Scientia Iranica, Vol. 14, No. 6, pp 612–624
© Sharif University of Technology, December 2007

SCIENTIA
IRANICA

A New Approach to Resource Discovery and Dissemination for Pervasive Computing Environments Based on Mobile Agents

E. Bagheri* and M. Naghibzadeh[1]

Pervasive computing, as a new branch in the field of distributed computing, has received wide contribution from different researchers. In this novel computing model, a vast range of computational and communication resources, along with other types of service, are gathered under a single system image based on certain predefined criteria. To create a transparent environment and provide end-users with the illusion of the local availability of multiple resources, some kind of manager is needed to coordinate the tasks and their required resources. The resource management system is mainly responsible for a balanced distribution of available resources among different tasks. Devising efficient resource discovery and dissemination algorithms is, hence, an important step towards preparing the bases for a resource centric management package. In this article, the aim is to provide two algorithms for this problem, using mobile agents. The proposed resource discovery algorithms use two different hierarchical and flat approaches. The simulations show a good performance for both of the proposed models; however, the hierarchical algorithm shows better results, based on some of the introduced factors.

INTRODUCTION

The growth of computer network protocols and the creation of a suitable infrastructure for international communication on the one hand and the development of great scientific research on the other hand, have increasingly created the need for scientific and equipment based synergy. Specialization, as the greatest concept in the scientific arena of our century, forces data transfer and group work to be used for solving enormous problems. For this reason, scientists, researchers and even industry-based teams have formed groups with members of different capabilities, who may not necessarily be located geographically close to each other but cooperate in problem solving activities through the collaborative environments created using new technologies [1]. The need for knowledge sharing and research cooperation is only one of the aspects of complex scientific development. Today's computation intensive problems need rich resources, which may not be in the reach of every researcher. The first solution is to use supercomputers to provide us with high computational and processing power. Although these computers enable us to solve complex problems and access required resources, they have a great amount of maintenance costs, which are, in most cases, infeasible. To solve this problem, many supercomputers allow remote access to their resources. This approach could, to some extent, cover the prior mentioned problem, but, would create new shortcomings, such as forming a bottleneck at high traffic times. Devising a single method to guarantee QoS (Quality of Service) in this model is also very hard [2].

The approach taken in the last few years has been to share available resources among interested parties. In this way, resources could be shared, based on different criteria and, hence, solve the resource shortage problem. This concept is very much beyond the peer to peer resource sharing models, where only one type of resource, such as files, is being shared [3]. These collaborative resource sharing environments are called pervasive environments. One of the well-known models of such an environment is the Grid [4], which can be formed based on different needs and can behave according to defined requirements.

*. *Corresponding Author, Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada.*
1. *Department of Computing, Ferdowsi University of Mashhad, Mashhad, I.R. Iran.*

Plurality and the variety of resources available in the grid, require an integrated management system to provide users with appropriate services. Having heterogeneous resources in the environment impedes resource access and, hence, demands well defined and standard interfaces to query resources. These interfaces will pave the way for creating a transparent resource access environment. In this environment, the tasks are only run by the user and the management system will transparently schedule and allocate suitable resources without user intervention [4]. It is obvious that the models used in the pervasive computing environment must, to a great extent, have built-in intelligence and the ability to learn, in order to show a better overall performance. The two mentioned characteristics, social behavior and intelligence, can be smartly implemented using software agents. Agents can be used in two different ways. In the first approach, agents are used to create configurable and extendable software from the software engineering aspects, while, in the latter approach, they are autonomous entities, which help one to solve the problem.

In this article, initially, the problem on hand will be introduced and the opportunities and threats that will be encountered will be analyzed. The following parts will, then, review some of the existing models and the authors' algorithms, for resource discovery and dissemination, will also be completely explained. The simulation tools and environment will, then, be introduced and the simulation results will be shown. The article will then examine the obtained results and the simulation correctness.

PROBLEM DEFINITION

Pervasive environments are mainly used to share a vast range of different resources among their users. Each user, which can be an organization, a university or even a single person, can join the environment, based on an agreement, share its resources or use the shared resources [5].

Resource access conditions are agreements between resource providers and users, which are called Service License Agreements (SLA). After joining the environment, every single user should become aware of all available resources or decide on the best suitable resource available for its running task. Therefore, the resource management systems, which handle all resource management tasks, are of great importance [6]. The resource management system is in charge of resource scheduling, allocation and redemption, discovery and dissemination, resource access control and many other resource related functions [7]. Different resource management packages have been prepared. Two main approaches have been taken up for resource discovery and dissemination. In the first approach, both algorithms are thought of as a single and are packaged inside the same module while, in the second approach, the two algorithms are separately devised and are only related through the resource information store organization. What is obvious is that integrating the two algorithms inside one module will enhance overall performance but, on the other hand, will degrade system modularity.

A resource is only recognized by its host machine once it has newly joined the environment. If the resource owner is willing to share the resource in the environment among different users, he/she should advertise the resource availability through different mechanisms. The mechanism used to introduce newly joined resources to the environment is called resource dissemination. Various algorithms have been proposed for resource dissemination, which show dissimilar performances being compared, based on different indicators, such as the number of local and global messages passed in the advertisement process. Choosing an unsuitable resource dissemination algorithm may cause severe network traffic inside the computing environment. Selecting the appropriate attributes of a resource for advertisement is also very important [8].

On the other hand, a task being submitted to the environment through the specified portals for execution must identify its resource needs. In this step, the resource management system should search for suitable resources, conforming to the constraints introduced by the task, and allocate the matching resources immediately to the task or reserve the resource if it is occupied. The process of searching for an appropriate resource, which can be done in both a central or distributed manner, is called resource discovery [8]. Various resource discovery models, based on information store organizations, have been implemented. If all resource information of an environment is gathered in a single resource information store, resource discovery can be simply done through enquiring this database. However, the information and status of the available resources are most of the time, scattered around the environment in different repositories in a decentralized manner. Although a decentralized resource information store complicates the functionality of the resource discovery algorithm, it avoids bottlenecks.

Resource discovery and dissemination also face some other difficulties. As the shared resources inside the pervasive environment are not only moderated by the resource management system and are also controlled by their own owners, accessing much vital information is, at times, impossible. For this reason, these algorithms should only cope with the information retrieved from resource host interfaces. Available algorithms in resource discovery and dissemination problems have used certain methods, such as queries, agents and pull/push message based models.

