Peer-to-Peer Resource Discovery in Mobile Grids

Luciana dos S. Lima[1, 2], Antônio T. A. Gomes[2], Artur Ziviani[2],
Markus Endler[1], Luiz F. G. Soares[1], Bruno Schulze[2]

(1) Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rua Marquês de São Vicente, 225 – 22.453-900 – Rio de Janeiro – RJ – Brasil
(2) Laboratório Nacional de Computação Científica (LNCC)
Av. Getúlio Vargas, 333 – 25.651-075 – Petrópolis – RJ – Brasil

{lslima,endler,lfgs}@inf.puc-rio.br, {atagomes,ziviani,schulze}@lncc.br

ABSTRACT

Grids are likely to be the mainstay of distributed computing. Grid was firstly applied as a concept for sharing computing resources among wired nodes, but there has been a growing interest in extending this concept to mobile environments. Although significant work has been done towards mobile grids, much of it has followed centralized approaches based on infrastructure wireless networks. We believe that a less restrictive, decentralized approach that supports *mobile collaboration* in ad-hoc wireless networks can cater for novel grid applications. To address this issue, we propose a middleware architecture called *MoGrid*. MoGrid orchestrates the distribution of grid tasks among mobile devices in a peer-to-peer (P2P) fashion. In this paper, we focus on the *P2P Discovery Protocol* (P2PDP), which is a central element of our architecture. P2PDP aids in distributing tasks among the most resourceful devices, while mitigating the overhead of control messages exchanged among them. We describe a prototype implementation of our architecture and discuss some issues related to the adoption of P2PDP as an ad-hoc resource discovery mechanism in mobile grids.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *Wireless communication*; C.2.2 [**Computer-Communication Networks**]: Network Protocols – *Applications*; C.2.4 [**Computer-Communication Networks**] Distributed Systems – *Distributed applications*.

General Terms

Design, Experimentation.

Keywords

Mobile grids, P2P protocols, resource discovery, mobile collaboration, ad-hoc networks, middleware.

MGC'05, November 28 – December 2, 2005, Grenoble, France.

1. INTRODUCTION

Grids are likely to be the mainstay of distributed computing. Originally conceived of as a concept for sharing computing resources among wired nodes, grids have been receiving growing attention as to their application to mobile environments [1, 11, 13]. There are two main approaches to integrating mobile environments into grids. In the first approach, mobile devices are used just as interfaces to access wired grids [9]. In the second (and more challenging) approach, mobile devices can take part in a grid as task processing nodes, shaping the so-called *mobile grids*.[1]

Mobile grids demand new control patterns for distributed resource sharing since mobile devices allow the establishment of spontaneous, highly-dynamic ad-hoc communities, for example those motivated by common interests or geographical proximity. Crucially, no centralized resource-discovery mechanisms may be available for such a community. In this context, mobile grids can be regarded as being supported by (and, conversely, a support for) P2P technologies. Examples of novel P2P applications a mobile grid can support include disaster handling [1], shared workspaces [7], and distributed recording [13].

Although significant work has already been done towards mobile grids, much of it has followed centralized approaches based on infrastructure wireless networks [12, 20].[2] We claim there still remains the need for a less restrictive, decentralized approach that can support applications, like the aforementioned, in *purely* ad-hoc networks. To address this issue, we propose a middleware architecture called *MoGrid*. The MoGrid architecture arised out of two convergent projects. MoCA [17] is a middleware architecture that supports the development of context-sensitive applications for mobile collaboration. MoCA offers application-level mobility transparency, but was initially conceived for infrastructure wireless networks. InteGridade [10] is a grid infrastructure over the GIGA network [5]—a nationwide academic multigigabit network in Brazil. InteGridade offers transparency and efficiency

[1] We use the term *mobile grid* instead of (the more common) *wireless grid* to stress our focus on mobility, which is not necessarily implied by wireless networks.

