Computing (2015) 97:439–458
DOI 10.1007/s00607-014-0427-4

THRD: Threshold-based hierarchical resource discovery for Grid environments

Mahdi Mollamotalebi · Raheleh Maghami · Abdul Samad Ismail

Received: 28 October 2013 / Accepted: 4 September 2014 / Published online: 25 September 2014

Abstract The resource discovery is one of the most important services in Grid systems because providing the resources is critical to handle the applications. But some inherent characteristics of Grid environments such as large-scale and dynamicity make the resource discovery a challenging task. The hierarchical structure is widely used in Grid resource discovery but it suffers from high message load especially in upper level indexing nodes. This paper proposes a technique to reduce the message load of upper level indexing nodes in the hierarchical resource discovery. It applies a threshold value for in-process messages of indexing nodes to limit the queries passed to upper levels. The queries are checked in the sibling nodes of current level when they are not allowed to be passed to the parent node.The proposed technique is simulated in GridSim and experimented by different configurations of threshold values and number of Grid nodes. The experimental results showed that the proposed technique is able to reduce the message load of indexing nodes as 21.2 % however it also affects negatively on the response time of issued resource queries as 19.3 %. Therefore, the proposed technique is an appropriate solution for the applications with priority of lower message loads relative to the response time.

Keywords Grid computing · Resource discovery · Threshold-based · Hierarchical · Message load · Response time

M. Mollamotalebi (✉) · R. Maghami
Department of Computer Science, College of Engineering, Buinzahra Branch, Islamic Azad University, Buinzahra, Iran
e-mail: motalebi@gmail.com; mmahdi2@live.utm.my

A. S. Ismail
Faculty of Computing, Universiti Teknologi Malaysia, 81310 Skudai, Malaysia

Computing Review Subject Classification C.2.0 · C.2.1 · C.2.4

Mathematics Subject Classification 68M10

1 Introduction

Grid infrastructures that enable wide integrated use of resources have become the de facto computing platform for handling complex applications in science, engineering and commerce. Grid computing environments include several heterogeneous resources that are shared by different organizations [1,2]. The resource discovery is one of the most important services in Grid computing systems. It gathers and keeps the information about existing resources from the network, and returns the resource owners' address to the requesting applications as demand [3].

With regard to the inherent characteristics of Grid environments such as large-scale and dynamicity, the resource discovery is a challenging task. The resource discovery techniques typically deal with high message loads during the search process which affects negatively the efficiency of Grid system [4,5]. The message load refers to the number of messages that are transferred per second during the search process. In general, the resource discovery systems follow either the blind or informed strategy to search the resources. In the blind search strategy, Grid nodes do not keep information about the existing resources, nodes and their location. Thus, the resource requests are propagated to a sufficient number of Grid nodes in order to find the required resource. The search by this manner typically suffers from the high message load especially in large-scale environments.

On the other hand in the informed search, Grid nodes keep some information about the existing nodes and their resources. Thus, the resource queries are forwarded toward the resource owners as directed by using the indexed information. The directed query forwarding of the informed search strategy enables it to find the required resources quickly and with low amounts of the message load. Therefore most of todays resource discovery techniques, such as the technique proposed by this paper, use the informed search strategy.

The contribution of this paper is to provide a threshold-based technique for Grid resource discovery in the hierarchical structure that reduces the message load of upper level indexing nodes. For this purpose, it applies a threshold value for in-process messages in the indexing nodes and controls the passing of query messages to the parent nodes in the hierarchical tree. Moreover, the queries are checked in the sibling nodes of current level when they are not allowed to be passed to the parent node. By this manner, this technique reduces the message load of upper level indexing nodes however the delay of query forwarding increases the response time of the search process.

The remainder of this paper is organized as follows: Section 2 represents the existing resource discovery approaches in Grid environments and their features. In Sect. 3, description of the proposed technique is presented in details. The simulation results of proposed technique in terms of the message load and response time are presented and discussed in Sects. 4, and 5 concludes the paper.

2 Related work

In the recent years, different techniques are proposed to handle the resource discovery in Grid environments. They are classified into centralized, hierarchical, peer-to-peer-based (P2P-based), super-peer-based, and agent-based approaches. The resource discovery techniques typically act efficiently in the Grid environments with small number of nodes. But with growing the number of Grid nodes, they deal with the problems such as false positive error, high message load, high maintenance costs, and delay of the search process. In the centralized techniques [6–8], Grid nodes send their resource information changes and resource requests to the central servers. Thus, the central servers suffer from the high message loads especially when a large number of nodes exist in the Grid. Also, the central servers are prone to be the single point of failure or bottleneck.

The hierarchical techniques [9,10] act better than the centralized ones because they divide the search domain between different indexing nodes that are relating to each other as hierarchical. But they still impose high message loads on the upper level indexing nodes in the hierarchy especially in large scale and dynamic conditions.

The P2P-based approach removed the centralized supervision, and the resource information is distributed among the Grid nodes. P2P-based techniques are classified to structured and unstructured. In the unstructured-P2P-based techniques [11–13], Grid nodes join to the system without strict indexing constraints and they typically do not keep any information about the existing resources and their position. Such techniques typically use flooding strategies to discover the required resources. Thus, they suffer from high message loads especially in large scale environments.

The structured P2P-based techniques [14–16] apply some constraints on the nodes to join the system, and some indexing nodes keep the information about existing Grid nodes by using distributed hash tables (DHT). Such techniques deal with high maintenance costs for keeping the DHT indexing nodes up-to-date especially in dynamic conditions [17,18].

The super-peer-based resource discovery techniques [19–21] combine the centralized and P2P structures. In such techniques, each node has the role of either regular-peer or super-peer. The super-peer nodes are connected to each other as P2P. Also each super-peer node acts as a central server for a set of regular peers. A regular peer publishes its resource information to, and searches its required resources through its related super-peer node [22]. With regard to the limited capacity of super-peer nodes, such techniques deal with false positive errors in large-scale environments.

In the agent-based resource discovery techniques [23–25], a piece of program acts as an agent to disseminate and discover the resource information. The agent has particular characteristics such as autonomy, ongoing execution and adaptability. In such techniques, the indexed resource information is updated periodically which reduces adaptability of agent-based systems to the dynamic conditions. Moreover, agent-based techniques use flooding mechanism to send the resource information and resource requests among the network which burden high message loads on the system [5]. With regard to the hierarchical structure of the proposed technique in this paper, the existing resource discovery techniques of this structure are presented in the following paragraphs.

