A C/S AND P2P HYBRID RESOURCE DISCOVERY FRAMEWORK IN GRID ENVIRONMENTS

Yili Gong[*,+] Wei Li[*] Yuzhong Sun[*] Zhiwei Xu[†]
[*](Software Division, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)
[+](Graduate School of the Chinese Academy of Sciences, Beijing 100039)
[†] (Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)
gongyili@ict.ac.cn

Abstract

Resource discovery is crucial to efficient deployment of a grid system whose dynamic, heterogeneous characteristics make it difficult. In this paper, Vega Infrastructure for Resource Discovery (VIRD) is developed, then augmented with new features (i.e., some new algorithms) to build a C/S (client/server) and P2P (peer-to-peer) hybrid resource discovery framework. The three layered architecture of the VIRD is developed to make advantage of the physical and logical topologies of the Internet to facilitate resource discovery. With our simulations and theoretical analysis, it is proved that VIRD is of good scalability with respect to the sizes of the underlying backbone. Even when the resource density is low and the max TTL (time-to-live) is small, VIRD still achieves high search success rates in a small amount of hops. Compared with flooding and random walk algorithms via the same search success rates, VIRD outperforms them in both network traffic and response time.

1. Introduction

Resource discovery, a crucial part of a grid, refers to the process of finding satisfactory resources for user requests, including resource description, resource organization, resource lookup, and resource selection. Both dynamism and heterogeneity of a grid make resource discovery difficult.

The current resource discovery systems fall into two categories: centralized and distributed. Centralized resource management systems may have better lookup performance due to little network communication overhead, but may be of poor scalability and have a single point of failure. Distributed systems, such as peer-to-peer systems, usually are scalable, but having a barrier that the flat structure of peers does not make full use of the physical and logical topologies of the Internet. For example, users in a LAN may share similar interests; users may have faster access speeds to resources in the same MAN than to those in a different MAN; the traffic cost of users in LANs is faster and cheaper than that in the backbone. To make full use of these properties, in [1], we proposed the three-layer VIRD for the first time, however, giving no specific inter-domain routing algorithms and no experiments on the overall architecture. In this paper, VIRD is extended, and the intra-domain information update algorithm, the intra- and inter-domain request routing algorithms are completely given. In addition, theoretical analysis, and, simulations of VIRD in comparison with flooding and random walk algorithms are presented.

Regarding of the large quantity and variety of resources in grids, and the constant changes of their states and attributes, it is not enough to descript a resource by a simple string (ID), such as human unreadable hash values. Therefore the more expressive and user friendly attribute-value description is used. Users can use it to describe what they want, instead of where to find things.

By far, there exist no general standards and benchmarks for evaluating grid resource discovery systems. On users' behalf, this paper is interested in the success rate and the response time of request lookups. From the perspective of systems, the main interest goes to the network traffic. The simulation results prove that the response time almost keeps constant as the number of BGRNSes grows up. The inter-domain traffic does not grow exponentially with the size of the backbone network, which conforms to our theoretical analysis. Therefore VIRD is of good scalability with respect to the

This work is supported in part by the National Natural Science Foundation of China (Grant No. 69925205), the China Ministry of Science and Technology 863 Program (Grant No. 2002AA104310), and the Chinese Academy of Sciences Oversea Distinguished Scholars Fund (Grant No. 20014010).

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 IEEE

size of the backbone network. In addition, even with rather low resource density and small TTL_{max}, success rate of VIRD is still high. Our simulation also shows that VIRD outperforms flooding and random walk algorithms in both network traffic and response time.

VIRD presented in this paper makes three key contributions: (1) it integrates Client/Server and Peer-to-Peer architecture to make full use of the physical and logical topology of the Internet; (2) it provides an attribute-value based naming scheme for users to query resources in grids; (3) it performs better in both network traffic and response time than flooding and random walk under a given success rate.

The rest of this paper is organized as follows. Section 2 discusses some related grid and peer-to-peer resource discovery systems. The VIRD architecture and algorithms are presented in Section 4. Section 5 explains the simulation methodology, environment and results and Section 6 concludes.