AVAILABLE RESOURCE DISCOVERY AND DISSEMINATION MODELS

Resource discovery and dissemination protocols were initially used in local networks to reduce the network management burden. Protocols, such as LDAP and X.500, were utilized as resource information store organizations. The main disadvantage of such protocols is their performance degradation in dynamically changing network topologies [9]. Other models, based on distributed objects, were formed to store resource and service information. CORBA, which is well recognized, resembles such technology. In networks such as Bluetooth, where resources and services rapidly join or leave the network, a dynamic method for network configuration seems to be essential. With this theory in mind, devices with zero configurations (Plug and Play) were produced. Different enterprises and many network device producers released protocols such as HAVi, Jini, SLP and UPnP [10-13].

The difference between the pervasive computing environments, such as the Grid and the traditional networks, can be looked at from different perspectives. On the one hand, only very simple network devices are known as resources in the traditional networks, while any kind of device or service can be named as a resource in the pervasive computing environment and, hence, be shared. On the other hand, there is no heterogeneity, or, it is kept at its minimum in the traditional network by a moderated resource, device and service selection, but, in the pervasive computing environment, lack of central control over available resources will increase heterogeneity. For these reasons, although many different resource discovery and dissemination protocols have been devised for traditional networks, they cannot be used for the authors purpose. However, the models could be well applied to create novel algorithms.

PROBLEM SOLVING APPROACH

The presented approach in devising resource discovery and dissemination algorithms is to create an independent module in such a way that it could be further extended in the future to interact with existing pervasive environment management systems, such as Globus [14], use their base capabilities and build on their functionality, although these features specifically fall into the realm of future work. As resource management functions are conceptually independent, but have functional dependency, and each module cannot re-implement all the required functions, such as resource access control models, resource access models, resource information based organizations and many other packages, the planned module should be able to exchange specific data with the base resource management system that has already implemented many of the required functionalities [15]. For this reason, an attempt has been made to implement both of the algorithms as an independent software package, which only relies on the services provided from the lower layers.

The other aspect that should be carefully considered, while designing the discovery and dissemination algorithms, is domain management. Although shared resources in the environment are, in the large-scale view, moderated by the global management system, each of these resources belong to a specific owner with a more detailed perspective, therefore, resource access models should be devised carefully from the ownership point of view. For this reason, the concept of virtual organizations has been shaped. Uniformly managed domains that provide the environment with a set of resources are named virtual organizations. Virtual organizations can be geographically dispersed but should be moderated in a logically integrated way [4]. Virtual Organizations can either conform to all the set up regulations of the global resource management system or create their own models and interact with global resource management systems through defined interfaces. The machine organization in the latter approach is called the cell model. In the cell models, all the internal complexities of a virtual organization are hidden from an outside viewer and the resources inside the virtual organization are only accessible through defined interfaces. The remaining tasks in the cell model are done similar to the hierarchical model.

One of the chief problems in using a cell machine organization is the bottleneck that may be created at the virtual organization boundaries. As all the internal functionality of a grid has been concealed from the overall environment management, a machine should be introduced as the virtual organization attorney. Referring to the fact that all virtual organization communication with the outside world should be done through this machine, information flow volume will create a bottleneck and degrade the pervasive environment performance [8]. In the authors' model, a rotational attorney selection model has been created where a single machine does not always act as the virtual organization representative, but is selected, based on different criteria.

Software agents can be well suited for pervasive environment management, regarding the fact that social behavior and intelligent decision making is of much importance. In our approach, mobile agents have been used for resource discovery. The advantage of using numerous software agents in the environment is that it increases fault tolerance and helps cooperative multi agent problem solving. Lightweight mobile agents are created when a resource is requested and are sent to different virtual organizations, based on the internal machine knowledge of the pervasive environment status. Dispatched agents then negotiate with the resource (or with the corresponding virtual organization attorney) and gather the required information. The mobile agents will return to their base and, based upon the gathered information, will decide on the suitable resource. Using software agents allows one to create a severe security policy in accessing the resources. Each agent is the delegate of a machine or a virtual organization which have defined access levels, so, entry to a certain machine or virtual organization is only granted if the inherited access rights enable the agent to do so. In the authors' model, agent identity recognition is performed by agent certificate authorities. If a user agent tries to access restricted machines or virtual organizations, it will automatically be destroyed using the agent functional environment.

A quality of service guarantee is also another aspect that should be considered when designing different parts of a resource management module. Many tasks have strict resource requirements, which can only be performed properly if the requirements were allocated at the suitable time. Real time multimedia applications are the best-known sample of these kinds of task. Allocated resources should match the requested quality and time, so, in the discovery process, all of the defined parameters should be considered. Although the provided QoS doesn't have direct relations with the resource discovery and dissemination algorithms and is more based on the resource, network and operating system execution models, the resource selected by these algorithms should have the capability to support and comply with the requested conditions [16].

Resources have two different classes of attribute, which can be categorized into two dynamic and static classes. The resource dissemination algorithm should decide on the resource attributes that it will advertise in its process. Some algorithms publish all of the dynamic and static attributes of a resource [9]. As the dynamic attributes of a resource (like the number of processes queued in the CPU) actively change, this information will quickly become out of date and useless. Three models have been used to solve this problem. In the first model, the resource advertises its attributes on a regular basis or whenever a change has occurred. This model is called the Push model. In the second model, the resource management system requests the resource attributes at defined periods of time, which is called the Pull model. These two models create a lot of system overload in the pervasive environment. In the third model, only static attributes of a resource are initially advertised and dynamic attributes are only queried if the need arises [8]. The latter approach has been used in the resource dissemination algorithm. In the authors model, only the static attributes of the resources are introduced. In the discovery process, the mobile agents are responsible for specifying resource dynamic attributes. Decreasing the number of messages passed in the pervasive environment and exact resource attribute information advertising are one of the algorithm's major benefits.

PROPOSED MODEL

The proposed model, for resource dissemination and discovery, consists of a resource dissemination algorithm, along with two completely different structured resource discovery algorithms. The resource discovery algorithms use hierarchical and flat structures. The hierarchical discovery model searches the source virtual organization for suitable resources and introduces the matching resources. In this approach, resources residing on the source virtual organization are preferred, but if a suitable resource is not found, other virtual organizations are also queried. On the contrary, the flat resource discovery algorithm starts off searching for the required resource in all of the available virtual organizations.