Elmroth and Tordsson [26] proposed a resource discovery technique which describes Grid resource information in terms of the computing and storage elements. A module named Information-Finder is responsible to discover the required resources and retrieve the details of resource information such as their usage policies. Once receiving a query, Information-Finder performs the resource locating process by querying the index servers as hierarchical. After finding the resources matched to the query, the static and dynamic information of the resources are retrieved. The parallel mechanisms of resource discovery and information retrieval are controlled by a thread pool to control the overloads. Moreover, a time limited cache is used to record the information of frequently requested resources and use them in the next search attempts in order to reduce the message load of the system. However this technique index and discovers the resources as hierarchical, but our proposed technique is different from it in the structure of indexed information such that there are not only the computing and storage categories. In our technique, the resources are completely heterogeneous and they can be of any type and sub types of resources.

Ma et al. [27] proposed a hierarchical resource discovery system based on the small-world network. In a small-world network, each node has different ranged (short or long) contacts with near and far nodes. The query forwarding is handled by long ranged contacts through the message passing between several nodes. Different hierarchical levels have their own administration node to meet the autonomous management. This technique uses redundancy to tackle with the potentially single point of failure and keep the data as consistent. Integration of small-worlds into clusters makes the message distribution faster than typical networks. However, it could provide an autonomous management for different levels of administration, but it also increases the number of messages transferred for redundancy between different small-worlds and our technique does not deal with such additional messages.

Huedo et al. [9] proposed a Grid architecture with different layers of meta-schedulers arranged in a hierarchical structure. Moreover, each target Grid is handled as a resource in a recursive way by using the same interfaces for resource management. This architecture allows a straightforward federation of Grid infrastructures observing organizational boundaries. It can also provide better controls over the shared resources. The operation inside each level of the hierarchy is autonomous and with different policies for user access control, scheduling, and resource sharing. On the other hand, it has higher overheads to handle different levels and their organizations. It uses the cache to reduce the overhead however it cannot be useful in dynamic conditions. Our proposed technique is different from this technique in the update mechanism of resource or nodes changes information such that it is not affected negatively by the dynamic conditions of the environment since the resource or node changes information are immediately distributed to upper level indexing nodes. On the other hand, this technique is similar to our technique in which the resource owners act autonomously in terms of user access control and sharing policies.

Chang and Hu [28] proposed a hierarchical technique to index and search the resource information. In its hierarchical tree, a non-leaf node is called an index server (IS). An IS needs to store who its parent node is, who its children are in the tree, and the resource information for all its children. Each IS keeps the resource information of its children nodes in the form of a bitmap. The value of the index bitmap in an indexing node is obtained by doing bitwise OR on the bitmaps of all its children. The queries are issued as bitmaps like the resource index bitmaps and each indexing node performs bitwise AND on the query bitmap and index bitmap to check the matching. This technique is similar to our proposed technique in the design of indexing nodes and relation to their children in the tree however it is also different from it such that our technique does not index the resource information of lower level nodes as bitmap. It is because our technique is able to handle highly heterogeneous resources with different sub categories. In such condition, the use of bitmaps is not useful since their size would be very long to cover all the resource types and sub types.

In this technique, if a resource with a specific numeric value could not be found, the resources with higher values will be looked and returned as the response. The index bitmaps only indicate existence of the requested resource among the descendant nodes and do not specify that the requested resource may be found in which one. So, the queries cannot be forwarded toward the resource owners as directed. The index bitmaps can be useful only in small scales because when the number of Grid nodes increases, most of the bitwise AND operations result non-zero especially in upper level indexing nodes. It causes the received queries being forwarded to all children nodes which subsequently increases the message load of the system. Our technique does not suffer from such weaknesses in large number of nodes or resources because it does not use bitmap indexes while it keeps the resource information of descendant nodes in the similar manner.

Zamanifar et al. [10] proposed a resource discovery technique with the dynamic time-to-live (TTL) considerations. The dynamic TTL is used to avoid scattering unlimited number of requests through the tree during the query forwards. In this manner, before transferring the message to children of an indexing node, TTL will be set to a small number. Then during the query forward process, the value of TTL will be increased as needed. By this way, limited numbers of the hierarchical levels are investigated. In this technique, it is necessary to know that the closest child who owns the resource can be reached after how many hops. Our technique is different from this technique in which it is independent from reachability hop count requirement information. Also, this technique acts poorly when a resource is located in lower level indexing levels since it sets the TTL values to small numbers and increases them step by step after failure of search process. But our proposed technique does not deal with such problem. Moreover, if a query requests more than one resource, it should be known that how many children have the resources. Thus, each node must provide the information about itself and its children to ancestors for each resource starting from the leaves. Therefore, increasing the number of nodes or dynamicity of the Grid environment causes the message and processing loads of TTL calculations being increased significantly while our technique is independent from TTL.

With regard to the above mentioned resource discovery techniques and other research works in the literature [5,20,29], the hierarchical structure for resource discovery in Grid environments suffers from the high message loads. This paper attempts to reduce the message load of the hierarchical resource discovery especially in the upper level indexing nodes.

3 The proposed technique

In this section, the technique which is proposed by this paper for Grid resource discovery is presented in details. The proposed technique, threshold-based hierarchical resource discovery (THRD), aimed to improve the hierarchical Grid resource discovery by reducing the message load of upper level indexing nodes.

The hierarchical resource discovery structure suffers from high message load on the upper level indexing nodes caused by the resource information update and query messages received from descendant nodes. Thus, the root node and other indexing nodes near to it are typically potential points of bottleneck in the hierarchical solutions.

3.1 Description of THRD indexing tree

THRD applies a threshold value on the number of in-process queries in the indexing nodes. The query messages that are not allowed to be passed to upper levels will be investigated in the sibling nodes. By this manner, the proposed technique reduced the message load of upper level indexing nodes however the average response time of issued queries is increased. This technique is appropriate for the applications that low message load is more important than the quickness of the search process.

The designed indexing tree of THRD is illustrated in Fig. 1. In this structure, the regular nodes which own the Grid resources are placed at level N of the hierarchical tree and they are related to the indexing nodes of level N – 1. Also, each indexing node knows its right hand side sibling node. This allows the indexing nodes to circulate the received queries among the sibling nodes as needed. The capacity of in-process messages in the indexing nodes is limited. Each indexing node maintains the resource information of all its lower level nodes and knows the right hand side sibling node on the same level.