2. Related Work

A lot of work has been done on the grid resource discovery. Matchmaker in Condor [3] uses a central server to match the attributes in the users' specification and those in the service providers' declaration. Such approach has a single point of failure and scales poorly. In Legion [4], Collections, the information databases, are populated with resource descriptions. The Scheduler queries the Collection and finds proper resources for applications. A few global Collections will prohibit the scalability of the system. Globus' MDS-2 [5] is a distributed resource information system: resource providers register to GIIS by the registration protocol; and users access the resource information in GIIS and GRIS by the enquiry protocol. But it does not work well when the number of resources grows and updates are common.

Peer-to-peer technologies, which have been recently popularized through file sharing applications, are divided into two categories: one is unstructured, such as Gnutella [2] and Freenet [6]; the other is structured, including Pastry [7], Tapestry [8], CAN [9] and Chord [10], etc. It is generally thought that the Gnutella protocol, which is based on broadcasting, is simple and robust, while puts heavy demand on the bandwidth and limits the scalability of the system. Both Gnutella and Freenet do not guarantee to find an existing object. In general, the structured systems have better lookup performance than the unstructured ones; they will obtain definite results within finite hops and keep scalability and autonomy as well. But because they are based on the prerequisite that every object has a unique ID and users know the IDs of the objects they want before hand, these methods lack adequate querying capability for complex resource queries.

INS [17] is a resource discovery and service location system for dynamic and mobile networks of devices and computers. INS is intended for small networks on the order of several hundred to a few thousand nodes. INS/Twine [18] is a peer-to-peer architecture for intentional resource discovery. The updating paths between resolvers may be across the whole network and the updating cost is a big problem. On the other side, Twine can do little for a category of queries whose every strand is popular.

A grid resource discovery model based on routing-forwarding is proposed and analyzed in [11]. The model may suffer from scalability problem because all the resource routers are equal peers and resource routing information need propagate across the whole network.

3. Design of VIRD

3.1. Overview

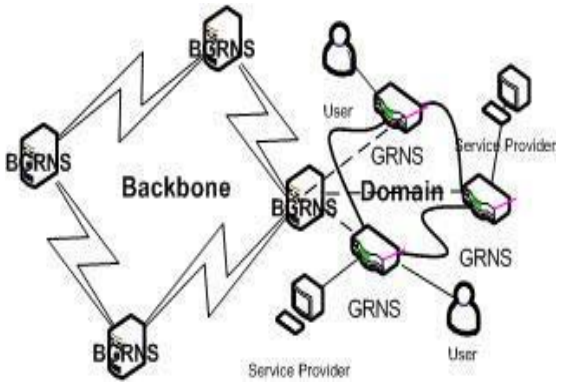

Figure 1. The VIRD architecture: backbone, domains and leaves.

A VIRD system is presented in Figure 1: the top level is the backbone consisting of BGRNSes (Border Grid Resource Name Server); the second level is domains and each domain consists of GRNSes (Grid Resource Name Server), and the bottom level is leaves which include all clients and resource providers.

BGRNS A BGRNS connects the backbone with one or more domains. It answers queries forwarded from GRNSes or other BGRNSes.

GRNS A GRNS not only knows the information about the resources registered to it, but that about all other resources in the same domain through information propagation. It also acts to queries according to its local information.

Service Provider A service provider reports its resource status periodically to its designated GRNS, i.e. the GRNS it connected to.

Client A client sends requests to a GRNS and receives responses. A host can be both a client and a resource provider.

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 **IEEE**

3.2. Topology

As mentioned before, the motivation of VIRD is to utilize the physical and logical structure of the Internet. The topology of VIRD obeys the topology of the Internet, i.e. a user or a resource provider corresponding to a host, a GRNS corresponding to a router in a domain, and a BGRNS corresponding to a border gateway. A BGRNS should be a stable, well-connected server, and a GRNS is recommended to act as a gateway of a LAN. Some kind of cache may be implemented at a GRNS to use the fact that users in a LAN may share many common interests. The interconnections between the GRNSes and BGRNSes also follow those of the routers on the Internet. Thus it can be said that the communication latency of two neighboring BGRNSes or GRNSes is generally shorter than that of two which are not neighboring. This is a difference between VIRD and many overlay networks, such as peer-to-peer networks.

