Collaborative Resource Discovery in Pervasive Computing Environments*

Jameela Al-Jaroodi†, Ahlam Kharhash, Amal Al-Dhahiri, Asma Shamisi, Aysha Dhaheri, Fatima AlQayedi and Sheikha Dhaheri
College of Information Technology
United Arab Emirates University, P. O. Box 17551, Al Ain, UAE

ABSTRACT
Collaborative resource discovery mechanisms are available for environments with stationary to semi-stationary resources. Yet in pervasive computing environments, where all nodes are mobile and many of which connect in an ad hoc manner, incorporating those dynamic mobile resources introduces a new set of requirements. The main issue is the lack of a reference point – a minimum of one stationary node that can be used as a start point for the environment’s organizational architecture. As a result, one of the requirements becomes discovering when a resource is available and making it available for users/applications. Another is identifying the resources, their current locations and mobility attributes. In addition, there is the need to provide users/applications with the tools to identify moving resources and discover when they become unavailable. In this paper, we introduce a middleware framework for collaborative resource discovery in fully mobile and pervasive environments. The proposed framework provides decentralized mechanisms to dynamically discover resources, include them in a usable collective, and manage their mobility requirements. The framework involves a decentralized collaboration of software agents residing on the group members’ nodes to collect, organize and efficiently exchange resources information, while keeping track of the nodes’ mobility.

Keywords: Collaborative Agents, Middleware, Pervasive Computing, Resource Discovery.

1. INTRODUCTION
Large computing environments whether stationary, mobile or both, suffer from the issues related to discovering, allocating and managing the available resources. Therefore, it is important to develop and use efficient resource discovery mechanisms for these environments. Our main concern here is pervasive environments where nodes and resources are connected in an ad hoc manner and generally all the nodes are mobile. As a result, they rely strongly on several enabling technologies including wireless networks, mobile computing, and middleware. In general, some current research approaches address the issues related to physical and computing resources (CPUs, memory, storage, servers, network connections, etc.) that support distributed environments such as heterogeneous clusters and pervasive computing devices. Further more, much research was done in middleware for distributed systems and for mobile ad hoc networks. For example the authors [12] survey several of these approaches and in [13] the authors provide a similar survey for wireless sensor networks, which may be considered important components in pervasive environment. However, very few discuss resource discovery as middleware services or as collaborative techniques among the nodes in the environment.

The initial research in distributed systems middleware architecture [1] provides a basis for developing distributed applications on heterogeneous environments. Furthermore, a framework for collaborative environments in this architecture was developed [3] earlier for stationary (fixed topology) distributed environments. Later a more generic framework supporting semi-stationary environments was proposed [2]. In this paper we move towards applying similar middleware collaborative techniques for pervasive systems. We aim to explore, in terms of resource discovery, the design issues for these environments and their requirements. Then a middleware framework for the mobile environment is introduced to facilitate pervasive applications and provide efficient collaborative resource discovery mechanisms.

In this paper, Section 2 is an overview of some related work and Section 3 discusses resource discovery concepts in
In this paper, we introduce a middleware framework for collaborative resource discovery in pervasive environments. As a result, we investigated several fields of research that are relevant to our work. Topics in pervasive and mobile computing, resource discovery, collaborative systems and distributed systems middleware were our focus. First, pervasive systems rely on mobile devices with networking and computing capabilities like Cell phones, PDAs (Personal Digital Assistants), GPS (Global Positioning System) devices and hand-held computers, the heterogeneity of these devices and the variety of available resources pose several challenges that need to be resolved to enable a wide range of pervasive applications. In the area of collaborative systems, the authors of [2] describe the characteristics of collaborative systems and the challenges faced when developing collaborative applications for mobile networks. In addition, several approaches such as MoCa [23], ReMMoC [11], 3DMA [10], and the MINEMA programme [17] were investigated. Furthermore, the need for efficient resource discovery and recovery is very important and research have covered several areas such as using statistical frameworks [4], Mobiscope [9], metadata trading [19], query mechanisms as in SWORD [20][21], and registration and lookup grid in NEVRLATE [8].