In this section, the algorithm details and the mathematical models for resource discovery time will be thoroughly explained. The resource request model, which has a great effect on the simulation results and follows the Zipfian law, is also clarified.

Resource Request Model

Every user in the pervasive computing environment will submit his tasks to the computing environment to use the available resources. Referring to this fact, resources would be requested when there is a demand for them. According to the pervasive computing environment's essence, some resources would have more requests compared with the others. This behavior is totally dependant on user submitted tasks and cannot be exactly formulated nor a precise probability distribution model be devised beforehand [15]. In data grids, for example, databases are of great interest; however, CPU is a more valuable resource in the computational grid.

Many real world events have been shown to follow an interesting routine, which is formulated as the Zipfian law. Frequency of word usage in the human language, keyword repetition in a programming language, requests submitted to an operating system and even colors in nature follow this rule. In our experiments, the Zipfian law has been used to create the request sequence. This model shows that the probability of a request for resource R_i would be, as follows:

$$P(R_i) = K/i^a, \qquad k = \Big(\sum_{j=1}^{n} 1/j\Big)^{\ 1}. \tag{1}$$

In Equation 1, n shows the number of available resources of type R_i. Suppose that the a constant is set to a value very close to one ($a \approx 1$), then, Equation 1 can be rewritten as:

$$P(R_i) = K/i. \tag{2}$$

As the formula shows, the probability of a request for a resource such as R_i would be $P(R_i)$. The probability is calculated separately for each resource and the request occurrence probabilities are independent of each other. The validity of this model is proved in different research, such as [17,18].

Internal Machine Organization

A centralized resource information store organization has always caused overall system performance degradation. The main reason for this is the bottleneck created while accessing the centralized information store. In the proposed algorithm for resource discovery and dissemination, a distributed resource information store has been used. In this approach, every existing machine has a partial view of the existing resources in the environment. This view varies for different machines, based on their location. For example, each machine has exact information about all of the resources residing in the same virtual organization, but has a very high level and vague information about the resources on other virtual organizations. The accurate resource information is only available to resources in the same virtual organization, hence, a machine on virtual organization Vo_1 is only aware of a resource, such as R_i, available on Vo_2, but is unaware of its physical attributes and placement. This kind of resource dissemination causes lower message passing inside the environment. The information stores in the author model are called the capability tables, which have been devised, based on a similar concept in [19]. Each machine would make decisions, based on the available information in its capability table.

Resource Dissemination Algorithm

Algorithms proposed for resource dissemination must be carefully examined for the traffic load they create in the pervasive computing environment. As most of the algorithms use message passing as the base concept, an inefficient algorithm may cause loads of unnecessary messages being passed through the network and cause unwanted system overload. A hierarchical resource dissemination algorithm has been proposed to avoid system overload and redundant message passing.

A resource dissemination algorithm is started when a resource is added to a machine located within the pervasive environment and the owner decides to share the resource, based on different criteria, with the other users. The resource management system initiates message passing at two simultaneous levels. In the first level, detailed information, including static resource attributes, resource access models and the resource physical location, is sent to every machine in the same virtual organization. In this way, the machines in the same virtual organization are aware of the exact details of the resource. On the other level and to introduce the resource to the machines in other virtual organizations, only the static attributes of the resource are sent. At this level, the message is not directly sent to the destination machine to follow the cellular environment structure. This structure would hide the internal complexities of every virtual organization from an outside viewer. To distribute the messages between other virtual organizations, only one message is sent to the destination's virtual organization's attorney, who is, itself, chosen, based on parameters such as accessibility, accessibility speed, message queue etc. Since the attorney is dynamically selected, based on the introduced parameters, a single machine would not act as the attorney throughout the virtual organization's lifetime and avoid bottleneck creation. The attorney having received the message, would, then, broadcast the message for all of the neighboring machines in the virtual organization. There are many advantages in using this model. On the one hand, restricting direct message passing between virtual organizations will enhance security issues, on the other hand, broadcasting the advertisement messages inside virtual organizations and only sending single messages between virtual organizations alleviates the message passing burden on the pervasive environment network backbone and distributes the message passing load inside the virtual organizations, which most often use a LAN as their network infrastructure. For this reason, the public environment infrastructure is not occupied.

The pseudo code showing the dissemination algorithm functionality is shown in Algorithm 1.

```
Function RMS.Resource Dissemination (Resource)
{
    Source_VO = Resource->VO;
    Machine_identifier=Resource-> machine_identifier;
    Broadcast (Source_VO, machine_identifier,
                Resource->static_attributes);
    For (every other virtual organization as DVO)
        Send(DVO, Source_VO,
                Resource->static_attributes);
}
```

Algorithm 1. Proposed resource dissemination algorithm.

Resource Discovery

Many different tasks are forked when an application is run on the pervasive computing environment. The tasks will gradually introduce their resource requirements. At each step, the task would not be successfully run until the resources are allocated to it. The resource discovery algorithm should be able to spot the suitable resources based on the task specified requirements.

The resource discovery algorithm is very much dependant on the resource dissemination model used [8]. If resource dissemination is done according to a centralized resource information store, the discovery process could easily be done with simple direct queries on this database.

As the authors' resource dissemination algorithm uses distributed resource information stores called capability tables, which are spread among all the machines in the environment and, as each capability table cannot provide an integrated view of the pervasive computing environment status, resource discovery cannot be easily implemented. A resource discovery algorithm needs to be designed based upon the pervasive computing environment behavior and requires some built in intelligence [20]. There are several factors that should be considered before any design decisions are made. The end platform and the type of applications that have been targeted should be specifically clarified. Real-time applications may compromise more than a few of the performance parameters to gain benefit from their punctuality. Hard real-time applications may enforce even more strict restrictions, since "after deadline solutions" are of no worth. However, other applications that may focus more on the quality of their services and tend to guarantee the QoS, seem to be in favor of the methods that, although lacking in speed, provide suitable functionality. Each of the two approaches seem to be totally reasonable; however, a clear distinction between the target applications at design time should be made. The focus is mainly on designing algorithms that favor QoS; hence, the main spotlight is to provide the means of reliable service. This intention may indeed affect time-to-deliver factors in the algorithms, but they can be conceded to allow QoS based applications like vital data-driven applications. To preserve fault tolerance and quality of service in the algorithm, software agents were chosen as one of the base elements of the proposed resource discovery methods [21]. Using lots of lightweight homogeneous mobile agents with the same functionality will decrease the dependency of the algorithm to each of the agents and, therefore, increase fault tolerance. In this way, if a number of agents are, for some reason, discarded in the environment, the total functionality of the algorithm would not be affected; however, optimal solutions may not then be reached. Mobile agent application could help increase information transfer, based on existing agent negotiation models between virtual organizations. The overhead that the use of mobile agents will create in the system is not neglected and a thorough analysis has been undertaken to show overall system resource occupation by the agents presented in the following sections.

