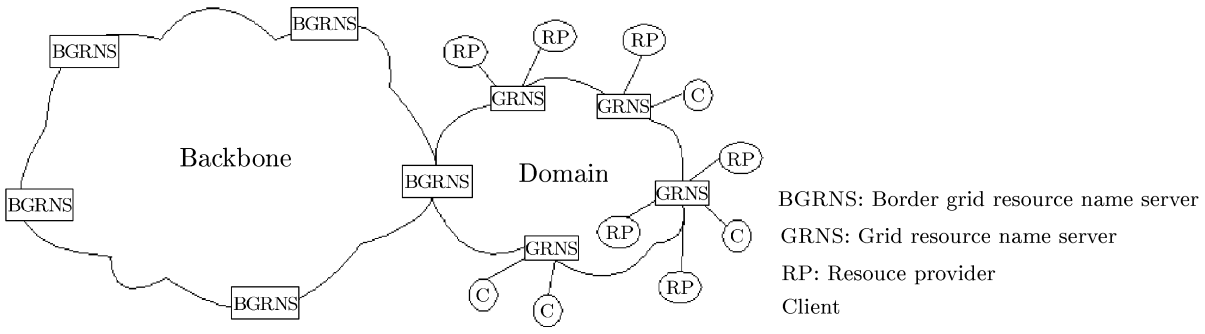

Fig.1. The three-level VIRD architecture: A backbone, domains and leaves.

clients, finds proper resources either by its local knowledge or through BGRNS servers, and then returns the satisfying resource providers to the clients.

• A Resource Provider registers itself to a GRNS, which is called the resource provider's designated GRNS, and then it will report its status information periodically and upon changes. Meanwhile it receives and serves user requests.

• A client sends resource requests to GRNS servers and receives replies. Of course a client can also communicate with resource providers: sending requests, handling responses, and using resources.

A host can act as both a client and a resource provider at the same time.

Above all, an important component in the architecture is resource naming that concerns resource representing, requesting, searching and discovering.

3.1 Resource Naming

Resource naming in grids has some characteristics corresponding to the way grids work. 1) In most cases, a user requires some resources qualified for a certain condition, while in advance the user has no idea of their locations. Furthermore, in general, given a request there are more than one satisfactory resource. 2) Resource status is changing dynamically. A user request not satisfied at one time might be satisfied at other time, and vice versa. 3) Both the number and the types of resources are changing. There are a lot of resources joining and leaving a grid at any moment. As the grid grows, more and more types of resources would be available and the resource space might explode.

For the above reasons, it is improper and not powerful enough to use current URL as the way to address resources. In grids, resources should be referenced by name (uniform resource name, or URN). Identifying resources by name allows users to access resources in a variety of ways without understanding the internal structure.

There are four key design goals for the naming scheme in VIRD. First, the naming scheme should be independent of our architecture. This allows users to make use of our infrastructure without having to know the details of the architecture and similarly allows the naming scheme to be adopted by other systems dealing with grids. Second, the naming scheme is of good scalability. All resource names in our scheme are in the form of objects that allow thousands of classes and instances. Third, the naming scheme should allow resources to be identifiable to a fine level of granularity but should not require such details that are not necessary. Fourth, our naming scheme should be user friendly. The resource names are the tools for users to access grids, hence the naming scheme must be convenient to use.

```
⟨res_name⟩ ::= (⟨class_name⟩|⟨class_id⟩) ["?"
                    ⟨specification⟩]
⟨class_name⟩ ::= ⟨string⟩
⟨class_id⟩ ::= ⟨number⟩
⟨specification⟩ ::= (⟨attribute⟩ ⟨cop⟩⟨value⟩) {⟨bop⟩
                    (⟨attribute⟩⟨cop⟩⟨value⟩)}
⟨attribute⟩ ::= ⟨string⟩
⟨value⟩ ::= ⟨string⟩|⟨res_name⟩
⟨bop⟩ ::= ">" | "=" | "<" | ">=" | "!=" | "<=" |
include | exclude | satisfy
⟨cop⟩ ::= AND | OR | NOT
```

Fig.2. The syntax of our naming scheme in BNF. The class name or class id specifies a resource type. The specification is attribute-value pairs and indicates the qualification for resources.