When a BGRNS starts up, it queries a well known site, which returns it with a list of BGRNSes according to its IP address. The BGRNS calculates the latency between itself and them, and chooses several nearest nodes to set up neighboring connections according to its own bandwidth and/or computing capacity. While a GRNS enters, it queries the site for a BGRNS of the domain it belongs to. Then it asks the BGRNS for a current graph of the GRNSes in that domain to calculate neighbors. We note that networking of nodes might put great influence on the performance of resource discovery algorithms. We are currently exploring the "best" topology for certain algorithms and how to construct it. Users or service providers may get their designated GRNS addresses from their system administrators like the way to get Internet connections. They can also get a list of GRNS addresses from some well known site and choose a nearest one by some *ping*s.

3.3. Resource Naming

A lot of work has been done on resource naming or description, such as [3], [4], [17], [19] and [20]. VIRD implements a simple naming scheme, which specifies what resources users are looking for, instead of where to find them. The naming scheme is independent form VIRD and replaceable.

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 as follows.

```
<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
```

The following resource name describes an instance of class ComputingRresource whose attribute OS is an instance of class OperatingSystem.

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

3.4. Intra-domain Information Updating

Intra-domain updating is to propagate resource information dynamically in a domain, while there is no information propagated between domains.

Resources refresh their attribute values periodically and GRNSes disseminate the information among the domain. Expired resources will automatically be eliminated after a timeout. This soft-state approach allows a fact that resources may join and leave the system without explicit de-registration.

A GRNS constructs a RSP (Resource State Package) periodically or when one or more of the following things happen: (1) a new neighbor comes or goes; or (2) the state of some registered resource changes. If no such things happen in a given time period, a GRNS should send an empty RSP to its neighbors to show its status, or its neighbors may think that it is down.

Each GRNS keeps track of the sequence number of its last RSP. When generating a new RSP, it will use the next number. At the same time, a GRNS will also keep all latest sequence numbers of other GRNSes' RSPs, and on receiving a RSP, it will compare the sequence number of the received RSP with the one of the same neighbor in its memory, if the former is bigger, the received RSP is newer and would be used to update the local routing table, or would be discarded.

3.5. Intra-domain Resource Discovery

On receiving a request from a user, a GRNS will check its local resources first. If there is a satisfactory resource, it will reserve the resource and return a handle for the resource. If no, it will search its resource table of the domain, if finding some, it will send an acknowledgement request to that resource's designated GRNS. If the acknowledgement fails for *Threshold* times; the requests will be forwarded to a corresponding BGRNS. In our simulation *Threshold* is set to 3.

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 **IEEE**

3.6. Inter-domain Routing

The inter-domain routing algorithm of VIRD is based on the Gnutella protocol with some modification. To limit traffic in the backbone, and due to the fact that the information tends to be out of date because of the long distance between GRNSes in different domains, it is designed that a BGRNS does not store any information about resources in other domains. On the other hand, because BGRNSes are all dedicated servers, no anonymity and firewall problems are considered, the VIRD inter-domain routing algorithm is a little different from the Gnutella: a search response does not return to the requester along hops of the coming path, instead, the GRNS having the satisfactory resource responds to the initial GRNS directly. This modification may reduce the response time by half.

3.7. Performance Analysis

Table 1. Symbols and definitions.

Symbol	Definition
N	The number of nodes (BGRNSes) in the backbone network.
E	The number of edges in the backbone network.
δ	The diameter of the backbone network.
$\overline{d}$	The average outdegree of the nodes of the backbone network: $\overline{d} = 2E / N$.
$NN(h)$	The average number of nodes in a neighborhood of h hops.
N_r	The number of a kind of resources in the grid.
N_{domain}	The average number of nodes (GRNS) in a domain.
D_r	The resource density of a kind of resources, defined as $N_r /(N* N_{domain})$.
p_{match}	The probability that a resource and a request match.
$p_{success}$	The probability that a request succeeds, i.e. at least one satisfactory resource is found.
TTL_{max}	The maximum allowed value of TTL (time-to-live).