Flat (Non-Hierarchical) Resource Discovery

In the first model, there is a non-hierarchical approach to resource discovery. In the flat resource discovery algorithm, the machine capability tables are queried when a resource is requested by a user submitted task. The query is made, based on the static attributes requested by the task. The reason for using the static attributes is that only these kinds of attributes are advertised and available in the capability tables as the result of the resource dissemination process. The query would then introduce the available resources in the pervasive environment that matches the static attribute constraints. A number of mobile agents matching the number of introduced resources by the previous query are created to explore the dynamic attributes of each resource. In this section, based on the mobile agent's destination, algorithm functionality is divided into two sections. This division is made, due to the fact that some agents may travel around the same virtual organization or may travel to other virtual organizations:

- Mobile Agents Exploring the Same Virtual Organization:
  This kind of mobile agent will not pass the source virtual organization boundaries. They are directly sent to the machines that are hosting the spotted resource. The agent would query the resource interfaces to extract its dynamic attributes (like the printer queue length). Having gained the required information, the agents will return to the source machine;
- Mobile Agents Sent to Other Virtual Organizations:
  As the capability table residing on each machine does not contain exact information about resources available in other virtual organizations and its information is limited to the static attributes of the resources and their virtual organization identifier, the source machine is compelled to send the mobile agents to the destination virtual organization's attorney.

The mobile agent sent to the attorney will then negotiate with the resource management system of the host and pass over the specifications of the needed resource. The specifications are then sent over to the resource discovery module which will, in turn, search inside its virtual organization for the required resource, based on the method presented in the previous section. In this process, local mobile agents are created to search the internal space of the virtual organization. The results are then summed up by the resource discovery module and the best choice is passed back to the resource management system, which will, in turn, present the waiting mobile agent with the best solution. The agent will return to its source virtual organization to provide its original host with the requested information.

The main reason for searching inside each virtual organization is that there might be several homogeneous resources with the same static attributes, which have different dynamic performance characteristics. At the end of the exploration, all the created mobile agents will return to their original host and the best choice would be made from the different solutions provided by each agent. The pseudo code for the flat resource discovery is shown in Algorithm 2.

The total exploration time required to explore and find a suitable resource follows the subsequent formula:

$$T_{\exp} = kT_{ac} + 2 * \operatorname*{Max}_{1 \le j \le k} \{T_{cvot_j}\} + k'T_{ac} + \operatorname*{Max}_{1 \le j \le k'} \{\vartheta_j\}, \tag{3}$$

where:

$$\vartheta_j = 2 * \operatorname*{Max}_{1 \le i \le j} \{T_{ivot_i} + T_{q_i}\}, \tag{4}$$

$T_{\exp}$:	total exploration time,
T_{ac}:	time to create one agent,
K, K':	number of agents,
T_{cvot}:	time to jump to next VO,
T_{ivot}:	time to jump to next node,
T_q:	time taken to query a resource.

```
Function RMS.Resource_Discovery1 (Rqst)
{
    no_match = lookupAvailableRsc (Rqst->condition);
    VOList = lookupVOList(no_match);
    createAgents (no_match);
    sendAgents ();
    ForAll (currentAttorney in VoList->Attorney)
        {
        negotiateAttorney (currentAttorney);
        initResourceDiscoveryInDestVO (VOList->ID);
        }
    returned_Res = summonAgents ();
    return conclude (Rqst->status,returned_Res);
}
```

Algorithm 2. Flat resource discovery algorithm.

Hierarchical Resource Discovery

Although flat resource discovery provides us with a reasonable solution, it has several shortcomings. For this reason, a hierarchical model for resource discovery is proposed. This algorithm basically functions the same as the flat algorithm. The main difference between the two algorithms is that the hierarchical algorithm gives high priority to the resources located in the same virtual organization as the process requesting the resource. The concept is implemented by activating the mobile agents in two phases. The first groups of agents are sent to the resources available in the same virtual organization, if the queried resources are shown to be suitable for the waiting task they are selected and the discovery process is terminated. But, if a suitable result could not be found, other agents are forked and sent to other virtual organizations for resource discovery.

The main advantage of this model is that it decreases the resource usage costs by preferring the resources in the same virtual organizations. However, if a suitable resource is not found in the first phase, the algorithm would take longer compared with the flat resource discovery algorithm. The hierarchical resource discovery algorithm is shown in Algorithm 3.

```
Function RMS.Resource_Discovery2 (Rqst)
{
    source_VO = Rqst->source;
    no_match = lookupAvailableRsc (Rqst->condition);
    createAgents (no_match);
    sendAgents ();
    negotiatemachine ();
    returned_Res = summonAgents ();
    if ( !conclude (Rqst->status,returned_Res))
    {
    no_match = lookupAvailableRsc (Rqst->condition);
    createAgents (no_match);
    sendAgents ();
    negotiateAttorney ();
    returned_Res = summonAgents ();
    return conclude (Rqst->status,returned_Res);
    }
return conclude (Rqst->status,returned_Res);
}
```

Algorithm 3. Hierarchical resource discovery algorithm.

The total exploration time required to explore and find a suitable resource in the hierarchical algorithm obeys the following formula:

$$T_{\exp} = \vartheta + \rho(kT_{ac} + 2 * \max_{1 \le j \le k}\{T_{cvot_j}\} + k'T_{ac} + \max_{1 \le j \le k'}\{\vartheta_j\}), \quad (5)$$

where:

$$\vartheta_j = 2 * \max_{1 \le i \le j}\{T_{ivot_i} + T_{q_i}\}. \quad (6)$$

ρ is set to one if the resource is not found in the phase and zero if the first attempt is successful.