THRD applies a threshold value for the number of query messages which the indexing nodes can process. An indexing node controls the received queries by comparing the in-process queries and the threshold value. The indexing node does not accept query messages from its children more than the threshold limit. In such case, the child node tries to find the required resource through its sibling nodes. The indexing node decreases the in-process value after finding the resource required by a query or passing the query to its parent.

The resource discovery process is initiated by a regular node in the last level of indexing tree. The regular node sends the query to its parent in level N – 1. The parent checks its local resource information index for the required resource. If the matched resource is found, it returns the resource owner address to the requester node. Otherwise it tries to send the query to its parent indexing node located at level N – 2 of the hierarchy.

The parent indexing node in level N – 2 compares its in-process messages with the threshold value. If the number of in-process messages is less than the threshold value, it accepts the query and looks up the required resource in all its branches except the one which query is received from. Otherwise it forwards received query to the right hand side sibling node and waits for the query to be returned back. Each indexing node

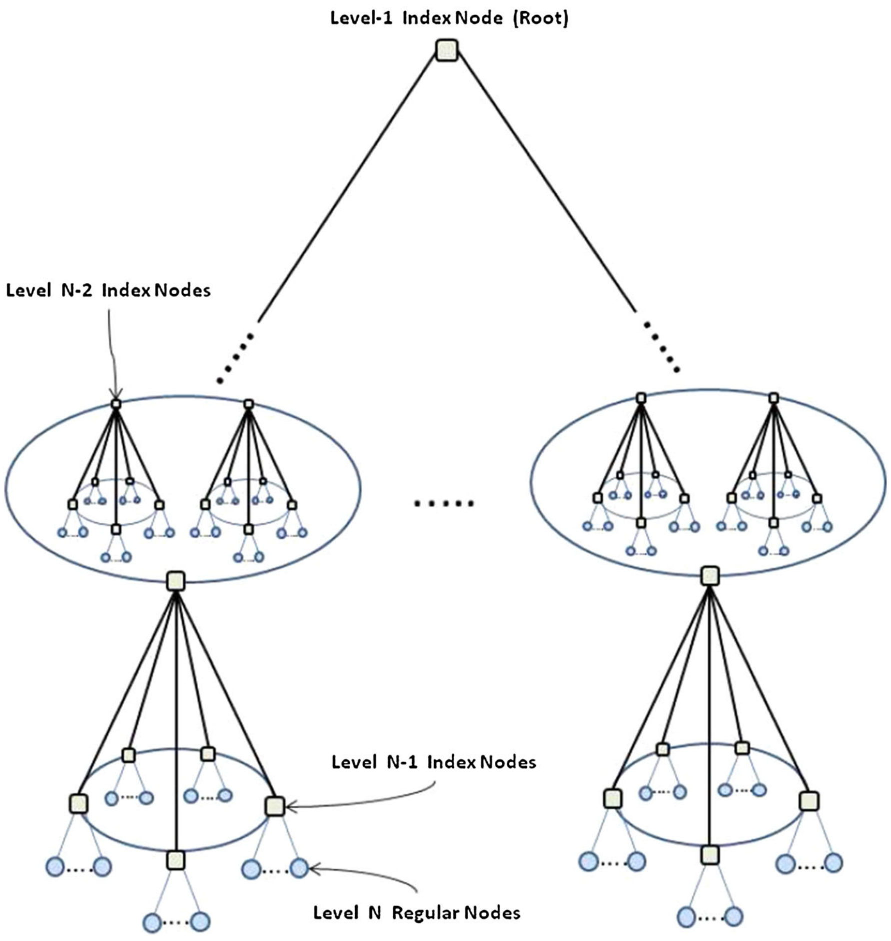


Fig. 1 The structure of threshold-based hierarchical resource discovery

looks its local indices for the required resource. If the resource is found, the indexing node returns resource owner address to the requester node and the discovery process will be terminated. Otherwise the indexing node passes the query to its right hand side sibling node. This process continues until the query is returned back to the first indexing node on the current level.

Once the query is returned back to the first indexing node on the current level, it forwards the query to its parent indexing node at upper level of the hierarchy. The parent indexing node does not check this query in other children indexing nodes repeatedly and immediately attempts to forward the query to its parent indexing node by the above mentioned mechanism. This process is repeated on each level until the required resource is found or the query reached to the root node. Once the root node receives a resource query, if the query was investigated in lower level sibling nodes, the search process will be finished with the response lack of the resource. Otherwise, the root node checks its children nodes except the sender node and if the resource owner was found, its address will be sent to the resource requester.

Each indexing node in the tree has four branches and the last level indexing nodes consist the regular nodes. The indexing tree used in this technique keeps the resource information in five levels and the number of existing regular nodes specifies the capacity of the indexing nodes on the last level. In each level of the tree, an indexing node has $m_l = \frac{n}{b^{(l-1)}}$ regular nodes indexed on its descendant nodes, which n is the total number of Grid nodes, b is the number of branches at each level, and l is the level of indexing node. Also, considering the last level of the tree as L, the number of indexing nodes existing in the lower levels of a node is $I_l = \sum_{i=1}^{L-l} b^i$.

The resources join and leave the network and these events cause to transfer more messages. Moreover, after finding a required resource, the found resource is allocated to the requester node. The resource information changes caused by the allocation events also should be considered. The update messages that are related to the allocation events are estimated by the Eq. 1 such that s is the success rate of the queries, r is the number of issued resource requests, and k is the rate of the multi-attribute query issues that request two resources.

$$n_1 = 2(k+1) \times s \times r \tag{1}$$

$$n_2 = h_1 \times n \tag{2}$$

The messages related to the join and leave events are estimated by the Eq. 2. Assuming h_1 as the rate of the join and leave events and n as the number of Grid nodes, this equation specifies the messages that are issued for the regular nodes' join and leave events. In addition to the above mentioned update messages, some messages are transferred for the random change events that are applied by the system on the resources. Each Grid node shares one to three resources. The Eq. 3 calculates these messages as well the allocation and join/leave events such that R represents the maximum number of resources in each Grid node and h_2 is the rate of change events on the resources.

$$m_c = n_1 + n_2 + \sum_{i=1}^{R} (h_2 \times n \times i) \tag{3}$$

Moreover, a requested resource can be found by the probability of $\frac{l}{n}$ in each Grid node of the last level regular nodes. Thus the probability of finding a resource at a level l of the tree can be estimated by the Eq. 4. Also, considering the number of issued resource requests as r, and k as the rate of the multi-attribute queries, the number of resource requests reached to an indexing node located at level l of the tree is estimated by the Eq. 5.

$$p_l = m_l \times \frac{l}{n} \tag{4}$$

$$m_r = \sum_{i=L}^{l} \left(2(k+1) \times r \times (1 - p_i)\right) \tag{5}$$

$$m = \sum_{i=L}^{l} \left(\left(\frac{m_c + m_r}{b^{(i-1)}} - T\right) \times \frac{b-1}{b^{(i-1)}}\right) \tag{6}$$

Considering the update and request messages, sum of the messages received by each indexing node at level l can be estimated. With regard to investigating a received query in the sibling indexing nodes except the node that the query is received from it, and the number of indexing nodes at each level that is $b^{(l-1)}$, the overall messages received by each indexing node is estimated by the Eq. 6 such that T is the specified threshold value of indexing nodes and b is the number of branches in the indexing nodes.