Known from the work of Faloutsos et al [12] and that of Jovanovic [13], both the routers in the Internet (whose distribution VIRD also follows) and the nodes in the Gnutella system obey all the power-laws described in [12]. In next part we will use these conclusions to analytically characterize some attributes in VIRD. The symbols used are listed in Table 1.

First, we study the number of messages related to a single request. In the worst case, six messages (three acknowledge requests and corresponding acknowledge fail responses) are generated in a domain and then the request is forwarded to the backbone, after h hops all the messages time out and no satisfactory resource is found. At this time, the number of the generated inter-domain messages, N_{inter_msg}, is

$$N_{inter\_msg} \le \overline{d} + \overline{d}^2 + \ldots + \overline{d}^h = \frac{\overline{d}\left(1 - \overline{d}^h\right)}{1 - \overline{d}} \quad (1)$$

According to Lemma 2 in [12], the number of edges, E, of a graph, can be estimated as a function of the number of nodes, N, and the rank exponent, $\Re$:

$$E = \frac{1}{2(\Re+1)}\left(1 - \frac{1}{N^{\Re+1}}\right)N \quad (2)$$

Substituting Equation 2 into $\overline{d} = \frac{2E}{N}$, it goes into

$$\overline{d} = \frac{1}{\Re+1}\left(1 - \frac{1}{N^{\Re+1}}\right) \quad (3)$$

It can be seen that when $N \to +\infty$, $\overline{d} \to \frac{1}{\Re+1}$, i.e. $\overline{d}$ has a supremum as $\frac{1}{\Re+1}$. Bring this supremum into Equation 1, it turns into

$$N_{inter\_msg} \to \frac{1}{\Re}\left(1 - \left(\frac{1}{\Re+1}\right)^h\right) \quad (4)$$

Thus in the worst case the number of messages generated by a single request is no more than $6 + \frac{1}{\Re}\left(1 - \left(\frac{1}{\Re+1}\right)^h\right)$, which is only related to h and has no direct relationship with N.

Next, let's quantify the hops, h. The average size of the neighborhood, $NN(h)$, within h hops as a function of the hop-plot exponent, H, for $h << \delta$, is

$$NN(h) = \frac{c}{N} h^H - 1 \quad (5)$$

where $c = N + 2E$.

The probability that a request can find at least one matched resource in h hops is given by

$$p_{success} = 1 - (1 - p_{match})^{D_r \times N_{domain} \times NN(h)} \quad (6)$$

Substituting Equation 5 and $E = \frac{1}{2}\left(\overline{d} \times N\right)$ into this equation, h would be given by

$$h = \left[N\left(\frac{1}{1+\overline{d}}\right)\left(\frac{\ln(1 - p_{success})}{D_r \times N_{domain} \times \ln(1 - p_{match})} + 1\right)\right]^{\frac{1}{H}} \quad (7)$$

Assuming h_{min} is the smallest integer which is no less than the number of the right side of Equation 4, if only $h \ge h_{min}$, the search success rate will be greater than $p_{success}$. From the above analysis, it can be seen that with a certain search success rate, increasing p_{match}, D_r and N_{domain} would decrease h_{min}. p_{match} depends on attributions of resources and requests, and their relationship, which can not be changed by a resource discovery system. On the other side, layering VIRD decreases the node number of the backbone and increases the node number in a

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 **IEEE**

domain, and in sequence decreases h_{min}. Of course, the node number of a domain can not be too big, and the issue of the size of a domain is discussed in [1]. In particular, from Equation 7 it is known that increasing $\overline{d}$ will decrease h_{min}, but at the mean time, it will increase the traffic that a request may bring to the network, and thus it is not desirable to decrease h_{min} by increasing $\overline{d}$.

Some people criticize Gnutella for not scaling well with the argument that if one wants to query a constant fraction of the Gnutella network, as the network grows, each node and network edge will be handling query traffic which is proportional to the total number of nodes in the network. But in reality, searching for a query is limited within a radius of TTL_{max}. To achieve a given search success rate, TTL_{max} should be set to h_{min}, which is proportional to $N^{\frac{1}{H}}$. In particular, because VIRD is made up of three layers and the flooding algorithm is only used in the backbone, the increase in the number of resources registered to a GRNS and the number of GRNSes in a BGRNS will have no influence on inter-domain searching performance except that it may increase the cost of searching the local routing table on BGRNSes.