The naming scheme is to divide resources into classes and define attributes for every class. The syntax of our naming scheme in Backus-Naur Form, or BNF, is shown in Fig.2. A resource name consists of two parts: class name or class id, and specification, which collectively identify a user's requirement.

A user indicates the type of the requested resource by class name or id. The class name is a string friendly for users to remember, while the class id is a number and is used for the sake of performance. With some user interfaces, e.g., Web portals, users can fill out resource requests easily. In the interior of the system, all resource types are represented by ids. The class name or id is mandatory and could be further qualified with the following specification.

The specification is a logic expression consisting of attribute-value pairs and is linked together by operators. This field is optional and when it is null it means that a user needs only resources of one kind no matter what the properties are. Here the operators "include" and "exclude" respectively mean that the attribution should contain or should not contain the specified value.

As the definitions in Fig.2 show, types of class's attributes could be classes, too. For instance, as shown in Fig.3, class ComputingResource has an attribute OS, and the attribute OS might be an instance of class OperatingSystem with members — type, version, supportedlibrary, etc. And a specification can just be seen as an object instance.

The "satisfy" operator denotes that the attribution should meet the qualification of the following object value.

```
ComputingResource?(CPU>"933MHz")AND (memory =
"256M") AND (OS satisfy (OperatingSystem ? (type =
"Linux") AND (version = "RedHat7.2")))
```

Fig.3. An example of resource name.

Our scheme just enables us to name individual resources. However, to request multiple numbers or types of resources at one time a request language is needed, which is not discussed here. Namespace and inheritance can also be introduced to enhance its generality and extensibility. The interested readers could also refer to [8–10].

3.2 Information Propagation Mechanisms

Information propagation in the VIRD is mainly responsible for resource information generating (registering and updating) and spreading within the infrastructure.

3.2.1 Two-Level Topology

It is impossible for one server to know about the whole grid, while it is practical and preferable for a node to know a part, especially the information about those servers and resources that have close relationship with it. Thus a two-level GRNS architecture is adopted. BGRNS servers and the connections between them build up the backbone and each BGRNS is also connected with one or several domains so that it joins the backbone and the domain(s). In a domain all the GRNS servers are independent except that there might be update information between each other.

This two-level architecture is preferred due to the following reasons.

• It has good scalability. Based on the experience of the Internet routing, the number of GRNS servers in a grid could be up to thousands.

• Every GRNS could have its own information organization. All servers are independent of each other and every server can put the local information into optimal structure and can have its own managing and searching policies.

• It would be more effective. Each GRNS knows all the knowledge about resources in a domain and most requests can be answered locally. Only those conditions that cannot be satisfied in the domain would be forwarded to a BGRNS.

• The conception domain in the VIRD architecture somehow conforms to the virtual organization (VO)[11] and can be integrated organically.

3.2.2 Information Propagation Algorithms

Consistent with the VIRD architecture, the information propagation protocols are classified into interdomain protocols and intradomain protocols.

3.2.2.1 Existing Resource Routing Algorithm

In [7], all the resource routers are au pair and have no difference structurally. It can be inferred that their algorithm is a RIP-like distance vector algorithm. Thus, this approach is simple and easy to implement, whereas it has some problems. 1) First of all, the distance vector routing algorithm is slow to converge[12]. 2) The authors experimented with a model composed of 5,000 nodes simulating the real network. But according to the practice and experience from IP routing, it is impractical to build such a big network of routers solely by RIP. 3) The requested router only knows that the biggest or best resource attributions in a certain direction and has little information about the details of individual resources, so it might lead to incorrect routing.

3.2.2.2 Intradomain Information Propagation Algorithm

Our intradomain information propagation algorithm consists of the following parts.

• Finding Neighbors: In this paper, a GRNS A's neighbor refers to a GRNS that directly connects with A in the topology and is a logical concept, different from the neighbor in IP routing. The starting neighbors of a GRNS may be predefined statically and a server reads them from a configuration file during startup. Then it tries to say hello to its neighbors to verify whether they are alive and ready to establish the real connections.