As it is shown in the Eq. 6, the threshold value impacts on the number of processed messages at each stage of different branches. After calculating all received join/leave and request messages to an indexing node of each level by Eqs. 3 and 5, the value of threshold is subtracted from the above received messages. It is because after reaching the threshold value, the received queries should be processed in the sibling nodes and if the requested resource could not be found in the circular sibling nodes investigations, then the request will be processed by the upper level indexing node. So, for each indexing node in different levels of the tree, threshold value is subtracted from all the received join/leave and request messages. With regard to the number of issued requests in our experiments, that is 1,000 requests for each experiment, the threshold values of 50, 100, and 150 are applied on the indexing nodes that are between 5 and 15 % of issued requests.

Choosing higher values for the threshold causes the proposed technique acting more similar to the typical hierarchical structure and reduces the impacts of threshold on the processing messages at each indexing node. But for more number of issued resource requests, the threshold may be chosen as higher values relatively. The number of Grid nodes impacts on the number of children nodes that can issue the resource requests. Thus, a Grid environment with more number of Grid nodes can be affected more by the in-process message threshold values.

3.2 Implemented modules of THRD

The implementation infrastructure for THRD consists of five modules, i.e. Request_Manager, Broker, Resource_Handler, Grid Information System (GIS), and Report_Writer. Request_Manager generates the events of the resource requests and nodes' join/leave. The generated resource requests are assigned to users randomly and sent to the module Broker. The module Broker handles and schedules the resource requests and responses. It acts as a medium between the resource discovery system and Grid environment. It analyzes received resource requests according to the receive time, separates multiple-attribute requests to make different independent queries, schedules all queries and forwards them in turn to GIS. Then it waits for the resource owner address to be returned from GIS. When the matched resource owner address is discovered by GIS and returned, Broker sends the allocation request to the found resource owner.

The indexing nodes and resource information are handled by GIS. Almost all modules are related to GIS to transfer the resource information or get the search results. When a resource is allocated to an application, its value will be updated and the new resource information is sent to GIS. Resource_Handler is responsible to send any resource information changes caused by resource allocation, join, and leave events to GIS and subsequently GIS updates all indices related to the changed resource information. Resource_Handler also manages the allocation requests sent by Broker. If a resource is available, then it will be allocated to the requesting application for specified period of time which is determined by the initial resource requester. GIS is responsible to index resource information and locate the resource owners according to the received requests. After finding the resource owners, GIS returns their address back to Broker.

Report_Writer receives different statistical information related to nodes requesting resources, transferred messages, and successful results of search processes to analyze and make the appropriate reports according to the comparison and analysis requirements.

Figure 2 presents the algorithm of the join/leave events for Grid resources. Once Request_Manager issues a join event, the process of *JoinRequest()* will be performed. This process chooses one of the free nodes to join the Grid environment. With regard to the random event issues of the join and leave events, usually there are some free nodes available to be chosen. After choosing a free node, its status will be changed to *'active'* and a suitable indexing node is found in order to index the new Grid node.

For the leave event, the process of *LeaveRequest()* first chooses a random node to be left. The leave event can be issued for the Grid node or a resource of the Grid node. It is specified by Request_Manager through the parameter *'type'* that is set to *'complete_leave'* if the Grid node should be left. Because each Grid node can have one to three resources, so a random number between one to three is chosen and then the resource will be removed from the chosen Grid node. If Request_Manager decided to leave an indexing node, it performs the process *LeaveRequestHNode()*. In this process, first a random indexing node is chosen to leave the Grid. Because the indexing nodes have some information about other Grid nodes, so the process substitutes a free indexing node for the leaving node. For this purpose, the process sets the substitute indexing node as the new parent for all the children of leaving node. Then the leaving node is added to the list of free indexing nodes.

Figure 3 presents the algorithm of the resource request issue process in THRD which is handled by method *RequestIssue()*. It first chooses a random Grid node to receive the request message. Because this request will be assigned to a regular node located in the last level of the indexing tree, the flag *"FromChild"* is set as *"true"*. This flag will be used in the search process to check that a request is received from a sibling node in current level or a child.

The query message can be issued to request a single or multiple resources. It is considered that a query can request one to three resources. The number of resources to be requested is specified as random by the variable *"NumResReq"*. Then the random resources are chosen and added to the resource request. After that, the resource request is assigned to a random Grid node and then the request is sent to the parent of the chosen Grid node.