4. Performance Evaluation

Flooding is optimal in delay for finding a resource among all unstructured P2P algorithms without respect to bandwidth limitation of underlying networks [21]. Regarding of the heavy traffic caused by flooding algorithms, random walk [21] are designed to supersede flooding algorithms in a bandwidth limited environment. Therefore we made a wide range of experiments to compare VIRD, flooding and random walk in terms of network traffic, response time and success rate by simulation.

We implemented a parameter-configurable, event-driven simulator, SimVIRD, by which we simulate our framework.

4.1. Assumptions

A resource discovery system is rather complicated and influenced by many factors. To simplify the problem, some assumptions are made: 1) because VIRD servers are more stable than nodes in a P2P network, transient nodes are not considered; 2) because running the simulator with different arguments may simulate different resources, at once only one kind of resource, computing resource, is simulated, and only one attribute of a resource, load, changes dynamically; 3) correspondingly, a request only queries a resource with load smaller than a certain value; 4) since it is implementation-specific and should be much smaller than the network latency, the processing time on a node is ignored; 5) it is assumed that the storage of a node is unlimited.

4.2. Metrics

Three metrics are considered. The former two are from the user's perspective, while the latter is from the system's perspective.

Response time ($T_{response}$): the time period from the point when a request arrives at its initial GRNS to the point when the first satisfactory resource is found. In VIRD when the resource is found, its designated GRNS will directly reply to the request's initial GRNS, while in Gnutella when the resource is found, the request will return along the coming path. To compare the resource discovery time, the response time is defined as the time to find the first qualified resource instead of that of returning to the client.

Request Success Rate: the percentage of satisfied requests of total request.

Traffic: system traffic ($Traffic_{sys}$), caused by propagating of system information and user traffic ($Traffic_{usr}$), caused by answering user requests. While for the flooding and random walk algorithms, only the latter traffic is incurred. Here the unit of traffic is the number of packages sent.

4.3. Inputs and Parameters

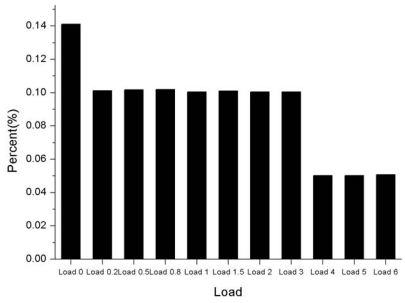

Figure 2. The distribution of the dynamic attribute, load.

The inputs of the simulations include:

Network topology: the networks of all domains and the backbone are generated by GridG [14], which extends the network generator, Tiers [15], and produces networks obeying the power laws of the Internet topology.

Resource distribution: the resources simulated are computing resources whose attributes are generated by GridG, including CPU number, CPU speed, memory size, etc, and a dynamically changing attribute, current load. It is assumed that resources are distributed across the network evenly. Resource density D_r is defined as

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 IEEE

$$D_r = \frac{\text{Resource Number}}{\text{BGRS Number} \times \text{GRNS Number per Domain}}.$$

Distribution of resource's attribute, load: the distribution of the dynamic attribute, load, is shown in Figure 2, which is calculated from the initial state generated by GridG.

Resource dynamic attribute's change period (T_{res_change}): the dynamic attribute, load, changes every period of time and the result also follows the distribution shown in Figure 2.

Resource information update period (T_{update}): every resource information update period, a GRNS will propagate the new resource information to its neighbors.

Request period ($T_{request}$) and distribution: user requests arrive randomly every user request period. The content of the request is to search for a resource with load smaller than a certain value, which also follows the distribution of Figure 2.

Maximum TTL value (TTL_{max}): this is the system default maximum TTL value.

4.4. Results

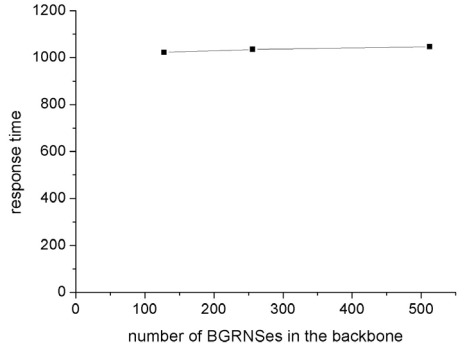

Figure 3. Response time function of number of BGRNSes in the backbone.