Using mobile agents can have two other advantages compared with other existing models. The first advantage of using mobile agents is the added ability to calculate network traffic load. Mobile agents most likely travel through two different paths in a round trip to their destination (this is very much related to the routing algorithm used in the internet protocol layers), which would be very useful to calculate the network traffic load or predict the future state. Being aware of the network traffic conditions, the time for a task to migrate to the destination machine would be conceivable.

In many resource management systems, having a module to track and update the information about the leaving resources is very essential. These modules are mainly designed based on message passing schemes. In the proposed model, no algorithm, other than the resource discovery algorithm, is used to detect unavailable resources. When a resource leaves the pervasive computing environment, no event is triggered and only the capability table of the host is reorganized. If a mobile agent is sent to the host to query a resource's dynamic attributes and finds it unavailable, it would return to its original host and would report a missing resource. The host would then update its capability table and broadcast a missing resource notification within its virtual organization. The machines receiving the message would also update their capability tables.

ALGORITHM PERFORMANCE EVALUATION

A suitable simulation environment was set up to implement the proposed algorithms and assess their performance. Modular design and implementation were precisely observed to allow rapid change in the environment structure, while testing different approaches. All the functionality was implemented within defined classes. The communication between different modules of the system was implemented based on the message passing schema, so that modular dependency is decreased to a minimum.

The implementation of the simulation environment was done using the Java programming language. Mobile agents were simulated by background threads which allow a parallel execution theme. A MySql database management system was used to create the agent capability tables. The reasons for using MySql were its complete support for SQL querying, easy installation, lightweight on deployment and customizability, which allows a rapid development of the simulation software. However, an actual implementation of the algorithms, due to their complexity, will, indeed, require a simple data structure implementation and the use of a DBMS will be infeasible. Although MySql provides a very lightweight DBMS configuration, it will still be too heavy for such cases and the use of a simple data structure will suffice.

The other very important aspect that should have been considered while designing the simulation environment was to create a stable criterion to allow global algorithm comparison. For this reason, a simulation clock in the heart of the system was prepared to provide the basis for an algorithm performance comparison. Most of the comparison criteria introduced in the following sections are mainly based on the number of cycles that the algorithm has taken to perform its specific tasks. The simulation clock was mainly responsible for keeping track of the number of cycles that have past since a particular time (T_0).

Performed Experiments

Resource discovery and dissemination algorithms can show dissimilar results under different resource distribution models. For this reason, and to evaluate the proposed algorithms under various environment conditions, several resource distributions were created. Therefore, environment functionality and essence were changed in different experiments, which is a good measure to assess the algorithms [22]. Fifteen sets of experiments were executed under three different resource distributions.

In the first five sets of experiments, the number of virtual organizations, along with the number of machines in the environment, were kept constant at 10 and 100, respectively. In these experiments, the number of resources were set at 10, 20, 30, 40 and 50 to evaluate the algorithm scalability. The distribution is shown in Figure 1. The experiments were conducted to study the effect of the number of resources available in the environment on the algorithm scalability and performance.

In the next five sets of experiments, the number of resources and virtual organizations were set to 50 and 10, respectively, while the number of machines varied in different experiments between 100 and 500. The experiments studied the effect of available machine

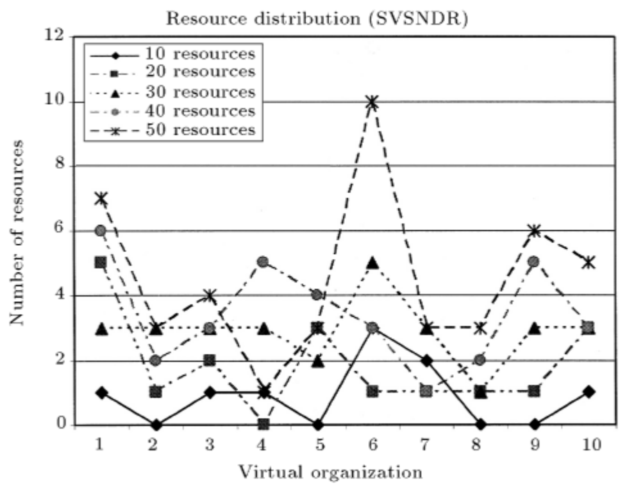

Figure 1. Resource distribution model in the first five experiments.

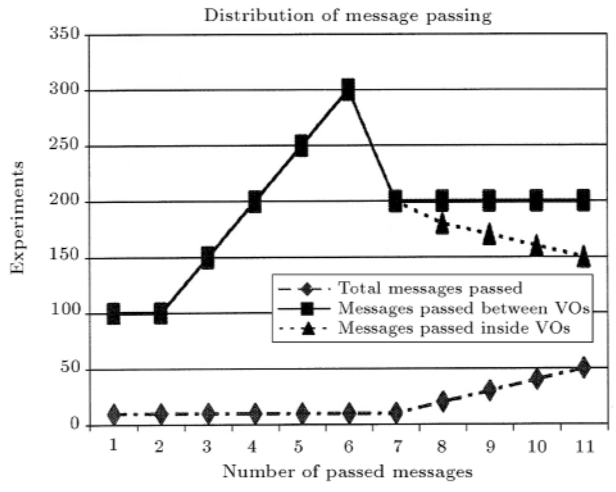

Figure 2. Message passing scheme in the resource dissemination algorithm.

number on the proposed algorithms. Many algorithms show performance reduction with an increase in the number of machines.

The last five sets of experiments changed the number of the available virtual organizations between 10, 20, 30, 40 and 50. These changes were made while the number of resources and machines were kept constant at 50. Scalability should be carefully considered, while the number of virtual organizations changes in different experiments.

Performance Assessment Criteria

In order to compare the designed and implemented algorithms under different experiments, several algorithm performance assessment criteria were devised. Eight main parameters were carefully designed, which will be precisely introduced in the following parts.