A new feature mechanism called shortcut is proposed. The main idea is that new connections between GRNS servers could be added dynamically according to some conditions. For example, in the case that the resources registered to a certain GRNS, R1, have been frequently used by the recent requests of GRNS R2, R2 could ask R1 to establish a shortcut between them. If agreed, updating information can be sent from R1 directly to R2, therefore time will be saved and information on R2

about R1 will be more up to date. The shortcut should not be written into the configuration file due to its dynamic characteristic and could be deleted when it is not necessary any more.

• Packing an RSP: Each GRNS constructs resource state packet, or RSP. A GRNS generates an RSP periodically or when it discovers that it has a new neighbor, the status of its registered resources has changed or a neighbor has gone down. If triggered update is used and triggering events have not happened for a certain long time, a GRNS must send an empty RSP to declare its living, or its neighbors would think that it has been down.

• Disseminating the RSP to all GRNS servers: Each GRNS keeps track of the sequence number it used last time when it generated an RSP; when it needs to generate a new one, it uses the next sequence number. When another GRNS receives an RSP, it compares the sequence number of the received RSP with the one from its memory (if it has one there) and assumes that the one with higher sequence number is more recently generated. And the sequence number would wrap around.

The advantages and features of our intradomain information propagation algorithm include that 1) our link-state-like algorithm converges more quickly than distance vector algorithms[12]; 2) it makes the resource discovery faster becauser some information needed to choose the resource provider for a request is locally available; in contrast, with a distance vector algorithm, a GRNS only knows that a requested resource is in a certain direction, and the request has to be forwarded to the corresponding neighbor; 3) the topology can be consciously changed when necessary.

It is not supposed that GRNS or BGRNS servers know the resource information about other domains they do not belong to, hence no such information is propagated between domains.

3.3 Resource Discovery Mechanism

The main function of the resource discovery mechanism is that GRNS servers answer requests from clients and find satisfactory resources.

3.3.1 Resource Discovery Process

The requests come in two flavors: recursive and iterative, also called nonrecursive.

3.3.1.1 Recursion

Recursion, or recursive discovery, is just a name for the discovery process used by a GRNS when it receives recursive requests, as shown in Fig.4.

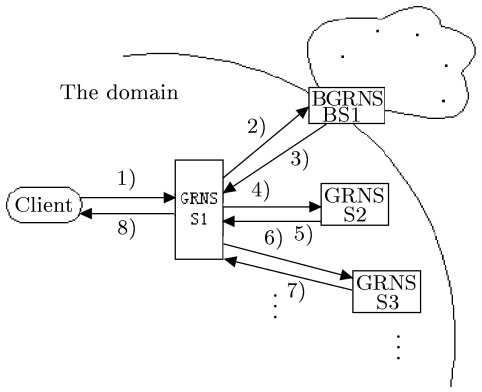

Fig.4. Discovery process for a recursive request.

The recursive process is as follows.

1) The GRNS S1 receives a recursive request. S1 first searches its local database of resource information for the satisfactory resource. If no qualified resource in this domain is found, S1 will forward the request to a BGRNS BS1 and go to 2); if found, S1 chooses an appropriate one from the candidates and if the chosen resource is local to S1, it will reserve the resource and return the reservation handle, go to 8); or else go to 4).

2) BS1 receives the forwarded request, finds an answer according to some mechanisms and returns it to S1. The interdomain resource discovery will be discussed in 3.3.1.3, and the details are omitted here.

3) S1 gets the reply from BS1 and go to 8).

4) S1 sends an acknowledge-and-reserve request to S2. If the resource satisfies the request, S2 will keep the corresponding resource for that client and send back a reservation handle. Otherwise, S2 will tell S1 that the resource is no longer qualified.

5) S1 receives the reply and if the reply is a reservation handle, S1 returns the handle, go to 8); if the reply implies disqualification of the resource, go to 6).

6), 7) S1 repeats the procedure similar to 1)–5) until the qualified resource is found or the time is out. Go to 8).

8) S1 returns the answer to the client either the reservation handle or the "no qualified resource is found" message. The resource discovery process stops.