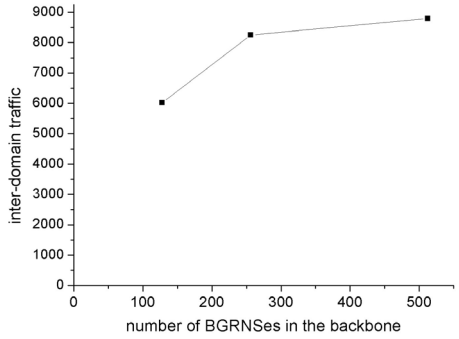

Figure 4. Number of inter-domain requests function of number of BGRNSes in the backbone.

Because some parameters in the simulation are randomly generated, every set of simulations repeats for 100 times and all results are averaged.

The influence of the ratio of T_{res_change} and T_{update} on the performance of intra-domain lookups has been demonstrated in [1]. Here we just set $T_{res\_change} = T_{update} =$ 100 ms, $T_{request} = 1$ ms, the number of the GRNSes in a domain is 64, the number of the BGRNSes is 128, 256 and 512 separately, thus the total GRNS number is 8192, 16384 and 32768.

The first group of simulations is to show the request response time and the inter-domain request number under different size of the backbone network, where $TTL_{max} = 4$. From Figure 3, observe that the response time almost keeps constant as the growth of the network. From Figure 4, it can be seen that the inter-domain traffic does not grow with the size of the backbone network exponentially. These two sets of simulation results show that the VIRD architecture has good scalability according to the size of the backbone.

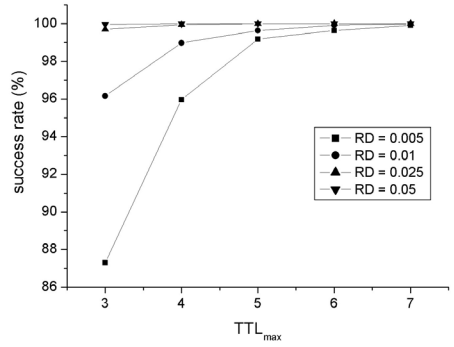

Figure 5. Success rate function of TTL_{max}.

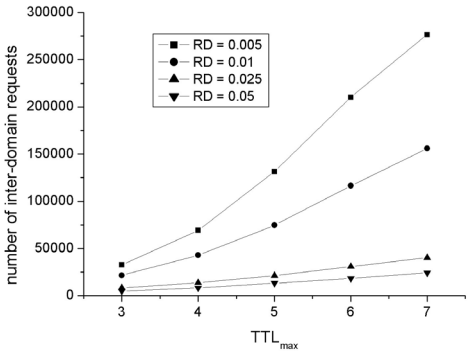

Figure 6. Number of inter-domain requests function of TTL_{max}.

The second group of simulations is about the success rate and the number of requests forwarded to the backbone under various resource densities and TTL_{max} values. The backbone used in this and following groups of simulation is a network with 256 BGRNSes, and every domain has 64 GRNSes, consequently there are 16384

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 **IEEE**

GRNSes in total. From Figure 5, observe that with the same RD, the success rate grows as the TTL_{max} grows, and with the same TTL_{max}, the higher the RD, the higher the success rate. Further more, even with rather low RD and TTL_{max}, the success rate of VIRD is still high. Figure 6 presents the relationship between the TTL_{max} and the number of requests forwarded among the inter-domain network. It is obvious that with the increasing of the RD, the number of forwarded requests decreases. Combining the two figures together, observe that with TTL_{max} = 5, 6, and 7, the increase of the success rate is not very significant (because they are already very high), while the number of forwarded requests increases dramatically. Therefore the TTL_{max} of VIRD is set to 4 in later simulations.

The third group of simulations is to compare success rate, network traffic and response time of VIRD, flooding and random walk, where RD = 0.05. Random walk uses the checking method, which means that a walker periodically checks with the original requester before walking to the next node. A walker checks once at every fourth step along the way as indicated in [21], which strikes a good balance between the overhead of the checking messages and the benefits of checking.