[2] In an infrastructure wireless network, communication tipically takes place only between the mobile nodes and a fixed access point. This access point can also act as a bridge to other wireless or wired networks [18].

in the usage of resources distributed over the GIGA network, but is based on a central point of resource coordination.

The MoGrid architecture aims at giving leverage to both aforementioned projects in ad-hoc networks by allowing the distribution of grid tasks in a P2P fashion among *mobile collaborators*. In our proposal, tasks involve not only computing resources, but also communication, peripheral, or storage resources. This is similar to the service-oriented view of grids presented in [3]. Mobile devices can split application requests (*e.g.* for file sharing or process execution) into tasks to be carried out by collaborators. Our architecture does not impose limitations on the ordering of tasks completion that a specific application may need, that is, tasks can be processed independently, sequentially, or in parallel.

In this paper, we focus on the *P2P Discovery Protocol* (P2PDP), which is a central element of the MoGrid architecture. P2PDP helps in coordinating the distribution of grid tasks among the most resourceful collaborators in the mobile grid. This is accomplished by a new mechanism in which replies from collaborators willing to deal with a request are delayed according to a timer. This timer is set to be inversely proportional to the availability of resources in the mobile device. It must be remarked that the determination of reply delays is flexible with regard to the resources to be taken into account and the relative importance among them. Replies are broadcast so that a collaborator having received a specific request can detect whether other, more resourceful collaborators have already answered this request. The first reply suppresses other replies, thus mitigating the amount of messages exchanged among collaborators.

We have developed a prototype implementation of the MoGrid architecture to obtain experimental results on P2PDP usage. In such a prototype, we define some general criteria for determining the availability of resources in mobile devices. We show in this paper the tests that we have conducted on the prototype implementation through a distributed file sharing application.

This paper is organized as follows. Section 2 presents some related work on mobile grids, including a few preliminary attempts towards ad-hoc approaches. Section 3 introduces the MoGrid architecture, giving special attention to the P2PDP protocol. In Section 4, some details of our prototype implementation are presented. Section 5 discusses some relevant issues to our architecture. Section 6 concludes this paper.

2. RELATED WORK

So far, a great deal of work on grids has focused on applying the classic client-server model to distributed computing. For instance, the OGSA architecture [3] is based on a three-tier model, where the middle tier is built around Web services. Nevertheless, the growing interest in enabling grid-based applications for mobile devices has prompted alternative models for grid computing. Such interest relates to the development of P2P grids [2, 4, 15], although most approaches to that still depend to some extent on core servers supporting a global grid system. The remainder of this section discusses some specific projects that mainly focus on mobile technologies.

The ISAM project [20] proposes the integration of three main concepts—context-awareness, mobility, and grid computing—in a pervasive computing environment. This environment supports the development of distributed mobile applications that have adaptive behavior. Nevertheless, ISAM assumes that mobile devices use some basic services deployed in a fixed network. Thus, its application is limited to infrastructure wireless networks.

Kurkovsky *et al.* [12] propose a grid-based problem-solving environment for mobile devices, which are viewed as collaborative agents in a multi-agent system. Such an environment addresses the issues of distribution, coordination, and assembly of complex grid tasks, as well as network instability, access transparency, and dependability. The overall approach depends on a central element running on (or close to) a base station. Such a central element is responsible for coordinating the resource sharing among the agents. Therefore, purely ad-hoc networks are again disregarded.

The K*Grid project [11] aims at providing a comprehensive research environment for both industry and academia through a nationwide grid in South Korea. K*Grid envisions the use of idle resources in a large number of mobile devices to form pervasive mobile grids. Nevertheless, such a project is in a very early stage of development, without any specific results so far.

The AKOGRIMO project [1] is an European-funded project aiming at architecting and prototyping a *next generation grid* based on OGSA. Currently, AKOGRIMO focuses on devising novel grid applications over evolving mobile IPv6-based infrastructures. The domain of such applications range from e-health, through e-learning, to disaster handling and crisis management. Like K*Grid, the AKOGRIMO project does not have any specific results yet.