Another important aspect arises when dealing with mobile devices and ad hoc networks, where more issues arise and resource discovery becomes more complex. In [16] the authors investigate the differences between resource discovery in structured networks and in ad hoc wireless networks. In addition they propose a framework for resource discovery that provides several functionalities such as directory information organization and query, dynamic domain formation, and intra- and inter-domain QoS information monitoring. In [5] the authors introduce an adaptive resource management architecture that clearly defines the functions involved in resource management including resource discovery. However, the approach is more suitable for fixed infrastructures. These techniques can be applied in several application domains and are used to support various types of environments like pervasive systems. In other areas such as managing distributed databases over a mobile environment, researchers such as [6] identified the challenges and proposed a framework called MOBI-DIC (MOBILE Discovery of LoCal Resources), which enables building resource discovery services for databases on mobile environments. Furthermore, the authors in [15] propose a middleware architecture called MoGrid that manages the distribution of Grid tasks over mobile devices in a peer-to-peer fashion thus enabling a decentralized approach to resource discovery and allocation in mobile Grid. In [7] the authors propose a distributed service discovery protocol for pervasive environments. Here the resources are the services needed by the various members of the pervasive environment rather than physical or information resources. The protocol relies on peer-to-peer caching of service advertisements and group-based intelligent forwarding of service requests and it does not require a registry and lookup server. Another issue we investigated is the use of software agents in pervasive environments. There are many approaches proposed and the general view is that software agents are an efficient approach to providing the required functionalities for collaborative resource discovery. In [24] the authors provide a mechanism for architectural evaluation of software agents in collaborative systems. The authors describe the quality attributes of agents and the different types of agents that can be used in distributed environment.

Based on the literature review and the examples introduced above, we arrive at some interesting observations. A limited number of resource discovery approaches address the issues of physical resources management and more importantly, very few of these address the resource discovery issues to support mobile and pervasive distributed environments. Furthermore, very few provide techniques to address these issues and provide solutions at the middleware level. In this paper, we approach the issues related to collaborative resource discovery for applications in mobile and pervasive environments as a middleware framework based on collaborative software agents.

3. RESOURCE DISCOVERY IN PERVERSIVE SYSTEMS

Resource discovery deals with issues of finding resources to match application demands. In pervasive systems, it is the physical resources and the computing services that need to be discovered. It is an essential step in the process of managing mobile systems resources and providing applications with the optimal set of resources when required. The issues involved in collaborative resource discovery require a special approach when designed for pervasive environments. Such environments have unique properties [14] some of which are contextual like dynamic computing environments, variable network properties, changing use environments and heterogeneous user platforms. While other properties are non-contextual like burst-type user access, need for immediate availability of services, and small user devices. As a result the middleware approach that we could use here differs significantly from that used for large scale distributed systems with fixed
Several approaches were discussed for middleware design in particular for mobile systems [22]. In our case we will introduce a model that is more closely coupled with the application layer, where the resource discovery process is not particularly concerned with the connectivity issues of the underlying network infrastructure. On the contrary, we are assuming the network protocols are in place and functioning efficiently such that nodes can easily connect to each other and can correctly route messages using multi-hop protocols. Our concern here is the application level discovery process where nodes and resources need to be located, allocated to the application and maintained throughout the application’s life. Therefore, the discovery process will require continuous access to all available resources in the system, which increases its complexity with (1) the increasing size of the environment, and (2) the mobility of the node.

Our approach relies on software agents that will reside on all the nodes in the system to manage the resources. Thee agents will form groups of nodes and will elect a node to become a head node in each group. The head nodes will then collaborate to collect and maintain resources information and manage the system. The abstract group representation will allow mobile nodes to join and leave groups dynamically. Our goal here is to establish a framework that could support an active and efficient resource discovery of mobile nodes and their resources. We are looking at a design that would be able to detect when members join or leave the groups and also detect their current status in terms of connectivity, activeness, available resources, and location.

4. THE FRAMEWORK DESIGN

The framework design for pervasive environments stems from the earlier approaches introduced for:

1. The hierarchical structure for self organization and collaborative resource discovery in large distributed systems with fixed infrastructure [3].
2. The adapted hierarchical structure [2] to support semi mobile environments where at least one node must be stationary in the system.

The middleware architecture (see Figure 1) includes a collaborative hierarchical structure at the middleware level that can support a large group of resources and efficiently manage them (see Figure 2). However this structure relies on the stationary location of resources and the underlying infrastructure. The introduction of a fully mobile environment adds to the complexity of the process and makes it difficult to maintain a fixed hierarchical structure since node cannot be fixed in any one location at all times. In addition, the movements of the nodes will result in continuous changes in the network connectivity and topology, which will result in difficulties in maintaining the hierarchical structure throughout the system. Therefore, it becomes necessary to adapt a decentralized approach for resource discovery.

In pervasive environments multiple mobile nodes are available and applications usually require the utilization of multiple resources available on different nodes. As a result, the system must be capable of making these resources available to the application and it must be capable of keeping these resources available for the full execution time of the application. During this time, some nodes may move around and others may join the network.

<table>
<thead>
<tr>
<th>Distributed and Pervasive Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Middleware Framework (using software agents) including Collaborative Resource Discovery Mechanisms</td>
</tr>
<tr>
<td>Network Services (e.g. Ad Hoc Network Routing)</td>
</tr>
</tbody>
</table>

Figure 1. An architectural view of the collaborative middleware framework.