Recursive queries place most of the burden of resource discovery on a single GRNS.

3.3.1.2 Iteration

Iteration, or iterative discovery, on the other hand, refers to the discovery process used by a GRNS when it receives iterative requests (Fig.5).

Although distinct from each other, the iterative process is somewhat similar to the recursive one.

Hence a general description instead of the details of each step is given here. The designated GRNS is only the initial place for the client to ask questions. If there is satisfactory resource registered to the GRNS it reserves the resource and returns a handle; if the resource is under some other GRNS in the domain the GRNS will return the address of that server; if there is no satisfactory resource in the domain the GRNS will return the address of a BGRNS. Then if the answer is not a reservation handle the client will continue the request.

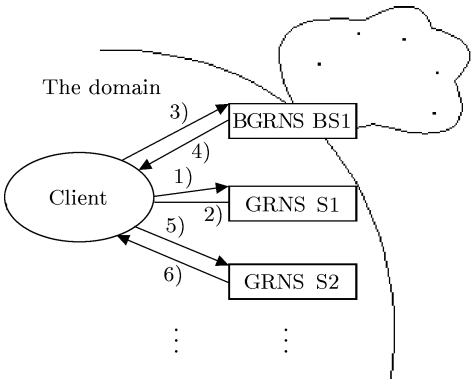

Fig.5. Discovery process for an iterative request.

The two methods (recursive and iterative) are applicable and specific to different situations. They coexist in grids.

3.3.1.3 Interdomain Resource Discovery

Since every BGRNS belongs to one or several domains, it has full knowledge of the resource information of the domain(s). When a resource request cannot be satisfied in a domain, it will be forwarded to the BGRNS connecting the domain to the backbone. The BGRNS broadcasts it to all or some other BGRNS servers. If a BGRNS finds a proper resource in the domains it knows, it will send an answer to the initial BGRNS. Then the initial BGRNS will choose the proper one from all the answers.

3.3.2 Searching and Choosing Resources

Searching, to find qualified resources for requests, has close relationship with organization of databases on GRNS servers. The VIRD architecture puts no requirement on information organization in GRNS servers, i.e., every server can have its own information format, storing and accessing methods and can combine them with the searching method to achieve best performance.

Given the whole knowledge of a domain and a set of qualified resources, there are several ways to pick out an appropriate one based on various metrics. The following is some commonly used policies.

• Choosing the most lightly loaded resource: taking the load status as the criterion, the most lightly loaded one is the most preferred.

• Choosing the "closest" resources: every edge between two GRNS nodes is given a weight and a weighted graph of the domain is obtained. Apply straightforward algorithms (e.g., Dijkstra algorithm) to calculate the distance between each pair of nodes. Metrics for the edges can be defined by fixed numbers, hops, or communication latency, etc.

• Choosing resources according to static preference policies: this measurement looks somewhat stiff, but it conforms to some practice in the grid, even in the Internet. There would be thousands of resources and users in girds, and some of them belong to the same organizations, realistic or virtual. Due to considerations on factors such as cost, effectiveness and security, some resources would be more appropriate than others even with the same capability.

There may be many other metrics and implementers can define their own metrics based on their situation or on the users' demand. All metrics can be implemented and deployed independently or employed together.

4 Experiments

In this section, the experimental environment is presented, including our simulator and assumptions. We also present the details about the inputs and the parameters, as well as the experimental results.

4.1 Experimental Environment

Because there are no standards for evaluating a resource discovery system in a grid, and our architecture is original and quite different from those in [3, 4, 6, 7], it is difficult to compare its performance with that of the others. So far, our experiments aim at verifying our three-level architecture for resource discovery by setting up and using an emulating grid environment. Considering that our interdomain policies are similar to common peer-to-peer technologies, and much work[6,13,14] has been done in this area, our experiments in the current stage are only focused on the intradomain algorithm.

Because the key of our approach is the dynamic propagation of the resource update information, the unscalable, static, discrete event simulators[15,16] cannot be used. Aiming at our needs, we have designed and implemented a configurable, event-driven grid simulator, SimVIRD. With SimVIRD, one can configure the parameters related to the resource discovery performance, such as the topology of GRNS servers, the request trace, the resource distribution, the resource information update period, etc.