The data in Figure 8 and Figure 9 are ratios to the values obtained at flooding TTL_{max} = 7 (abbreviated as F7), because most Gnutella systems set their default TTL_{max} as 7.

Figure 7 shows that the request success rate of VIRD with TTL_{max} = 4 (short for VIRD4) (99.99%) is higher than that of F11 (98.95%), while that of F7 is only 87.05%. The success rate of random walk with TTL = 1000 (abbreviated as RW1000) is 100%, but its response time is 9.24 times of that of F7 and the traffic is 1.68 times of that of F7 and 6.29 times of that of VIRD4. As mentioned before, the traffic of VIRD is composed of system traffic ($Traffic_{sys}$) and user traffic ($Traffic_{usr}$). As the number of requests arriving in a certain time unit increases, $Traffic_{usr}$ increases, while $Traffic_{sys}$ remains the same. Figure 8 shows that $Traffic_{usr}$ and the sum of $Traffic_{sys}$ and $Traffic_{usr}$ of VIRD4 are only 3.48% and 26.83% of that of F7, and only 0.30% and 2.31% of that of F11. From Figure 9, the response time of VIRD4 is 97.48% of that of F7, while it is 15.25% faster than F11. The response time of VIRD4 is higher than that of flooding with small TTL_{max} values (such as TTL_{max} = 4，5 and 6), which is because the later systems only search in a small neighborhood, and their request success rates are very low (56.79%-81.03%), which are unacceptable for a grid resource discovery system. From this group of simulations, it can be seen that when VIRD4, F11 and RW300 have approximately same success rate, 99.99%, 98.95% and 98.89% respectively, the traffic of VIRD4 is 2.31% of that of F11 and 17.43% of that of RW300, and its response time is 84.75% of that of F11 and 10.55% of that of RW300. Random walk is proposed to reduce traffic in the sacrifice of response time, but VIRD performs better in both aspects: far less traffic and faster response time.

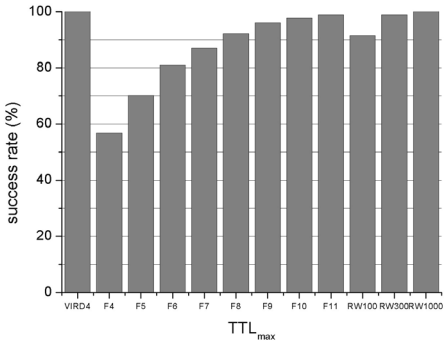

Figure 7. Success rate of VIRD, flooding and random walk under TTL_{max}. TTL for VIRD is 4, TTLs for flooding are from 3 to 11, and TTLs for random walk are 100, 300 and 1000.

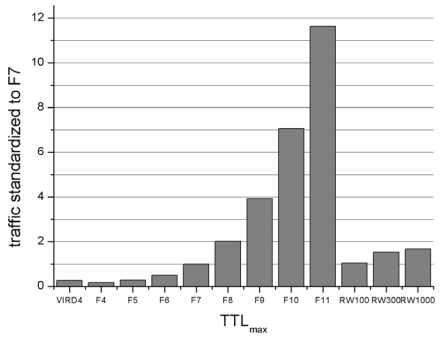

Figure 8. Network traffic of VIRD, flooding and random walk under TTL_{max}. TTLs for different approaches are the same with those in Figure 7.

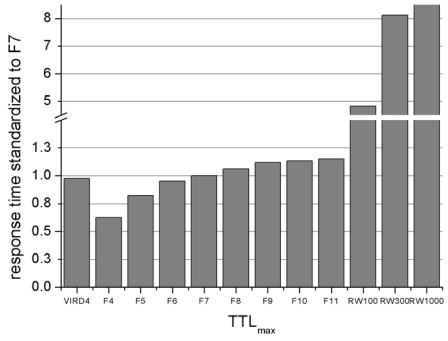

Figure 9. Response time of VIRD, flooding and random walk under TTL_{max}. TTLs for different approaches are the same with those in Figure 7.