- Number of Messages Passed:
  As previously explained, the resource dissemination algorithms are compelled to use the message passing schema. The algorithms with fewer messages passed throughout the advertisement process show better performances. As can be clearly seen in Figure 2, in the devised algorithm, more than 85 percent of the messages have been sent locally (messages broadcasted inside each virtual organization). The ratio of the number of messages passed inside the pervasive computing environment infrastructure to the number of local messages is 0.17, which shows very low network traffic creation and, therefore, low network resource consumption. The proposed resource dissemination algorithm shows to be efficient under different conditions;
- Mean Advertisement Time:
  A change in the environment (e.g., resources added to a machine) requires the capability tables to be updated. The average latency time required to update the capability tables, from the time that the change had actually occurred, calculated in different experiments, is called the mean advertisement time. Although the dissemination algorithm performance is shown using the number of messages passed factor, the mean advertisement time factor can be studied to show algorithm scalability. As shown in Figure 3, the algorithm has kept a constant rate in different experiments, from this factors point of view, and is shown to be scalable;
- Path Length Ratio:
  This parameter is used to show the discovered path optimality in the resource discovery algorithm. The calculation is done using an omniscient in the environment. When a resource request is sent to the resource management system all possible resources are selected and the longest path is then calculated. This length is divided by the length of the selected

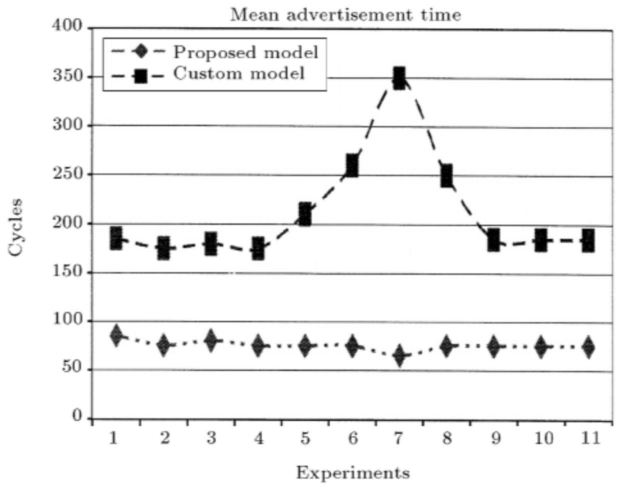

Figure 3. Mean advertisement time in the proposed algorithm.

path provided by the discovery process. The path length ratio shows near 200% improvement in the path length selection in both proposed algorithms (Figure 4). As can be seen in this figure, the ratio is mostly between 0.4 and 0.55, which confirms a much more optimal path selection scheme in the authors' algorithm;

- System Overhead (Load):
This parameter is calculated using the multiplication of the average number of active agents by the mean lifetime of each agent, which is shown with O_i in Equation 7.

$$O_i = \#\text{agents}_i \times \overline{\text{Age}(\text{agents}_i)}. \tag{7}$$

The more active mobile agents with a longer lifetime are created, the more computational and network resources are consumed and, therefore, algorithm performance decreases. The proposed discovery algorithm's performance has been shown in Figure 5. An appropriate overlay model, based on the experiments' essence, shows that, whenever the requests for resource discovery increase, a rise in the systems overload happens. However, the number of available resources of a specific type is also relevant to the created overhead;

- Minimum Discovery Time:
The first agent to return to its original host with the required information will have the minimum (fastest) resource discovery time. Although this factor is not used in the resource discovery algorithm performance evaluation, it assists one in proving simulation correctness. As can be seen in Figure 6, the minimum resource discovery times in both algorithms are very close to each other. The main reason for the performance similarity is related to the algorithm functionality. In the flat resource discovery algorithm, the fastest agent to return the requested information would be the mobile agent that has been sent to the same virtual organization. On the other hand, in the hierarchical resource discovery model, the mobile agents used in the first phase would end up faster and have a shorter time of resource discovery. Conceptually comparing both algorithms shows that the minimum resource discovery time in both models should be very much similar and follow the same trend. Figure 6 shows the simulation results, which confirm this concept and prove simulation validity. The slight difference seen in the figure is related to the random resource distribution that can be neglected in the real world experiments;

- Maximum Resource Discovery Time:
This factor calculates the average time spent to locate a suitable resource in the pervasive computing environment by each proposed resource discovery algorithm. T_{exp} can be computed, based on Equa-

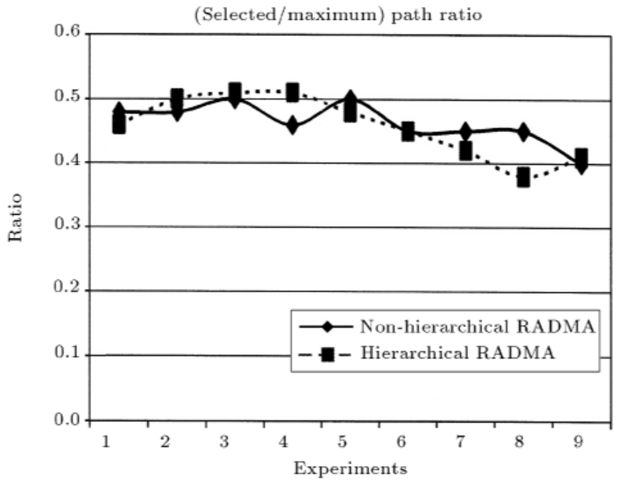

Figure 4. Path length ratio in both proposed resource discovery algorithms.

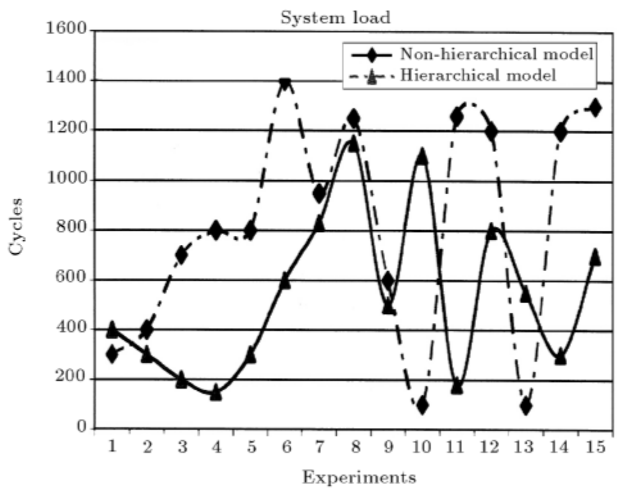

Figure 5. System overhead created in each algorithm.