A resource discovery system is very complicated and influenced by many factors. To simplify the problem, some assumptions are predefined based on the goals of our simulation. 1) The latency of locally searching resource information is ignored, because it is negligible compared to the network latency, and the focus of our simulation is to verify the overall architecture instead of the local searching policy. 2) All the requests are of recursive flavor. Since interdomain resource discovery has not been simulated, no TTL (Time to Live) is considered and the retry time for requests is set to 3, i.e., on receiving a request, a GRNS will try to find it an answer in the domain and try to acknowledge the corresponding resource for at most three times. If it fails all the time, it will forward the request to a BGRNS. 3) Resources are modeled as having only one dynamic attribute in the form of a decimal fraction ranging from 0 to 1. 4) Accordingly, requests are simplified to require only one resource whose attribute is smaller than a certain value.

While these assumptions are not realistic, it helps us to understand the effectiveness of our architecture and the impacts of main parameters on the resource discovery performance.

4.2 Experimental Inputs and Parameters

The performance of the resource discovery mechanism depends on the following factors.

4.2.1 Topology

The topology of GRNS servers influences the performance of our resource discovery approach. In the extreme, if a node receives RSPs directly from all the other nodes, it is most possible that the node can make the best decision for user requests. However the influence of them on the discovery performance is beyond the discussion of this paper. In our simulation, the GNRS topology and the network latency between GRNS nodes used in our simulation are generated by the Tiers network generator[17]. It is assumed that there are no failures in communication between nodes.

4.2.2 Resource Distribution

There are two main factors about resource distribution. The first is the distribution of resources in a domain. Sometimes resources are centralized in a small region, while there are few elsewhere. Sometimes they scatter evenly. For generality, a resource is randomly chosen to reside on a GRNS node in our simulation. The second issue is resource frequency. Some resources are very common, of large number and widely available, while some others are rare. For a certain type of resources, the ratio of its number R to the number of GRNS nodes in the domain N is used to represent the resource's frequency. Resource discovery performance is compared using different values of R/N. The comparison results show the effectiveness of our approach under various resource frequencies.

4.2.3 User Request

User request patterns can surely make a difference to the resource discovery performance, but no real user request traces can be found for simulation, which is a problem often met in research during the design phase.

The factors shaping a user request pattern include user request distribution (what kinds of resources the user wants) and user request arriving rate. In our model, only one kind of resources is simulated, and every resource has only one dynamic attribute. This simplification is reasonable in terms of the principles of the architecture. A request requires a resource whose dynamic attribute is smaller than a randomly generated value. The request arriving rate is set to 1,000 according to a common Web server's request receiving rate, i.e., a GRNS receives a request randomly within every 1,000 ticks, where a tick is just a general time unit.

In each simulation, every GRNS reads requests from a trace file. To keep the experiment long enough to simulate the real running process of our model, the request number is set to (100 $*$ the resource information update period)/the request arriving period, where the resource information update period will be detailed in the following section. For example, when resource information update period is 50,000 time units, the trace file of every GRNS has 5,000 request entries.

4.2.4 Update Periods

The update periods refer to the period of updating the resource information and the period of changing the resource state. In our experiments, a GRNS periodically sends to its neighbors the new RSPs generated and received by it. Resources change their status for every resource state change period.

4.3 Results

We are interested in the following four statistical parameters.

1) In_domain rate: the rate of requests that can be satisfied in the domain.

2) Forwarding rate: the rate of requests that cannot be satisfied in the domain and should be forwarded to a BGRNS; its value is equal to (1 - in_domain rate).

3) Forwarding error rate: the rate of forwarded requests that can be satisfied in the domain at that moment.

4) Looking-up error rate: the rate of looking up results that have been acknowledged as wrong by the designated GRNS of the found resource.

Because some of our parameters in the simulation are random values generated during runtime, in order to avoid errors caused by randomness, every experiment is repeated 200 times and all the results are the averages.