Overall, what still appears to be missing in the realm of mobile grids is an adequate collaboration support for applications running on purely ad-hoc networks. Crucially, such grid applications need a completely decentralized and collaborative approach to the resource discovery and coordination. Although K*Grid and AKOGRIMO seem to be promising approaches, there are currently no particular results on their effectiveness to support purely ad-hoc networks.

3. MOGRID ARCHITECTURE

To support grid services in ad-hoc networks, we propose a mobile grid middleware architecture, called *MoGrid*, that is independent of centralized elements for decisions on resource sharing. The MoGrid architecture comprises a *P2P discovery layer* and a *transparency layer*, as depicted in Figure 1.

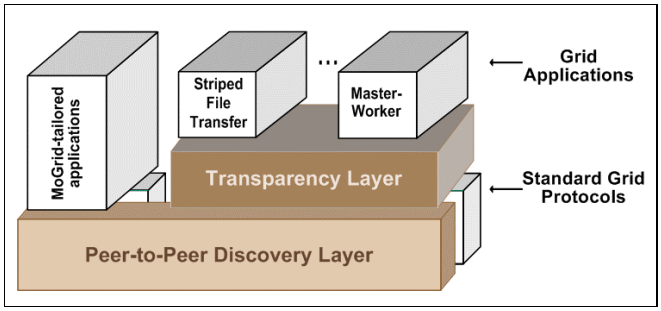

Figure 1. MoGrid middleware architecture.

The middleware architecture supports two main types of applications. *MoGrid-tailored applications* make direct use of the

services offered by the P2P discovery layer (see Section 3.1). For *standard applications*, the use of such services is made through the transparency layer (see Section 3.2). Irrespective of the application type, the middleware supports the application in two phases. First, resources are discovered among the participating devices by means of the P2PDP protocol. Second, tasks are submitted to selected participants according to the resources they make available. Task submissions can be carried out through various standard grid protocols (*e.g.* GridFTP [6]), and are out of the scope of this paper.

3.1 P2P Discovery Layer

The P2P discovery layer is composed of three main parts: the *discovery API*, the *entities* involved in resource discovery, and the *P2P Discovery Protocol*.

3.1.1 Discovery API

Applications executing on a mobile device must use operations available at the discovery API—either directly or through the transparency layer—to take profit of the MoGrid services. The discovery API provides applications with operations for resource registration and announcement, context definition, and resource discovery. Figure 2 shows the main operations of this API.

```
resID = register( resourceDescriptor )
deregister( resID )
reqProfile = createRequestProfile( ctxtInfo,
                        numMaxReplies, maxReplyDelay )
repList = discover( resourceQuery, reqProfile )
```

Figure 2. Discovery API.

When a mobile device registers some resource in the middleware (operation `register()`), such resource becomes accessible to other devices in the mobile grid. The registration can be immediately announced to the other devices, or announcements of new available resources can be done on demand, as a result of application requests for resource discovery (operation `discover()`). Moreover, the API allows applications to previously customize their requests for resource discovery (operation `createRequestProfile()`). Such a customization is made with regard to the number of collaborators the initiator wishes to involve (`numMaxReplies`; zero means as many collaborators as are available in the mobile grid), how much time the application is willing to wait for replies (`maxReplyDelay`), and the contextual information of interest (`ctxtInfo`). The context information determines which resources are to be taken into account in a request and the relative importance among them. For instance, a long-lived CPU-intensive grid application can state that plenty of battery power is more important than highly available bandwidth when submitting its tasks for execution.

3.1.2 Discovery entities

The P2P discovery layer defines three main entities, which correspond to the different roles a device can play in the mobile grid, as depicted in Figure 3. *Collaborators* are available to run grid tasks. A mobile device is capable of being a collaborator after having resources registered in the middleware. *Initiators* submit application requests to collaborators for task processing. Any mobile device can be an initiator in the mobile grid at any time. Finally, *Coordinators* act in between initiators and collaborators. Coordinators broadcast initiator requests for resource discovery to collaborators and, based on received replies, inform the initiators about the most appropriate collaborators. Importantly, although our focus is on ad-hoc networks (see Figure 3(a)), a centralized coordinator could be deployed if an infrastructure network were available, thus freeing the mobile devices from additional processing overhead. Figure 3(b) illustrates this alternative scenario.