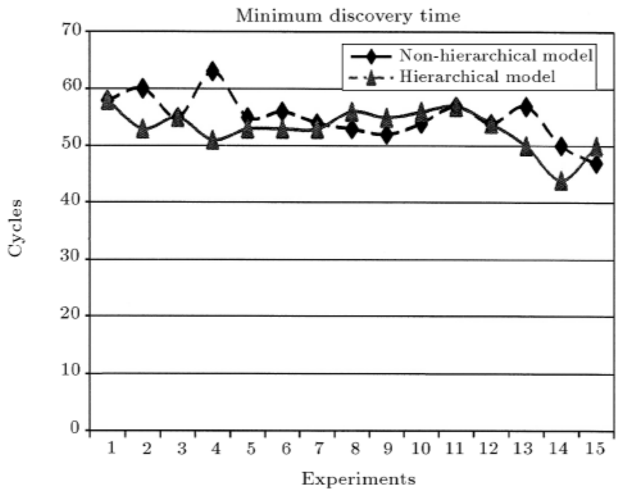

Figure 6. Minimum (fastest) discovery time in the proposed algorithms.

tion 3 or Equation 5. For faster resource discovery, one requires lower effective resource discovery time from the designed algorithms. Figure 7 shows faster resource discovery time for the hierarchical resource discovery algorithm. However, the flat resource discovery algorithm shows to be very scalable and provides similar speeds in resource discovery under different resource distribution conditions. As can be clearly seen in Figure 7, although the non-hierarchical algorithm follows a more lengthy resource discovery process, it follows a constant trend, shows similar behavior under different circumstances and seems to be scalable;

- Algorithm Performance:
  To precisely calculate the resource discovery process performance, the AP_i factor was devised, considering the pervasive computing environment essence. AP_i is based on three main factors, which are: The probability of the request for a specific resource type, the available number of resources from a specific type and the effective resource discovery time. The more resources available for a specific request, the faster the resource discovery process will be performed. This shows that the number of available resources of the specified type has a reverse effect on the effective resource discovery speed. On the other hand, the probability of a request for a specific resource type has direct effect on the weight of the effective resource discovery time of that specific resource on the total resource discovery algorithm performance. The algorithm performance measure has been formulated and shown in the following equation:

$$\text{AP}_i = \frac{P(R_i)}{\#R_i} \times T_{\text{exp}_i}. \tag{8}$$

The algorithms' performance has been compared in Figure 8. The most important fact in the comparison is that both algorithms have similar performance measures. Following the very close and similar trend lines of both of the algorithms that have been demonstrated in Figure 8, it can be inferred that, although the algorithms were tested under different resource distribution and network conditions, a constant similar movement has been followed in both approaches. As the algorithm performance values under different circumstances show to stay between 8.5 and 13, the algorithms seem to be scalable from this point of view;

- Algorithm Efficiency:
  In order to evaluate the relation between the effective resource discovery time and the overall overhead, created by the performed process on the pervasive computing environment, the E_i factor was designed. The total overhead that had been forced on the environment, based on the use of mobile agents, was calculated. The total computed value was divided by the number of available resources to assume an evenly spread load among all available resources for the sake of comparison and simplicity; however, the overhead was not actually evenly spread. The generated overhead has a reverse relation with the resource discovery algorithm efficiency. The algorithm efficiency factor has been formulated in the following equation :

$$E_i = (\#R_i \times T_{\text{exp}_i})/O_i. \tag{9}$$

In this factor, a similar parallel trend can also be shown for both algorithms. The trend lines depicted in Figure 9 show that both algorithms are conceptually behaving the same; however, the non-hierarchical algorithm has a higher efficiency. Figure 9 compares both algorithms' efficiency factors under different simulation conditions.

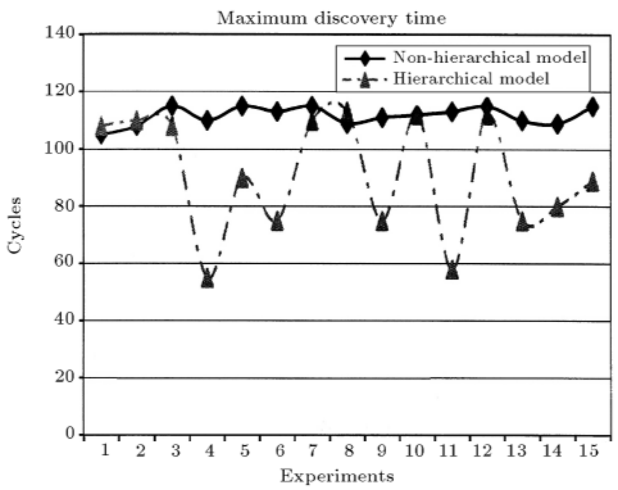

Figure 7. Effective resource discovery time for the proposed algorithms.

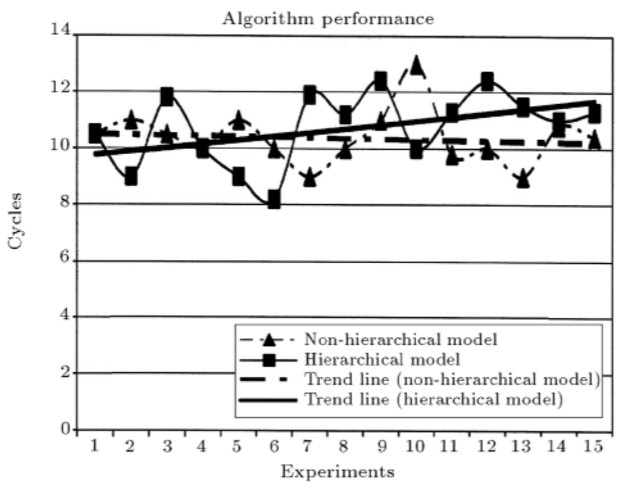

Figure 8. Resource discovery performance comparison in the two different proposed algorithms.