Fig.6 gives looking-up error rates for various ratios between the resource information update period and the resource state change period. As the ratio grows, the amount information on a GRNS about other GRNS nodes decreases in time, hence the possibility of getting wrong looking-up results increases. The intuition is confirmed by Fig.6. It can be seen that for all the resource state change periods the looking-up error rates grow along with the ratio between the resource information update period and the resource state change period.

The influence of the node number and the resource frequency on the rate of requests that can be satisfied in the domain is shown in Fig.7. When the resource frequency is 1/10, the rate is between 54.64% and 59.09%. When it is 10, the rate can be as high as 97.8%. For a number of testings, the environments are similar to this one except that the ratios between the resource information update period and the resource state change period are different (0.5 and 2), and similar pictures are obtained. They are not presented here due to the space limit.

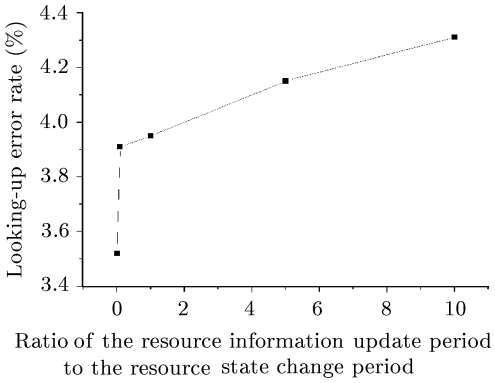

Fig.6. The looking-up error rate as a function of the ratio of the resource update period to the resource state change period. The resource state change period is constant: 100,000; the ratios of them are 0.01, 0.1, 1, 5 and 10 respectively. The node number is 100 and the resource frequency is 10.

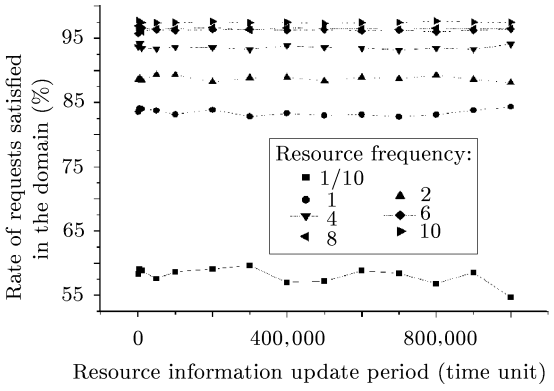

Fig.7. In_domain rates in the environments with different resource information update periods and resource frequencies. The node number is 100. The ratio between the resource state change period and the resource information update period is 1.

In our experiments the request numbers for different resource frequencies are all the same; while in reality the requests for rare resources should be much fewer than those for common resources. So the overall requests forwarded for interdomain response would not be too many and the burden on BGRNS servers would not be too heavy. These experimental data indicate that the layered architecture is acceptable in terms of the traffic on a BGRNS.

In Fig.7 it is obvious that as the resource frequency increases, the rate grows. At the same time, the rate of requests satisfied in the domain bears little relation to the resource information update period. In the environments with other node num-

bers, the same outcome is obtained. (Due to the limited space, the graphs are not presented here.) The resource information update period and the resource state change period are set to 100,000 in the later experiments.

From Fig.8 it can be seen that as the GRNS number of a domain increases, the forwarding rates under different resource frequencies change little, and this shows the scalability of our architecture and algorithm. The reason for this phenomenon is that a GRNS node always tries to find a proper resource for a request as close to it as possible, so the nodes far from it have little impact on it. As for why the forwarding rates vary much when the resource frequency is 1/10, the reason may lie in the randomness for the small number of resources. Of course, the scale of a domain cannot extend unlimitedly, because as a domain grows, the overhead for the resource information propagation grows as well. On the other hand, the resources on nodes too far away may seldom, if not never, be used, and hence the overhead for such resources' information is of little meaning.