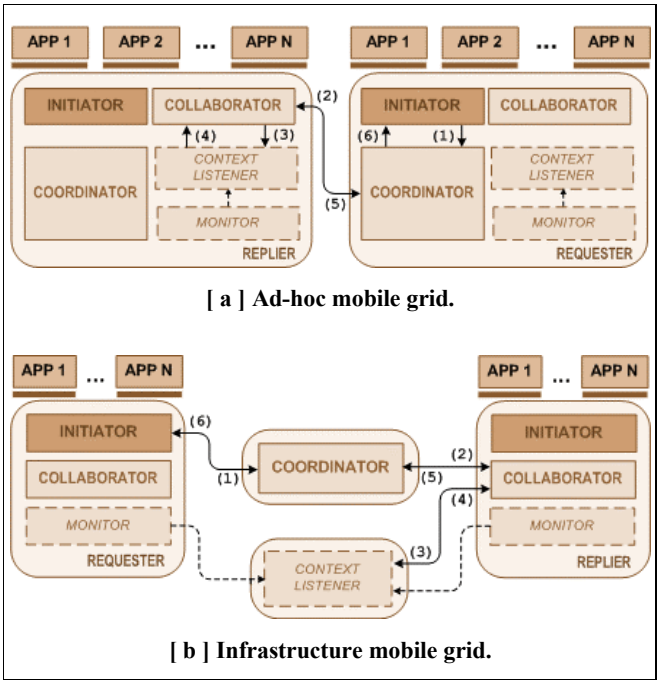

Figure 3. Discovery entities in the MoGrid architecture.

Two other underlying services play important roles in the MoGrid architecture: the *monitor* and the *context listener*. Each device in the mobile grid has a monitor resident in it. The monitor service is responsible for collecting state information from the mobile devices, including connectivity status, CPU load, remaining battery power, and available memory and storage space. In an ad-hoc mobile grid, each device also has a resident context listener (see Figure 3(a)). In contrast, an infrastructure mobile grid may have a centralized context listener (see Figure 3(b)). The context listener service periodically receives from the monitor service the collected state information, and deduces from such information the resource availability of the devices. When a coordinator queries collaborators about their resource availability to perform a grid task, the collaborators interact with the context listener service to check whether they can participate.

Basically, a collaborator uses two criteria to decide upon its participation in an initiator requested task. The first one acts like an 'admission controller', assessing whether the collaborator *is able to* provide the enquiring initiator with the required resources (*e.g.* whether the device has enough memory and battery power to execute a process). The second one defines the *suitability* of the collaborator to participate. Crucially, the suitability of a collaborator is measured according to the contextual information of interest provided by the enquiring initiator (`ctxtInfo` in operation `createRequestProfile()`). Such a measurement is used for setting up a timer that will determine how fast the collaborator will reply to the initiator request. Taking again the example of a long-lived grid application, collaborators with higher

energy levels will typically reply first to a request for process execution. Notice that, in fact, the timer associated with each reply is set to be inversely proportional to the suitability of each collaborator to participate. As aforementioned, replies are broadcast so that a collaborator having received the request can detect whether other, more resourceful collaborators have already answered it. The first replies can thus suppress additional replies, reducing the amount of P2PDP messages exchanged among devices in the mobile grid.

Coordinators handle the cases of reply collisions and request underservings. Reply collisions occur when delayed replies are unable to suppress other unnecessary replies (*e.g.* if the reply delay timers of two or more collaborators are set to similar values). As a consequence, the coordinator can receive more replies than the initiator needs. In such cases, the coordinator selects the most appropriate replies based on a certain criterion, before forwarding them to the interested initiator. Request underservings happen typically because of replies being lost. As a consequence, the coordinator can receive fewer replies than expected. In such cases, the application (or the transparency layer) that triggered the initiator request will have to deal with it, for instance by means of an additional request.