![Example Hierarchical Structure](image)

Figure 2. Example Hierarchical Structure. Squares Are Leaders and Circles Are Agents. Each Virtual Cluster Is Fully Connected, While Top Level Leaders May Be Connected by Multi-hop Connections.

The proposed framework is based on a dynamic creation and maintenance of resources in the groups. Since we are dealing with ad hoc-type networks, we are assuming the network protocols are successfully handling the connectivity of the nodes and the multi-hop message delivery. However, from the application view, this may not be enough to ensure the continuity of execution if some nodes move away. The assumption is when nodes are identified to be usable by an application and assigned to it, the application will keep track of which nodes these are and
how to directly communicate with all of them. When a node relocates in the network, it will still be connected (but through different routes), but the application may not be able to locate it directly. Therefore, it will lose its resources and may eventually fail. Therefore, we need to devise a middleware-level mechanism to find these nodes and make sure they remain visible to the application during its execution.

The approach we propose is to equip each node with a software agent that will be responsible of all the resources available and for making sure these resources are efficiently allocated and maintained for the applications. The main idea of the solution is to divide the system into different regions or groups of nodes and each group must elect a head node using a predefined election algorithm (See Figure 3). The head node will be elected according to a predetermined attribute such as the mobility level, distance from other nodes, tasks associated with the node, availability, network bandwidth, type of resources, application needs and length of connectivity. The head node will then be responsible for all the other nodes in the group, handling all of their information, and collaborating with the other heads.

Figure 3. An example showing how nodes may be grouped within the system.

4.1 Message definitions
To facilitate the description of our framework we will first describe a number of messages that the agents will need to exchange:

1. MSM: MemberShip Message is sent by the agent on a node that wishes to join the system to the nearest group head. It includes node identification information and a request to join the system.

2. RCM: Registration Confirmation Message this message will be sent from the head to the node to inform it that it is accepted in the group. It includes the head identification information and an update on connectivity information.

3. RRM: Registration Relocation Message is sent from the head to the new node and the new group’s head if the group was divided into two groups and the node is allocated in the new group. This establishes membership for the new node in the new group.

4. HEM: Head Election Message is a notification for all the nodes in a group requesting to initiate the election process to find a group head. This message is either sent by regular nodes when forming a new group or by the current head when it needs to move out of the system (or group) and a new head is needed to replace it.

5. RUM: Resources Update Message sent by a node to its group head with information about the available resources on the node. The same message is used by a head node to update other heads about its collective resources.

6. CLMC and CLMN: Changing Location Messages where the agent sends CLMC to its current group head and the CLMN to the new group head. These messages must be sent sequentially (CLMN followed by CLMC). This ensures that the agent must get the new group RCM before sending the CLMC to leave the current group and to give it the information of its new location.

7. RFRM: Request For Resources Message is used by a node or head to ask for resources needed by an application. It includes the node’s identification information and a list of the required resources.

4.2 Group division policy
In an ad hoc environment, usually few nodes will form a network and applications will start using the available resources, as time passes, more nodes/resources will join the system, which will lead to problems and delays managing these resources. Therefore, the first step will be for the agents to form groups. Forming the groups include the following steps

1. The first set of nodes that started in the system will form one group and members will send a HEM to each other to initiate the election process to choose a head for the group.

2. New nodes entering the system will send MSM to the head to register in the group.

3. If the group size is within the predefined threshold value the head will accept the node and reply with a RCM.

4. If the group size is at the threshold value, then the process to initiate a new group and divide the nodes is executed. At the end of this process, the new node will be sent either a RCM to confirm that it joined the
current group or an RLM to inform it that it has joined the new group. In this case the new group’s head will also receive the RLM identifying the node as a member in the new group.

4.3 Initiating a new group

When a group reaches its size limit, it must be divided into two groups. This process involves three main steps:

1. Deciding which nodes will be moved to the new group.

This decision depends on the current status of the group members. There are several factors that can be considered to select which members to move. One main factor is the members’ involvement in live applications. All members that are busy executing or providing resources to an application need to remain together. As a result, this will be the first criteria where the non-busy members are moved to form a new group. If all members are involved in the same application, then the new node requesting to join will be directed to form its own group with only one member, which also automatically becomes the head. On the other hand, if all members are free then we use the connectivity information to divide them. Therefore, nodes that have better connectivity with each other will be in one group, while others will be in another group.

2. Selecting a head for the new group

After a new group is formed, a head needs to be selected. This process is done through an election algorithm based on some predefined criteria. In the case of this framework, we defined the mobility factor as the selection criteria such that the node with least mobility (does not move a lot) is elected to be the head. This process is also repeated when the head moves out of the group and sends a HEM, which informs the rest of the group that they need to elect a new head to replace the leaving one.