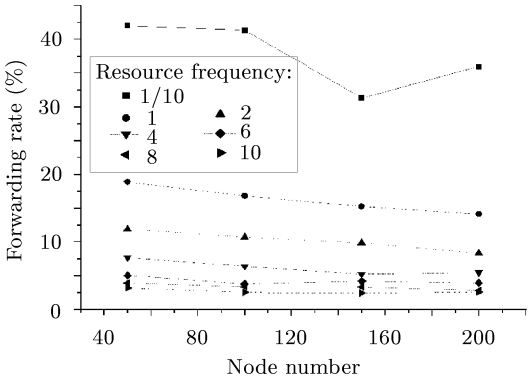

Fig.8. Forwarding rates for the environments with various node numbers and resource frequencies. The resource information update period and the resource state change period are 100,000.

Table 1 presents forwarding error rates under some environments with fixed resource update period and resource state change period. It can be seen that the forwarding error rates are rather low, and the maximum is 8%, while the minimum is just 0.06%. Together with the conclusion from Fig.7 that the forwarding rate is acceptable, it can be concluded that our architecture and resource information update algorithm are viable. Of course, further work needs to be done on the approach to reduce forwarding error rates consequently to improve the resource discovery performance.

Table 1. Forwarding error rates under some environments with various node numbers and resource frequencies. The resource update period and the resource state change period are always 100,000.

Node Number	Resource Frequency (%)						
	1/10	1	2	4	6	8	10
50	0.06	4.48	3.26	4.07	0.69	0.90	0.38
100	1.10	5.02	3.49	1.95	1.33	1.10	0.60
150	8	5.52	2.74	1.57	1.10	0.98	0.56
200	0.52	4.07	3.28	1.94	0.97	0.84	0.54

5 Conclusions

This paper presents an approach to the grid resource discovery issue, called VEGA Infrastructure for Resource Discovery (VIRD). We have developed a simulator, SimVIRD, to evaluate the VIRD. The analysis and experiment results indicate that this is a viable approach, especially with respect to scalability and dynamic change of grid resources. This architecture is proposed as a resource discovery framework. In fact it can also be applied in the resource management and scheduling possibly with a little modification.

The VIRD three-level architecture helps improve scalability by taking advantage of locality and layered distribution of resource information. In VIRD, the name of a resource is independent of its physical attributes, which facilitates the transparency of physical resources to applications. A link-state-like algorithm is adopted to propagate changes in resource information within a grid domain, which converges faster than distance-vector algorithms. This algorithm is augmented with a short-cut scheme, which could further improve the efficiency of resource information updates. All these mechanisms help deal with the dynamic change of grid resources.

This paper also defines several measures to gauge the effectiveness of a resource discovery method, including the in_domain rate, the forwarding rate, the forwarding error rate, and the looking-up error rate. When a request is used to discover a resource, there could be three outputs: (1) a resource matching the request is returned; (2) no resource is found because there is no resource matching the request; (3) no resource is returned but there exists at least one matching resource. The third case generates a wrong output. The effectiveness of a resource discovery method can be measured by the *forward error rate*, which is defined as the rate of wrong outputs in the total outputs. In our simulation, when a resource is not found in a grid domain (Cases 2 and 3), the request will be forwarded to another domain via a BGRNS.

A few conclusions can be drawn from experiment results regarding the effectiveness of the VIRD approach. The looking-up error rate grows with the ratio between the resource information update period and the resource state change period. The in_domain rate reflects locality, which has close relationship with the resource frequency. When the resource frequency is 1/10, the rate is just between 54.64% and 59.09%; when the resource frequency increases to 10, the rate can be as high as 97.8%. Because the GRNS always tries to find resources on closest nodes, the effectiveness of the VIRD approach has little relation with the number of nodes in a grid domain.

The experiments also indicate another phenomenon: the looking-up and the forwarding error rates could be significant (the highest error rate in our experiments is 8%), especially when the resource frequency is low. This suggests that resource information updating rate should not be fixed, but rather variable with respect to different types of resources. For infrequent resources, the corresponding update rate should be kept low.

References

[1] Foster I, Kesselman C (eds.). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, 1998.

[2] Xu Zhiwei, Li Wei. The research on VEGA grid architecture. *Journal of Computer Research and Development*, August, 2002, 39(8): 923–929. (in Chinese)

[3] Li Wei, Xu Zhiwei, Li Bingchen, Gong Yili. The VEGA personal grid: A lightweight grid architecture. In *Proc. the IASTED Int. Conf. Parallel and Distributed Computing and System*, 2002.