Section 4 discusses at greater length some sound criteria for defining suitability for collaborators and selecting the most 'appropriate' replies by coordinators.

3.1.3 P2P Discovery Protocol

The P2PDP protocol defines three messages, as shown in Figure 4.

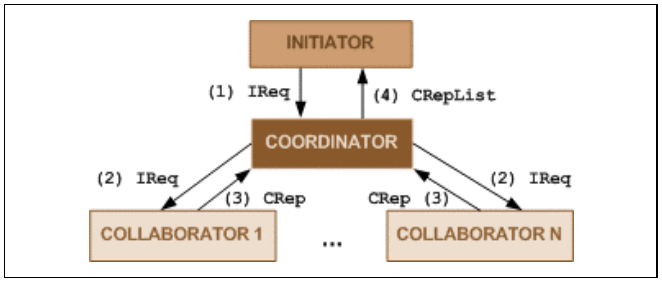

Figure 4. P2PDP messages.

`InitiatorRequest` (`IReq`) messages are sent from initiators to coordinators and forwarded by the latter to collaborators. `IReq` messages are triggered by calls to operation `discover()` in the discovery API. An `IReq` message conveys: (i) a request ID used to match requests to replies, (ii) the maximum reply delay that the enquiring initiator accepts (`maxReplyDelay`),[3] (iii) the number of collaborators that the initiator wishes to involve (`numMaxReplies`), and (iv) the contextual information that the application is interested in (`ctxtInfo`).

`CollaboratorReply` (`CRep`) messages are sent from collaborators to coordinators in response to `IReq` messages. A `CRep` message informs a coordinator about resource availability in a specific collaborator, according to the interested contextual information, as indicated in the corresponding `IReq` message. A `CRep` message also conveys a request ID matching that of the corresponding `IReq` message.

Finally, `CollaboratorReplyList` (`CRepList`) messages are sent from coordinators to initiators. A `CRepList` message conveys a list summing up selected replies from collaborators in the context of a same request. Coordinators build `CRepList` messages as follows. When a coordinator receives an `IReq` message, it inserts a new entry describing the request into its *table of pending requests*. Each entry in such a table comprises the request ID and the number of collaborators that the initiator wishes to involve in the request. Such an entry is associated with a timer that is set to the maximum reply delay the enquiring initiator accepts.[4] When such a timer expires, the coordinator summarizes all the `CRep` messages associated with the pending request that were received up to then, discarding unnecessary `CRep` messages if needed. The resulting summary is then sent to the enquiring initiator in a single `CRepList` message.

3.2 Transparency Layer

In contrast to MoGrid-tailored applications, standard applications are oblivious to the particular details of the discovery API. To address this issue, the transparency layer (see Figure 1) handles the resource coordination among collaborators using the proposed P2PDP protocol. The transparency layer is composed of two main parts: the *Transparent Resource Access Sublayer* (TRAS), and the *Adaptation Sublayers* (see Figure 5).

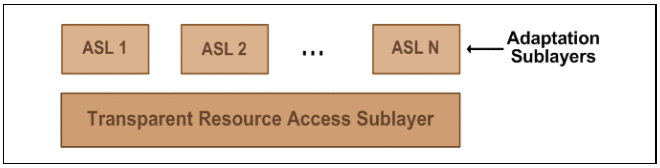

Figure 5. Transparency layer.

3.2.1 Transparent Resource Access Sublayer

The purpose of the TRAS sublayer is to mask from grid applications some issues related to the irregular connectivity that is inherent to mobile environments. Normally, upon receiving the list of the most resourceful collaborators in response to a P2PDP `IReq` message, the enquiring initiator may start the task distribution among such collaborators by using a standard grid protocol. Each collaborator, having fully performed a task, sends the obtained results back to the initiator. Nevertheless, during the task submission or execution the initiator or any of the collaborators may endure a period of disconnection, which can be either voluntary (*e.g.* device being switched off or entering 'doze' mode) or involuntary (*e.g.* abrupt loss of RF signal strength).