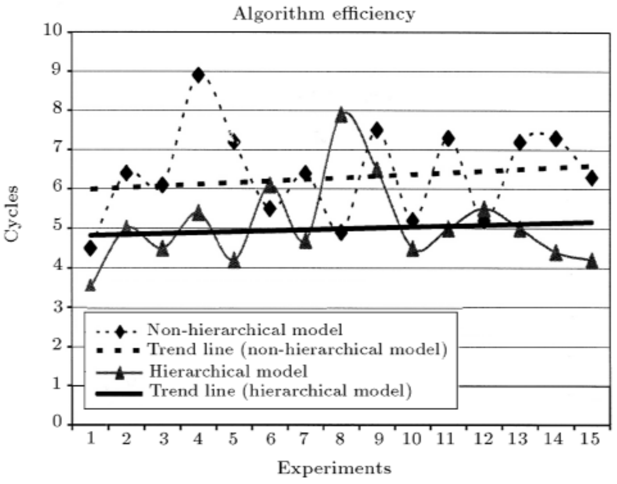

Figure 9. Resource discovery efficiency factor for the two proposed resource discovery algorithms.

CONCLUSION

Two different approaches to resource discovery and dissemination have been proposed in this article and their details have been precisely explained. The exact formula for the resource discovery time has been provided, while the complete algorithms have been shown in pseudo code. The simulation environment under which the experiments were conducted, was thoroughly clarified. The resource request model in the simulations that followed the Zipfian distribution was also explained.

The conducted simulations were divided into three completely different sets of experiment, which were inherently different in nature because of the type of resource distribution in the environment. The experimental results for the two models were compared under eight devised factors and the simulations were proved to have been executed correctly.

The results show that the resource dissemination algorithm performs in the same way under different experimental conditions and proves to be scalable. As the main message passing load has been moved inside the virtual organizations, the algorithm avoids message contention in the pervasive computing infrastructure.

The comparison between the two designed resource discovery algorithms shows that the hierarchical algorithm performs better, based on the efficient resource discovery time and the system overhead created. On the other hand, the non-hierarchical algorithm shows more efficiency, (E_i). This algorithm is also scalable and shows similar behavior under different simulation conditions.

REFERENCES

1. *Realizing the Information Future: The Internet and Beyond*, National Academy Press (1994); http://www.nap.edu/readingroom/books/rtif/.
2. Stevens, R., Woodward, P., DeFanti, T. and Catlett, C. "From the I-WAY to the national technology grid", *Communications of the ACM*, **40**(11), pp 50-61 (1997).
3. Oram, A., Ed. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly Press, USA (2001).
4. Foster, I. and Kesselman, C., Eds., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann (1999).
5. Oppenheimer, D., Albrecht, J., Patterson, D. and Vahdat, A. "Distributed resource discovery on planet lab with SWORD", In *Proceedings of the First Workshop on Real, Large Distributed Systems* (December 2004).
6. Nabrzyski, J., Schopf, J.M. and Weglarz, J., Eds., *Grid Resource Management: State of the Art and Future Trends*, Chapter 26, Kluwer Publishing (2003).
7. Czajkowski, K., Fitzgerald, S., Foster, I. and Kesselman, C. "Grid information services for distributed resource sharing, high performance distributed computing", *Proceedings 10th IEEE International Symposium*, pp 181-194 (7-9 Aug. 2001).
8. Buyya, R., Stockinger, H., Giddy, J. and Abramson, D. "Economic models for management of resources in peer-to-peer and grid computing", *SPIE International Conference on Commercial Applications for High-Performance Computing*, Denver, USA (2001).
9. Cao, J., Jarvis, S., Saini, S., Kerbyson, D. and Nudd, G. "ARMS: An agent-based resource management system for grid computation", *Scientific Programming*, pp 135-148 (2002).
10. Waldo, J. "The jini architecture for network-centric computing", *Communications of the ACM*, pp 76-82 (1999).
11. Veizades, J., Guttman, J., Perkins, E. and Kaplan, S., *Service Location Protocol*, RFC 2165 (1997); http://www.ietf.org/rfc/rfc2165.txt.
12. Fox, S., Chawathe, Y. and Brewer, E. "Adapting to network and client variation using active proxies: Lessons and perspectives", *IEEE Personal Communications*, pp 10-19 (1998).
13. Brumitt, B., Meyers, J., Krumm, A. and Shafer, S. "Easy living: Technologies for intelligent environments", *Handheld and Ubiquitous Computing 2000 (HUC2K)*, pp 97-119 (2000).
14. Brunett, S., Czajkowski, K., Fitzgerald, S., Foster, I., Johnson, A., Kesselman, C., Leigh J. and Tuecke S. "Application experiences with the globus toolkit", In *Proc. 7th IEEE Symp. on High Performance Distributed Computing*, IEEE Press, pp 81-89 (1998).
15. Johnston, W.E., Gannon D. and Nitzberg, B. "Grids as production computing environments: The engineering aspects of NASA's information power grid", In *Proc. 8th IEEE Symposium on High Performance Distributed Computing*, IEEE Press, pp 197-204 (1999).
16. Buyya, R., Abramson, D. and Venugopal, S. "The Grid economy", *Proceedings of the IEEE 2005*, **93**(3), pp 698-714 (2005).

17. Chervenak, A.L. "Tertiary storage: An evaluation of new applications", Ph.D Thesis, UC Berkeley (Dec. 1994).

18. Sitaram, D. and Shahabuddin, P. "Scheduling policies for an on-demand video server with batching", *Second Annual ACM Multimedia Conference and Exposition*, pp 15-23 (1994).

19. Cao, J., Kerbyson, D. and Nudd, G. "High performance service discovery in large-scale multi-agent and mobile-agent systems", *International Journal of Software Engineering and Knowledge Engineering*, pp 621-641 (2001).

20. Aloisio, G., Cafaro, M., Epicoco, I., Fiore, S., Lezzi, D., Mirto, M. and Mocavero, S. "iGrid, a novel grid information service", *Proceedings of the First European Grid Conference (EGC)* (2005), *LNCS 3470, Lecture Notes in Computer Science*, Springer-Verlag, pp 506-515 (2005).

21. Badshaw, J., *Software Agents*, AAAI Press/MIT Press (1997).

22. Bester, A.J., Bresnahan, J., Chervenak, A.L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D. and Tuecke, S. "Secure, efficient data transport and replica management for high-performance data-intensive computing", In *Mass Storage Conference*, pp 13-27 (2001).