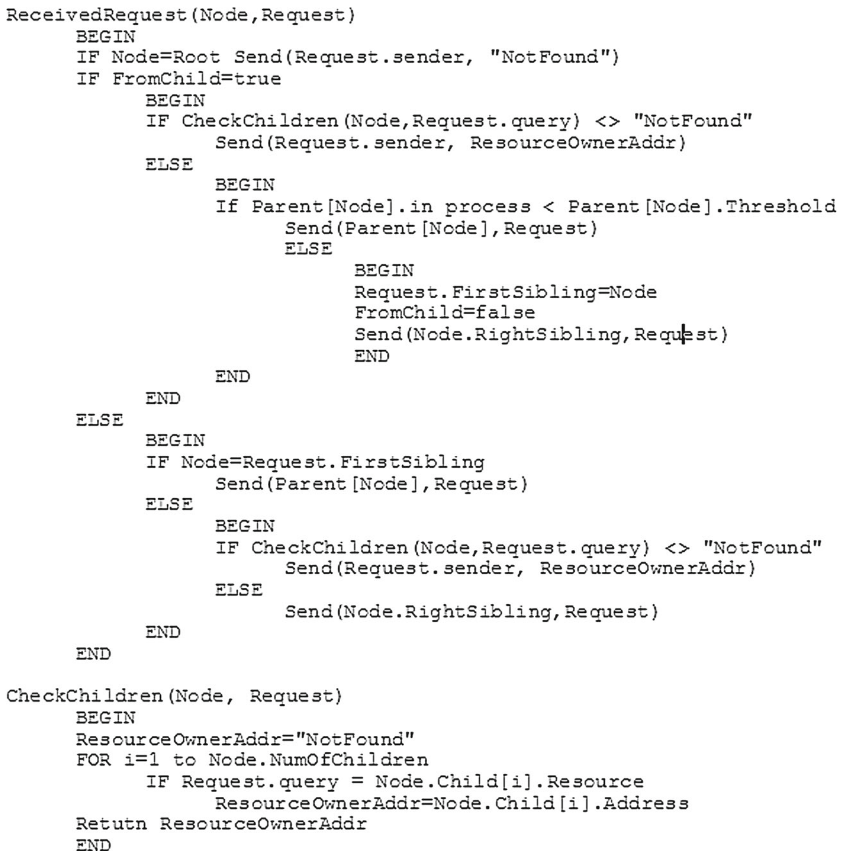
```
JoinRequest()
  BEGIN
        IF FreeNodes.isNotEmpty
              BEGIN
              NewNode = FreeNodes.fetch
              Node[NewNode].isActive = true
              Resources.add(Node[NewNode].resource)
              FOR i = 0 to activatedHs.size()
                    IF Activated HNodes[i].isNotFull
                          BEGIN
                          Node[NewNode].parent= Activated HNodes[i]
                          Break
                          END
              END
        END
LeaveRequest(type)
        BEGIN
        RandNode = Random(NumOfActivatedNodes)
        NumOfResPropsInNode = Node[RandNode].ResProps.length
        IF type="complete leave"
              BEGIN
              Node[RandNode].Remove
              FreeNodes.add(RandNode)
              END
        ELSE
              BEGIN
              RandResToLeave = Random( NumOfResPropsInNode )
              Node[RandNode].Res[RandResToLeave].remove
              END
        END
LeaveRequestHNode()
        BEGIN
        RandHNode = Random(NumOfActivatedHNodes)
        SubstituteHnode = FreeHNodes.fetch
        FOR i=1 to Activated HNodes[RandHNode].children.length
          Activated HNodes[RandHNode].children[i].parent=SubstituteHnode
        Activated HNodes[RandHNode].Remove
        FreeHNodes.add(RandHNode)
        END
```

Fig. 2 The algorithm of Grid nodes' join/leave events in THRD

```
RequestIssue()
      BEGIN
      RandNode = Random(NumOfActivatedNodes)
      FromChild=true
      MultiRes=3
            NumResReq=Random(MultiRes)
            For i=1 to NumResReq
                  BEGIN
                  RandRes=Random(Resources.length)
                  ResReq = Resources[RandRes]
                  Node[RandNode].Request.query.add(ResReq)
                  END
      Send(Parent[RandNode], Node[RandNode].Request)
      END
```

Fig. 3 The algorithm of resource request issue in THRD

When a node receives the resource request, it performs the search through the method *ReceivedRequest()* that is shown in Fig. 4. If the receiving node is the root, it means that all the descendants are investigated and the required resource could not be found. Thus the search process will be finished by sending *"NotFound"* message to



```
ReceivedRequest(Node,Request)
        BEGIN
        IF Node=Root Send(Request.sender, "NotFound")
        IF FromChild=true
                BEGIN
                IF CheckChildren(Node,Request.query) <> "NotFound"
                        Send(Request.sender, ResourceOwnerAddr)
                ELSE
                        BEGIN
                        If Parent[Node].in process < Parent[Node].Threshold
                                Send(Parent[Node],Request)
                                ELSE
                                        BEGIN
                                        Request.FirstSibling=Node
                                        FromChild=false
                                        Send(Node.RightSibling,Request)
                                        END
                        END
                END
        ELSE
                BEGIN
                IF Node=Request.FirstSibling
                        Send(Parent[Node],Request)
                ELSE
                        BEGIN
                        IF CheckChildren(Node,Request.query) <> "NotFound"
                                Send(Request.sender, ResourceOwnerAddr)
                        ELSE
                                Send(Node.RightSibling,Request)
                END
        END

CheckChildren(Node, Request)
        BEGIN
        ResourceOwnerAddr="NotFound"
        FOR i=1 to Node.NumOfChildren
                IF Request.query = Node.Child[i].Resource
                        ResourceOwnerAddr=Node.Child[i].Address
        Retutn ResourceOwnerAddr
        END
```

Fig. 4 The algorithm of resource discovery in THRD

the request sender. Otherwise the method *CheckChildren()* investigates the children nodes for the required resource, and if the search was successful, the found resource owner address is sent to the requester node.

If the requested resource could not be found in the children nodes, the node checks the status of in-process messages in its parent indexing node. If the value of in-process messages in the parent node is less than the threshold value, the request will be passed to the parent. Otherwise the request is forwarded to the right hand sibling node to search the required resource. Each receiving sibling node performs the above process once receiving the request. If the receiving node is the first sibling node that initiated the request circulation, it passes the request to its parent. This process continues until reaching to the root node or finding the required resource.

With regard to the algorithms of request issue, join/leave events, resource discovery and mathematical arguments presented in previous subsection, the order of different algorithms are extracted. The resource request issues depend only to the number of resources exist in the environment and considering the presented issue process, it has the order of O(R), which R is the number of existing resources in the established

Grid environment. For the join/leave events, the related algorithm needs $\sum_{i=L}^{l}(\frac{n}{b^l})$ iterations that indicates the order of O(n), and n refers to the number of Grid nodes, b is the number of branches in each level, l and L refer to the processing level and last level of the hierarchical tree respectively. Also, for the resource discovery process, in the best case, it has the order of $\log_b^n$ iterations. But in the worst case, it has the complexity of O(n).

4 Simulation results and evaluation of THRD

In order to perform the simulation experiments for the proposed threshold-based technique, different processes of join/leave events, random resource request issues, and the search method are implemented in Java. The proposed technique is simulated by GridSim 5.2 [30]. It is a well-known java-based simulation tool for Grid environments which includes extensible information system and is able to model the interaction of users, resource brokers, resources and the network.