3. Establishing collaboration between the heads to maintain resources information.

As soon as the head is elected, it has the responsibility to contact the heads of the other groups to announce its existence, update them with the group information and request updates for the other groups. The communication between the heads of the groups can be supported by a content-based routing mechanism that allows for exchange of information based on the content rather than specific destinations. A suitable model would JOR [18], which provides an object-oriented, content-based routing mechanism for distributed systems. Decentralized hash indexing could be used to help the heads and their nodes to connect with each other and to connect the heads together too and maintain resource information. Here, the resources are given unique keys, and a hash function is used to build a deterministic mapping between the key and the nodes which store the directory information of the resources. Each head will keep information about the resources available on each node in its group and will continually probe the group members to update that information.

4.4 Agents Collaboration

When groups are formed nodes will maintain connectivity and update the head through their agents. Each agent will collect the resources information of its node and send them to the head of its group. In addition, the agent will keep track of the mobility patterns and connectivity attributes of the node. It will keep track of resources that are currently in use by the applications. The agent will update the head by sending a RUM if any of the following occur:

1. The application releases the resources
2. The node acquires new resources
3. The head requests an update
4. A new application requests resources from the node and they get allocated

The head node will collect the information and aggregate them to form a single image of the collective resources on all the nodes in the group. This information is then exchanged between the heads of all groups such that all group heads will be aware of the available resources across the system. Furthermore, when an application accesses any node and requests resources for its execution, the node will first determine if it has all the resources needed for that application. If so, it will allocate those resources, start the application and update the head with these changes using the RUM. Otherwise, it will contact the head with an RFRM indicating the required resources. Upon receiving the RFRM, the head will check the resources available in the group and allocate the required resources if available. Then the head will need to send a RUM to the other heads indicating those changes. However, if the group does not have all the required resources for the application, then the head will request them from the other group heads and they will collectively provide the resources for the application and update their resource information.

4.5 Monitoring and Handover process

Since the nodes in the system are mobile nodes, at some point some node will move to a new location. The effects of this movement will have one of the following effects:

1. The node is currently not in use by any application, therefore, it will send CLMN to the head of the new group it wishes to join and when it receives the RCM from the new head it will send a CLMC to its current head announcing the move and providing information about its new group. Then RUMs will be exchanged to reflect the changes resulting from the move.

2. The node is a head node, in which case it will act like a regular node to complete the move. However, before leaving its current group (instead of sending the CLMC) it will send HEM to initiate the election
process and then it will handover the information it carries about the group to the new head.

3. The node (regular node or head) is currently in use by some application (See Figure 4). In this case it is not only important to correctly move to a new group, but we also need to make sure that the application will not lose the resources during or after the move. In this case, the head, after receiving the CLMC from the moving node, will first lookup its resources information to find matching resources in the group. In that case it will reallocate resources to the application from other nodes in the group. However, if that is not possible, then the application is given new connectivity information to the node’s new location to maintain utilization of the resources on that node.

![Figure 4. An example of resources allocation for an application.](image)

MA: Agent on mobile node
Head: Agent acting as group head
● : Resource being used by the application

5. CONCLUSION

Resource discovery in mobile pervasive environments involves various issues that differ from other environments where a well defined infrastructure is available. In this paper we introduced a middleware framework for collaborative resource discovery for pervasive computing environments involving a completely dynamic set of nodes and resources. The lack of infrastructure dictated that we deal with nodes in a decentralized mode, where no one particular node will have individual control over the other nodes. Our solution included techniques to allow the service to adapt continuously with the changes in the environment. Several methods were included: one to allow nodes to form logical groupings which increases locality within smaller subsets of the environment; Another to monitor and adapt the groups based on the members’ status where handover, mechanisms allow nodes to seamlessly change groups and election algorithms can be used to select new head nodes when the current one either moves out of the group, it fails or a new group is formed. In addition, the agents in each node keep track of resources in use and maintain their availability to the application throughout its life time. The framework utilization of software agents allowed for higher flexibility and better control over the resources. In addition the dynamic group structure allows each group’s head node to have an up-to-date view of the group’s resources, while maintaining connectivity with other groups’ head nodes to keep track of a generic overall view of the resources in the environment. Agents collaborate continuously to achieve these goals and maintain a single system image of all nodes involved in the system.

We are currently in the process of implementing a simulation environment that would imulate a distributed pervasive system including multiple mobile nodes and applications emulator representing possible application needs for resources. In addition, we are implementing the software agents and the resource discovery protocols as described in this paper to be used with the simulation environment. Using these two components, we will be able to run simulations mimicking various application/environment settings and measure the performance of the framework. This simulation will also allow us to more accurately study the different problems associated with the environment and fine tune the framework to function correctly and efficiently.

REFERENCES