When the coordinator is centralized (see Figure 3(b)), it can deal with disconnected initiators by acting as a proxy for them, caching the task results sent by the collaborators. In case of disconnected collaborators, the coordinator acts on behalf of the enquiring initiator by selecting new collaborators with the help of the P2PDP protocol. When collaborator disconnections are voluntary, independent of the coordinator being centralized or distributed, mechanisms for task migration among collaborators can be used [14].

[3] This delay value can be statically defined or dynamically adjusted by an adaptive algorithm based on RTT estimations.

[4] In case of a centralized coordinator, this timer must also consider the transfer delay between initiators and the coordinator.

For decentralized coordinators (see Figure 3(a)), as is typically the case in ad-hoc networks, some additional considerations must be made. The remainder of this section focuses on the TRAS support for ad-hoc networks and its relation with the P2PDP protocol.

In case of voluntary disconnections, initiators notify current collaborators that they are going to become disconnected. When reaching the end of a task, the notified collaborators store the corresponding results until the initiator advertises that it is in service again, or a *waiting-for-initiator* timer expires, in which case the collaborators discard the cached results. When a collaborator detects that it is going to become disconnected, it interrupts the ongoing tasks and notifies all the associated initiators. In response to such notifications, initiators trigger the P2PDP protocol to select new collaborators, and then re-submit the tasks to the newly selected collaborators. Hence, such tasks will be re-executed from the beginning. Notice that another protocol (complementary to P2PDP) is needed to manage voluntary disconnections. The specification of this protocol is left for future work, since in this paper our focus is on the proposal of the MoGrid architecture, and the P2PDP protocol in particular.

For involuntary disconnections, the transparency layer relies on the events received by the context listener service to detect possible disconnections. Each adaptation sublayer may take specific actions to overcome such involuntary disconnections, as discussed in Section 3.2.2.

3.2.2 Adaptation Sublayers

The TRAS sublayer provides application-independent transparency mechanisms for resource utilization. Nevertheless, each kind of application has its specific requirements on a mobile grid. For instance, master-worker applications demand higher processing power but are less susceptible to intermittent connectivity, whereas storage capacity and connection stability have greater importance to data replication applications. The main purpose of the adaptation sublayers is to implement the handling of mobility and connectivity-related events in a way that it is more adequate for each type of grid application.

Each adaptation sublayer has to implement some callback operations that define the expected behavior of initiators in case of exception events, such as request underservings or voluntary collaborator disconnections. Crucially, the TRAS sublayer triggers a callback when any events happen. An example of adaptation sublayer implementation is presented in Section 4.

4. IMPLEMENTATION

To experiment with our approach, a middleware-architecture prototype was implemented in Java, using the Connected Device Configuration (CDC) of J2ME [19] as the reference implementation platform. The `java.net.DatagramSocket` and `java.net.DatagramPacket` classes are used to implement the P2PDP protocol over UDP. P2PDP messages exchanged between coordinators and collaborators are sent to the 'all 1s' IP broadcast address. To avoid the problem of reply implosion, we only consider single-hop ad-hoc networks in our implementation; multihop networks will be addressed in a next stage.

Collaborators and coordinators use the `java.util.Timer` and `java.util.TimerTask` classes to implement their timers. In the prototype implementation, each collaborator sets its reply delay timer to t units of time, as given by

$$\tau = \left(1 - \omega \sum_{i=1}^{N} \left(\frac{\alpha_i P_i}{\sum_{j=1}^{N} P_j} \right) \right) \frac{D_{\max}}{\gamma}, \qquad \begin{matrix} 0 \le \alpha \le 1 \\ 0 < \omega \le 1, \\ \gamma = 2 \end{matrix} \tag{1}$$