The experiments are performed by different number of Grid nodes i.e. 1,000, 5,000, 10,000, and 20,000. Based on the literature [31], the scale of a Grid environment with more than 5,000 nodes is considered as large. Thus, the ranges between 1,000 and 20,000 of nodes are chosen in the experiments in order to cover the small to large scale Grid networks. Also as mentioned in the last section, three threshold values i.e. 50, 100, and 150 are considered for in-process messages to investigate its impact on behaviour of the system in terms of the message load and response time. The nodes to join and leave the Grid environment, and resource properties to be requested and the requester nodes are chosen as random using the normal distribution. Moreover, the confidence level of the simulation is 95 %. Furthermore, 1,000 resource requests are issued for each experiment. The reported values of the response time and message load are the average value resulted from all 1,000 resource requests. The experiments are performed under the hardware platform including CPU Intel Core 2 Duo 2.93 GHZ and 4 GB of RAM.

Moreover, to evaluate the performance of THRD in terms of the response time and message load, its experimental results are compared with two existing hierarchical resource discovery techniques, [9] and [28] which are referred as HR1 and HR2 respectively in the rest of this paper. The algorithm of HR1, HR2 and THRD are simulated in the same hardware and software platform. Also, to have the same comparison conditions for them, the hierarchical tree for all of them has five levels and each indexing node has four children. In addition, to obtain the reliable results, 1,000 resource requests are issued per experiment and the average values of the message load and response time are calculated. The message load refers to the number of messages transferred per second during the experiment, and the response time indicates the time between issuing the resource request and returning the found resource owner address to the requester node.

Table 1 presents the message load of THRD for different number of Grid nodes and threshold values. The results indicate that the upper level indexing nodes have higher amounts of the message load. Also, increasing the threshold value causes THRD to have the higher amounts of the message loads. It is because in the cases with high

Table 1 The message load (msg/s) of THRD for different number of Grid nodes and threshold values

Threshold/level	Number of grid nodes			
	1,000	5,000	10,000	20,000
Thresholds = 50				
1	12.7	37.9	73.4	138.1
2	5.5	12.6	16.3	26.97
3	2.7	5.9	11.4	19.8
4	1.5	3.4	6.3	11.7
5	0.58	2.27	3.36	6.4
Thresholds = 100				
1	13.1	41.3	76.6	141.9
2	5.74	13.2	18.57	34.6
3	2.75	6.1	11.88	20.66
4	1.54	3.47	6.8	12.01
5	0.59	2.27	3.37	6.42
Thresholds = 150				
1	14.01	44.23	84.61	158.2
2	5.78	13.79	20.58	38.01
3	2.82	6.42	12.02	21.35
4	1.57	3.51	6.91	12.41
5	0.6	2.28	3.38	6.47

amount of threshold, the indexing nodes allow more number of messages being passed to the parent nodes which it subsequently increases the message load of upper level indexing nodes in the hierarchy. But it cannot impact significantly on the lower level indexing nodes since their in-process messages are less probable to reach the threshold limitation.

Figures 5 and 6 present the message load of all techniques for 20,000 Grid nodes in high dynamic and low dynamic environments respectively. The resource modification messages and resource requests received from descendant Grid nodes cause that whatever the indexing nodes are located at the upper levels, they deal with more amounts of message loads and the top most indexing nodes are prone to be the points of the bottleneck problem. The environment is in the low dynamic condition when less than 5 % of Grid nodes deal with join and leave events. As it is expected, HR2 acts better than HR1 and our proposed technique in low dynamic conditions. It is because of cache used in HR2 that makes it more efficient when the environment is lowly dynamic. But relative to THRD, in high dynamic conditions, HR2 deals with more message overhead to update the information of different levels and their organization.

With growing the number of nodes, the domain of search and resource information updates will be increased. It also increases the rate of resource allocations because more number of nodes increases the probability of resource request matches. The results indicate that THRD, HR1 and HR2 act almost similar in lower levels of the

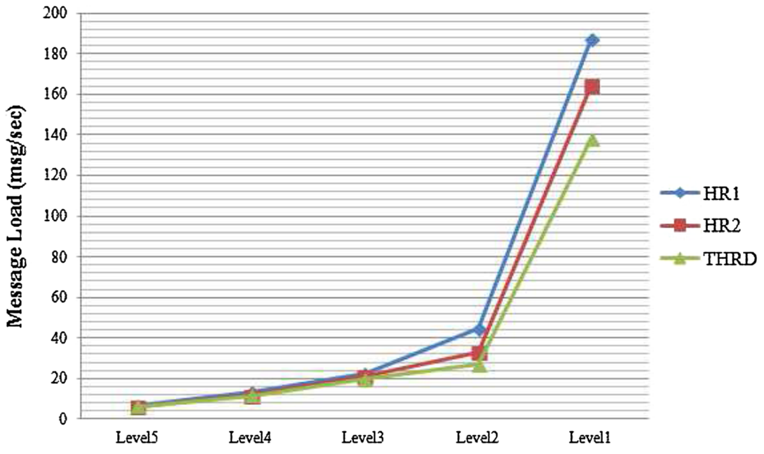


Fig. 5 The message load (msg/s) of THRD, HR1, and HR2 for 20,000 Grid nodes in high dynamic conditions

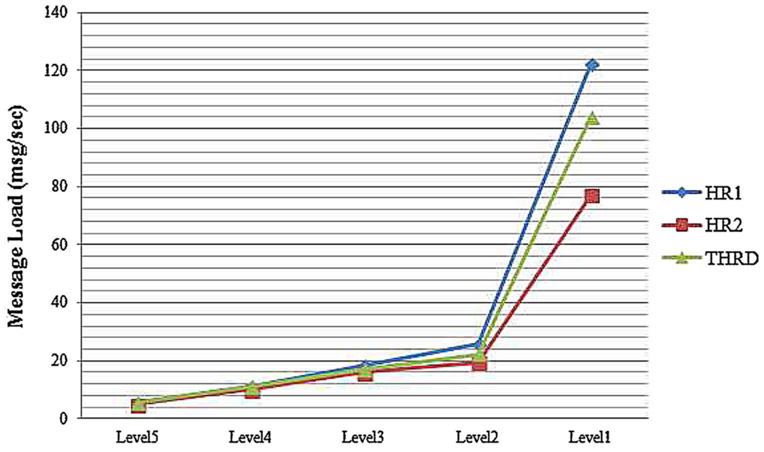


Fig. 6 The message load (msg/s) of THRD, HR1, and HR2 for 20,000 Grid nodes in low dynamic conditions

hierarchy since on these levels, the message load amounts are low and they are less likely to reach the threshold value. But relatively, HR deals with more message loads at upper levels as compared to THRD. It is because in THRD, some requests are not allowed to be passed to upper levels and they are satisfied by the sibling nodes. Also, HR2 suffers from high message overheads to handle different levels of organization. It acts even worth in the highly dynamic conditions due to lack of cache usage.