[4] Fitzgerald S, Foster I, Kesselman C *et al.* A directory service for configuring high-performance distributed computations. In *Proc. the 6th IEEE Symp. High-Performance Distributed Computing*, 1997, pp.365–375.

[5] Czajkowski K, Fitzgerald S, Foster I *et al.* Grid information services for distributed resource sharing. In *Proc. the 10th IEEE Int. Symp. High-Performance Distributed Computing* (*HPDC-10*), IEEE Press, August 2001, pp.181–194.

[6] Foster I, von Laszewski G. Usage of LDAP in Globus. http://www.globus.org.

[7] Iamnitchi A, Foster I. On fully decentralized resource discovery in grid environments. In *Proc. the Int. Workshop on Grid Computing*, 2001, pp.51–62.

[8] Li Wei, Xu Zhiwei, Dong Fangpeng, Zhang Jun. Grid resource discovery based on a routing-transferring model. In *Proc. the 3rd Int. Workshop on Grid Computing*, Baltimore, MD, 2002, pp.145–156.

[9] von Laszewski G, Helm M, Fitzgerald M S *et al.* GOSv3: A data definition language for grid information services. Grid Forum Working Draft GWD-GIS-021-001, 2002. http://www.gridforum.org.

[10] Apgar J, Grimshaw A, Harris S *et al.* Secure grid naming protocol (SGNP): Draft specification for review and comment. Grid Forum Draft Specification, February 2002. http://www.gridforum.org.

[11] Dong Fangpeng, Gong Yili, Li Wei, Xu Zhiwei. Layered resource representation in Grid environment: An example from VEGA Grid, Accepted by *the 3rd Int. Conf. Computational Science 2003* (*ICCS2003*), June 2003, Melbourne, Australia.

[12] Foster I, Kesselman C, Tuecke S. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. Supercomputing Applications*, 2001, 15(3).

[13] Perlman R. Interconnections (second edition): Bridges, routers, switches, and internetworking protocols. Addison-Wesley Publishing Co., October 1999.

[14] Gnutella protocol specification. http://www.clip2.com/articles.html.

[15] Ripeanu M. Peer-to-peer architecture case study: Gnutella network. In *Proc. the Int. Conf. Peer-to-peer Computing*, August 2001, pp.99–110.

[16] Casanova H. SimGrid: A toolkit for the simulation of application scheduling. In *Proc. the First IEEE/ACM Int. Symp. Cluster Computing and the Grid* (*CCGrid 2001*), May 2001, Brisbane, Australia, pp.430–437.

[17] Song H, Liu X, Jackobsen D *et al.* The Microgrid: A scientific tool for modeling computational grids. *Scientific Programming*, 2000, 8(3): 127–141.

[18] Doar M. A better model for generating test networks. *IEEE Global Internet*, 1996, pp.86–93.

GONG YiLi received her B.S. degree in computer science from Wuhan University in 1998. She has been pursuing her Ph.D. degree at the Institute of Computing Technology, Chinese Academy of Sciences since 2000. Her main research interests include grid, high performance servers and operating system.

DONG FangPeng is now a graduate student of the Institute of Computing Technology, Chinese Academy of Sciences. He got his bachelor's degree from the Department of Computing Science & Technology, Peking University in 2000. Since 2001, he has been working as an RA for the Vega Grid Project. His main research area includes distributed systems and grid computing. Dong will get his master's degree this summer.

LI Wei graduated from the Department of Computer Science & Engineering, Beijing Institute of Technology in 1999. From 1999 to 2000, he worked in the Institute of Computing Technology, Chinese Academy of Sciences. From 2001, he works there as an associate professor and leads a team to develop the Vega Grid Project. His main research area includes operating system, high performance computing and grid computing.

XU ZhiWei received his Ph.D. degree from University of Southern California, USA. He is a professor and deputy director of the Institute of Computing Technology, Chinese Academy of Sciences. His research interests are grid and cluster computing, high performance computer architecture and secure operating system.