N represents the number of the different resource types the collaborator should take into account. P_i is the weight that describes the relative importance of each resource type i, $1 \le i \le N$. Both N and P_i are described as part of the contextual information sent by the initiator in the corresponding `IReq` message. $D_{\max}$ is the maximum reply delay, which is also obtained from the `IReq` message. g is a constant (arbitrarily set to 2 in our implementation) used for considering the transfer delays that `CRep` and `CRepList` messages may experience. a_i is the normalized level of availability of resource type i at the collaborator, as indicated by its monitor service. Finally, w indicates the *willingness* of the collaborator to participate in the execution of distributed tasks. It is a subjective, user-defined factor that describes the user level of interest in allowing its device to collaborate with others on the mobile grid. $w = 0$ means the collaborator is not willing to participate; thus no `CRep` messages are sent by it.

Equation (1) is quite general as it allows the definition of different criteria for determining the suitability of collaborators. For the selection of replies at coordinators, we preferred a rather straightforward criterion based on a 'first in, first selected' policy. Although simple, such a criterion allows a coordinator to build and send a `CRepList` message before the timer associated with a corresponding request expires, reducing the total time spent with the resource discovery procedure.

The implementation of the monitor service was borrowed from the MoCA architecture [17]. The MoCA monitor is deployed as a daemon running on each mobile device, being responsible for collecting state information such as RF signal strength, energy level, CPU usage, and free memory. Since MoCA monitor is implemented in C/C++, the use of the Java Native Interface (JNI) was needed to integrate the monitor into our implementation.

We have tested our prototype with a typical distributed file sharing application. For such an application, we defined the context information of interest as only comprising the battery power and the RF signal strength (*i.e.* $N = 2$), both with the same relative importance (*i.e.* $P_1 = P_2 = 1$).

The test application was implemented over an experimental adaptation sublayer. This sublayer offers three operations: `registerFile()`, `deregisterFile()`, and `getFile()`. The first two operations provide a simple interface to manage resource announcements in MoGrid. The third operation triggers a P2PDP `IReq` message for discovering only one source of the desired file.[5] The callback operation that the adaptation sublayer provides for TRAS reacts to underservings by re-enquiring the discovery service offered by the P2PDP protocol.

[5] One improvement in this sublayer implementation could involve the discovery of multiple sources. For instance, it could cache additional responses to use them in case of involuntary disconnections.

The MoCA monitor provides an *emulating mode* that allowed us to use fixed nodes mimicking the behavior of mobile devices as regards RF signal strength and battery power readings. To test the P2PDP protocol, emulation scripts were used for describing the behavior of five devices with various levels of resource availability that randomly connect and disconnect from an ad-hoc mobile grid. As an outcome of our tests, we observed the effectiveness of P2PDP in selecting more resourceful collaborators while suppressing unnecessary response messages.

5. DISCUSSION

This work corresponds to an initial stage in the effort to develop a fully-fledged mobile grid middleware architecture. A central point that we envision as needing further developments is the description of resources. Currently, there is no standardized support for such descriptions in our implementation, which are left completely to the application. To address this issue, we plan to use an (or define a new) XML-based resource description language similar to those applied at OGSA [3] and WSDA [8]. Such a description language would allow the use of XML-targeted technologies like XQuery and XPath as inputs to the operation `discover()` of our discovery API.

Other future work will involve the implementation of proxies that allow devices in the mobile grid to interoperate with devices in conventional grids. We are starting to investigate such integration through the use of Globus components, as part of the InteGridade project [10].

A key aspect not covered in our implementation is privacy and trust establishment. Social relationships based on reputation mechanisms have constantly been pointed out [13, 16] as a possible solution to this problem. We are currently examining the use of w in Equation (1) as a parameter that aids in describing the collaborators reputation in a mobile grid.

6. CONCLUSION

In this paper we have proposed a middleware architecture that allows the distribution of tasks among devices in a mobile grid. Central to the proposed architecture is a resource discovery protocol called P2PDP. Fundamentally, P2PDP helps in coordinating such a distribution among the most resourceful and available mobile devices, while mitigating the overhead of control messages exchanged among them. We have implemented a prototype of the architecture and also a simple test application to evaluate the correctness of the protocol and to trial some main features of the design, such as the criteria for collaboration suitability.