Table 2 shows the reduction rates of message load in the root indexing node for different number of Grid nodes and threshold values.

The indexing nodes located at lower levels of the tree transfer lower number of messages because more number of indexing nodes is existed in such levels. For example, level 2 of the tree consists of 4 indexing nodes however the forth level of the tree

Table 2 The message load reduction rates of THRD for different threshold values

Number of nodes	1,000 (%)	5,000 (%)	10,000 (%)	20,000 (%)	Overall reduction rate
Threshold = 50	21	25	26	26.2	24.5
Threshold = 100	18.5	19	23	24.3	21.2
Threshold = 150	13	13.9	15	15.5	14.3

Table 3 The average response time (s) of THRD for different threshold values compared to HR1 and HR2

	Number of Grid nodes			
	1,000	5,000	10,000	20,000
HR1	0.31	0.45	0.53	0.6
HR2	0.35	0.53	0.67	0.88
Thresholds = 150	0.34	0.50	0.60	0.71
Threshold = 100	0.36	0.53	0.63	0.75
Threshold = 50	0.37	0.55	0.65	0.82

consists of 64 indexing nodes. This causes that each indexing node of level 4 transfers less number of messages compared to the second level. But the threshold value is the same for all the indexing nodes. Thus, the lower level indexing nodes less probably reach to the threshold value of in-process queries compared to the upper level ones. This means that upper level indexing nodes allow less rates of the receiving messages to be passed to the parents. Also, Table 2 indicates that the message load is reduced 24.5 and 14.3 % for the threshold values equal to 50 and 150 respectively. It is because for higher values of the threshold, less messages are required to be investigated in sibling nodes and they will be passed to the parent. This subsequently causes to have lesser rates of the message load reductions in the cases with higher amounts of the threshold. The proposed technique also impacts on the response time of the resource discovery.

Table 3 and corresponding Fig. 7 present the average response time of THRD for different threshold values and different number of Grid nodes compared to HR1 and HR2. Also, the rates of response time increases for different values of the threshold are shown in Table 4. Although the message load of upper level nodes could be reduced by applying the threshold for in-process messages in the proposed technique, the results indicate that it increases the response time. With regard to delays of the queries during the forwarding process caused by the threshold limitations, the response time of the issued queries is increased. HR2 acts even worth than THRD because it deals highly with organizational relationship updates in dynamic conditions. Moreover, HR2 acts as the best case among the experimented techniques in the low dynamic conditions. It is because of cache beneficiary in low dynamic environment that decreases the process time of queries. However HR1 and THRD are not affected significantly by the percent of join and leave events for Grid nodes.

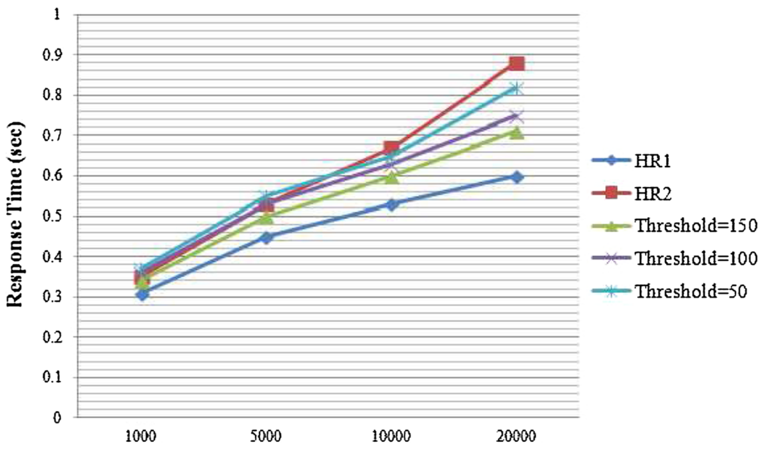


Fig. 7 The average response time (s) of THRD, HR1 and HR2

Table 4 The rates of response time increases for different threshold values

Threshold values	Number of Grid nodes				Overall increase rate (%)
	1,000 (%)	5,000 (%)	10,000 (%)	20,000 (%)	
Threshold = 150	9.5	11	13.2	18.3	13
Threshold = 100	16	17.7	18.8	25	19.3
Threshold = 50	19	22	22.6	36	24.9

The response time increasing rates shown in Table 4 indicate that higher values of the threshold cause the queries being answered faster. The reason is that the indexing nodes with higher values of threshold allow more number of queries being passed to upper levels without waiting. Also, when a query is passed to the parent, it is investigated as parallel in all children. But when a query is not allowed to be passed to the parent, it will be investigated in the sibling nodes as serial that increases the response time. This indicates that increasing the threshold value of indexing nodes results to less rates of message load reductions however it accelerates the search process.

With regard to the above results, the message load reduction and response time increase rates shown in Tables 3 and 4 indicate that when the threshold is set to 150, the message load is reduced 14.3 % and the response time is increased 13 %. These rates are 21.2 and 19.3 % for message load reduction and response time increasing rates respectively when the threshold is set to 100. Also, in the cases with lower values of the threshold such as 50, the response time is increased with the higher rates relative to the message load reduction.

With regard to the Eq. 6, after calculating all join/leave and request messages received to an indexing node of each level, the value of threshold is subtracted from these messages. It is because after reaching the threshold value, the received queries are processed in the sibling nodes. So, the threshold value is subtracted from the received join/leave and request messages in each indexing node. The threshold values were experimented between 5 and 15 % of the issued resource requests, and the results showed that by choosing higher values of the threshold, the behaviour of the proposed technique would be more similar to the typical hierarchical approach. The reason is that by choosing higher values of the threshold, more number of messages can be passed to the upper level indexing nodes as typical approach. Also, the number of Grid nodes impacts on the number of children nodes that can issue the resource requests and as it is shown in the results, the Grid environments with more number of Grid nodes are affected more by the in-process message threshold values. Therefore, the usefulness of THRD depends on the application requirements so that if the message load is more important than quickness of the search process, THRD can be used as an appropriate solution in the hierarchical structure.

5 Conclusion

This paper proposed a technique to reduce the message load of upper level indexing nodes in the hierarchical resource discovery approach. For this purpose, it applied a threshold value on the indexing nodes for the number of in-process messages. The indexing nodes do not accept messages more than the threshold value to process. In such cases, children nodes forward the requests to their sibling nodes instead of their parent. By this manner, some requests are satisfied by the sibling nodes without needs of passing to the upper level indexing nodes.