5. Conclusion and Future work

Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)
0190-3918/05 $20.00 © 2005 **IEEE**

This paper proposes a C/S and P2P hybrid resource discovery framework, VIRD, to make advantage of the physical and logical topologies of the Internet.

According to our simulations, VIRD is of good scalability regarding response time and inter-domain traffic. Even with rather low RD and TTL_{max}, the success rate of VIRD is still high. From the comparative simulations, it is shown that with the same success rate, VIRD outperforms flooding and random walk in both network traffic and response time.

On the other side, VIRD has some disadvantages. Even if there is satisfactory resource in the current grid, VIRD may fail to find it. From the user's point of view, high request success rate may compensate for this shortcoming.

The performance impact of the network topology and cache is not discussed in this paper, which will be our future work. Additionally, the automatic recovery from BGRNS failure will be discussed.

References

[1] Y. Gong, F. Dong, W. Li, Z. Xu. VEGA Infrastructure for Resource Discovery in Grids. Journal of Computer Science & Technology, vol. 18, No.4, pp.413-422, July 2003.

[2] Clip2, The Gnutella Protocol Specification v0.4 (Document Revision 1.2), http://www.clip2.com, June 2001.

[3] R. Raman, M. Livny, M. Solomon: Matchmaking: Distributed Resource Management for High Throughput Computing. Proc. of IEEE Intl. Symp. on High Performance Distributed Computing, Chicago, USA, 1998.

[4] S. J. Chapin, D. Katramatos, J. Karpovich, A. Grimshaw. Resource Management in Legion. Future Generation Computer System, Vol. 15, No. 5: pp. 583594, 1999.

[5] Czajkowski K, Fitzgerald S, Foster I, Kesselman C. Grid information services for distributed resource sharing. In Proceedings of the 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, pp.181-194, Aug. 2001.

[6] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In Workshop on Design Issues in Anonymity and Unobservability, pp. 311–320, ICSI, Berkeley, CA, USA, July 2000.

[7] A.Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer system. In Proc. IFIP/ACM Middleware 2001, Heidelberg, Germany, Nov. 2001.

[8] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for faultresilient wide-area location and routing. Technical Report UCB//CSD-01-1141, U. C. Berkeley, April 2001.

[9] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In Proc. ACM SIGCOMM'01, San Diego, CA, Aug. 2001.

[10] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup service for Internet applications. In Proc. ACM SIGCOMM'01, SanDiego, CA, Aug. 2001.

[11] W. Li, Z. Xu, F. Dong and J. Zhang. Grid resource discovery based on a routing-transferring model. In Proceedings of the 3rd International Workshop on Grid Computing, pp.145-156, Baltimore, MD, 2002.

[12] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In SIGCOMM, pp. 251-262, 1999.

[13] M. Jovanovic, F.S. Annexstein, and K.A. Berman. Modeling Peer-to-Peer Network Topologies through "Small-World" Models and Power Laws. In TELFOR, Belgrade, Yugoslavia, Nov. 2001.

[14] D. Lu, P. Dinda, Synthesizing Realistic Computational Grids, in Proceedings of Supercomputing 2003 (SC 2003), Phoenix, AZ, Nov. 2003.

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

[16] S. J. Chapin. Distributed scheduling support in the presence of autonomy. In Proceedings of the 4th Heterogeneous Computing Workshop, IPPS, pages 22-29, April 1995, Santa Barbara, CA.

[17] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, J. Lilley. The design and implementation of an intentional naming system. In Proceedings of the ACM Symposium on Operating Systems Principles, pages 186-201, 1999.

[18] M. Balazinska, H. Balakrishnan, D. Karger. INS/Twine: A Scalable Peer-to-Peer Architect ure for Intentional Resource Discovery. In Proceedings of the International Conference on Pervasive Computing, Zurich, Switzerland, August 2002.

[19] Globus Project, The Globus Resource Specification Language RSL v1.0, http://www.globus.org/gram/rsl_spec1.html.

[20] O.Smirnova, Extended resource specification language, reference manual, http://www.nordugrid.org/documents/xrsl.pdf.

[21] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In Proceedings of the 16th ACM International Conference on Supercomputing (ICS'02), New York, USA, June 2002.