We are aware of the need for conducting more comprehensive experiments to evaluate the performance and scalability of our approach. This includes evaluating the behavior of the P2PDP protocol and the services provided by the transparency layer when they are faced with different mobility patterns, including periods of voluntary or involuntary disconnections. One specific problem that we plan to tackle in the context of the transparency layer is the migration of tasks among collaborators, for example when a device infers that it is losing connectivity with a mobile grid.

7. REFERENCES

[1] AKOGRIMO Project, http://www.akogrimo.org/ (2005).

[2] Bhatia, K. *Peer-To-Peer Requirements On The Open Grid Services Architecture Framework*. GFD.49, GGF Document Series, OGSAP2P Research Group, 2005.

[3] Foster, I., Kesselman, C., Nick, J.M., and Tuecke, S. *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*. Globus Project, 2002.

[4] Fox, G., Gannon, D., Ko, S., Lee, S., Pallickara, S., Pierce, M., Qiu, X., Rao, X., Uyar, A., Wang, M., and Wu, W. Peer-to-peer Grids. In *Berman, F., Fox, G., Hey, T., eds.: Grid Computing - Making the Global Infrastructure a Reality*, 471-490. John Wiley & Sons Ltd, 2003.

[5] GIGA Project, http://www.projetogiga.org.br/ (2005).

[6] Globus Project, http://www.globus.org/ (2005).

[7] Gutwin, C., Greenberg, S., and Roseman, M. Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation. In *Proc. of HCI'96: People and Computers XI*, 281-298, London, UK, 1996.

[8] Hoschek, W. The Web Services Discovery Architecture. In *Proc. of the Int'l IEEE/ACM Supercomputing Conference*, 1-15, Baltimore, USA, 2002.

[9] Hwang, J., and Aravamudham, P. Middleware Services for P2P Computing in Wireless Grid Networks. *IEEE Internet Computing*, vol. 8, nº. 4, 40-46, 2004.

[10] InteGridade Project, http://integridade.lncc.br/ (2005)

[11] K*Grid Project, http://gridcenter.or.kr/ (2005).

[12] Kurkovsky, S., Bhagyavati, and Ray, A. Modeling a Grid-Based Problem-Solving Environment for Mobile Devices. In *Proc. of the IEEE Int'l Conference on Information Technology: Coding and Computing (ITCC-04)*, Las Vegas, USA, 2004.

[13] McKnight, L.W., Howison, J., and Bradner, S. Wireless grids: Distributed resource sharing by mobile, nomadic, and fixed devices. *IEEE Internet Computing*, vol. 8, nº. 4, 24-31, 2004.

[14] Milojicic, D.S., Douglis, F., Paindaveine, Y., Wheeler, R. and Zhou, S. Process Migration. *ACM Computing Surveys*, vol. 32, nº. 3, 241-299, 2000.

[15] P-Grid Project, http://www.p-grid.org/ (2005).

[16] Rheingold, H. *Smart Mobs: the next social revolution*, Perseus Publishing, 2003.

[17] Sacramento, V., Endler, M., Rubinsztejn, H.K., Lima, L.S., Gonçalves, K., Nascimento, F.N, and Bueno, G.A. MoCA: A Middleware for Developing Collaborative Applications for Mobile Users. *IEEE Distributed Systems Online*, vol. 5, nº. 10, 2004.

[18] Schiller, J. Mobile Communications, Addison Wesley, 2003.

[19] Sun Microsystems. *Java 2 Platform*, Micro Edition. http://java.sun.com/j2me/ (2005).

[20] Yamin, A.C., Barbosa, J.B., Augustin, I., Silva, L.C., Real, R., Geyer, C., and Cavalheiro, G. Towards Merging Context-Aware, Mobile and Grid Computing. *Int'l Journal of High Performance Computing Applications (jHPCA)*, vol. 17, nº. 2, 191-203, 2003.