The proposed technique is simulated in GridSim and experimented with different number of Grid nodes i.e. 1,000, 5,000, 10,000, and 20,000. Also, the impact of different threshold values i.e. 50, 100, and 150 for in-process messages is investigated. The results of the performed experiments indicated that the message load of indexing nodes is reduced however the proposed technique affects negatively the average response time. It is because the queries are delayed during the investigation when they are not allowed to be passed to the parent nodes. In such cases the queries are forwarded to the sibling nodes and investigated as serial. Such delays cause increasing of the average response time for the issued queries.

As a result, applying the threshold for in-process messages of indexing nodes to limit the receiving queries can be useful to reduce the message load of upper level indexing nodes in the hierarchical structure. But the negative impacts of such manner on the response time should be considered. The usefulness of the proposed technique depends on the application requirements so that if the message load reduction is more important than quickness of the search process, this technique is an appropriate solution in the hierarchical structures.

References

1. Poshtkohi A, Abutalebi AH, Hessabi S (2007) Dotgrid: a.net-based cross-platform software for desktop grids. Int J Web Grid Serv 3(3):313–332
2. Malik S, Nazir B, Qureshi K, Khan I (2013) A reliable checkpoint storage strategy for grid. Computing 95(7):611–632

3. Hasanzadeh M, Meybodi M (2013) Grid resource discovery based on distributed learning automata. Computing 96(9):909–922
4. Reinhard V, Tomasik J (2008) A centralised control mechanism for network resource allocation in grid applications. Int J Web Grid Serv 4(4):461–475
5. Hameurlain A, Cokuslu D, Erciyes K (2010) Resource discovery in grid systems: a survey. Int J Metadata Semant Ontol 5(3):251–263
6. Kaur D, Sengupta J (2007) Resource discovery in web-services based grids. Int J World Acad Sci Eng Technol 31(1):284–288
7. Moltó G, Hernández V, Alonso J (2008) A service-oriented wsrf-based architecture for metascheduling on computational grids. Int J Future Gener Comput Syst 24(4):317–328
8. Shaikh A, Alhashmi S, Parthiban R (2011) A semantic-based centralized resource discovery model for grid computing. Int J Grid Distrib Comput 26(1):161–170
9. Huedo E, Montero R, Llorente I (2009) A recursive architecture for hierarchical grid resource management. Int J Future Gener Comput Syst 25(4):401–405
10. Zamanifar K, Abootalebian H, Malazizi L (2012) Dynamic ttl based algorithm for hierarchical resource discovery model in grid. Int J Theor Appl Inf Technol 42(1):18–25
11. Palmieri, F (2010) Percolation-based replica discovery in peer-to-peer grid infrastructures. In: Proceedings of networks for grid applications, Springer, Berlin, pp 45–56
12. Torkestani AJ (2012) A distributed resource discovery algorithm for p2p grids. Int J Netw Comput Appl 35(6):2028–2036
13. Noghabi H, Ismail A, Ahmed A, Khodaei M (2012) Optimized query forwarding for resource discovery in unstructured peer-to-peer grids. Cybern Syst 43(8):687–703
14. Talia D, Trunfio P, Zeng J (2007) Peer-to-peer models for resource discovery in large-scale grids: a scalable architecture. In: Proceedings of high performance computing for computational science, Springer, Berlin, pp 66–78
15. Li M, Qi M (2009) Facilitating resource discovery in grid environments with peer-to-peer structured tuple spaces. Int J Peer-to-peer Netw Appl 2(4):283–297
16. Ali HA, Salem MM, Hamza AA (2012) A framework for scalable autonomous p2p resource discovery for the grid implementation. Int J Comput Syst Sci Eng 27(4):275–284
17. Mokadem R, Hameurlain A, Tjoa AM (2010) Resource discovery service while minimizing maintenance overhead in hierarchical dht systems. In: Proceedings of 12th international conference on information integration and web-based applications and services, Paris, France, pp 630–638
18. Padmanabhan A, Ghosh S, Wang S (2010) A self-organized grouping framework for efficient grid resource discovery. Int J Grid Comput 8(3):365–389
19. Salter J, Antonopoulos N (2007) An optimized two-tier p2p architecture for contextualized keyword searches. Int J Future Gener Comput Syst 23(2):241–251
20. Mastroianni C, Talia D, Verta O (2008) Designing an information system for grids: comparing hierarchical, decentralized p2p and super-peer models. Int J Parallel Comput 34(10):593–611
21. Javanmardi S, Shariatmadari S, Mosleh M (2013) A novel decentralized fuzzy based approach for grid resource discovery. Int J Innov Comput 1(1):23–32
22. Trunfio P, Talia D, Papadakis H, Fragopoulou P, Mordacchini M, Pennanen M, Popov K, Vlassov V, Haridi S (2007) Peer-to-peer resource discovery in grids: models and systems. Int J Future Gener Comput Syst 23(7):864–878
23. Yu J, Zhao C, Pan Y (2006) Grid resource management based on mobile agent. In: Proceedings of international conference on computational intelligence for modeling, control and automation, Sydney, Australia pp 255–255
24. Muthuchelvi P, Mala G, Ramachandran V (2009) Agent based grid resource discovery with negotiated alternate solution and non-functional requirement preferences. Int J Comput Sci 5(3):191–198
25. Kang J, Sim KM (2012) A multiagent brokering protocol for supporting grid resource discovery. Int J Appl Intell 37(4):527–542
26. Elmroth E, Tordsson J (2005) An interoperable, standards-based grid resource broker and job submission service. In: Proceedings of first international conference on e-science and grid computing, Melbourne, Australia, pp 212–220
27. Ma Y, Gong B, Zou L (2008) Resource discovery algorithm based on small-world cluster in hierarchical grid computing environment. In: Proceedings of seventh international conference on grid and cooperative computing, Shenzhen, China, pp 110–116

28. Chang RS, Hu MS (2010) A resource discovery tree using bitmap for grids. Int J Future Gener Comput Syst 26(1):29–37
29. Puppin D, Moncelli S, Baraglia R, Tonellotto N, Silvestri F (2005) A grid information service based on peer-to-peer. In: Proceedings of 11th international Euro-Par conference on parallel processing, Berlin, pp 454–464
30. Buyya R, Murshed M (2002) Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Int J Concurr Comput Pract Exp 14(13–15):1175–1220
31. Mastroianni C, Talia D, Verta O (2007) Evaluating resource discovery protocols for hierarchical and super-peer grid information systems. In: Proceedings of 15th EUROMICRO international conference on parallel, distributed and network-based processing, Naples, Italy, pp 147–154