Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

An Event Service for Pervasive Grids

G. Della Vecchia – A. Coronato – M. Ciampi

RT-ICAR-NA-06-06 **03-2006**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

An Event Service for Pervasive Grids[1]

G. Della Vecchia[2] – A. Coronato[3] – M. Ciampi[2]

Rapporto Tecnico N.: **RT-ICAR-NA-06-06**

Data: **03-2006**

---

[1] Sottoposto per pubblicazione

[2] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Napoli, Via P. Castellino 111, 80131 Napoli
[3] Sviluppo ed Applicazione dei Sistemi Informativi Territoriali, SASIT-CNR, Sede di Napoli, Via P. Castellino 111, 80131 Napoli

An Event Service for Pervasive Grids

Mario Ciampi[1], Antonio Coronato[2], Gennaro Della Vecchia[1]

[1]ICAR-CNR, Via Castellino 111, 80131 Napoli, Italy
{mario.ciampi,gennaro.dellavecchia @na.icar.cnr.it}
[2]SASIT-CNR, Via Castellino 111, 80131 Napoli, Italy
{coronato.a@na.drr.cnr.it}

Abstract. Event driven paradigms have become the leading model for large classes of applications and environments. In particular, pervasive, ubiquitous, and grid computing, in which services dynamically plug in and get out, rely on middleware services that use communication channels and events to integrate and coordinate application level services. This work presents an advanced asynchronous communication service, which is a core component of a middleware, under construction, for pervasive grid environments. Such a service handles hierarchies of structured classes of events. In particular, events are grouped in hierarchies of classes. Producers and consumers can subscribe the communication service for one or more classes; i.e., a consumer that subscribes a class receives all the events of that class and of any other hierarchically upper. The implemented model relies on well established, structured paradigms as the producer/consumer model, the message passing model, and the heredity technique.

1. Introduction

During the last decade, new computing models have emerged and affirmed. In particular, terms like Grid Computing and Pervasive Computing have become of common use not only in the scientific and academic world, but also in commercial and business fields. The Grid computing model has emerged and rapidly affirmed as the new

computing paradigm for large-scale resource sharing and high-performance distributed applications. The term "The Grid" was primarily introduced by Foster and Kesselman to indicate a distributed computing infrastructure for advanced science and engineering [1]. Successively, it has been extended to denote the virtualization of distributed computing and data resources such as processing, network bandwidth and storage capacity to create a single system image, granting users and applications seamless access to vast IT capabilities. As a result, Grids are geographically distributed environments, equipped with shared heterogeneous services and resources accessible by users and applications to solve complex computational problems and to access to big storage spaces. Recently, Grid computing environments are being extended in order to present some characteristics that are typically found in pervasive computing environments.

Differently, the goal for Pervasive Computing is the development of environments where highly heterogeneous hardware and software components can seamlessly and spontaneously interoperate, in order to provide a variety of services to users independently of the specific characteristics of the environment and of the client devices [2]. Therefore, mobile devices should come into the environment in a natural way, as their owner moves, and transparently, that is owner will not have to carry out manual configuration operations for being able to approach the services and the resources, and the environment has to be able to self-adapt and self-configure in order to host incoming mobile devices.

The conjunction of the two paradigms is leading towards Pervasive Grid environments [3]. A key feature for Pervasive Grids is the possibility of making mobile users able to get access to the Grid and to move both among different areas within the same Grid environment and among different Grid environments.

Such kind of environments are definitively highly dynamic, that is users as well as services can be integrated on-the-fly in the environment. This requires the environment be able to handle, compose and coordinate services on-demand, depending on user's needs and context. In other words, such characteristics call for advanced middleware services, which must confer to the environment autonomic behaviours as self-management, self-configuration, and self-adaptation.

The programming paradigm that best fit emerging needs is the event-driven model. In this case, applications and services are driven by

events generated by the environment or components of it. This enables to decouple services each other.

This work describes an asynchronous communication broker that handles structured classes of events. The proposed model allows to hierarchically classify events and enables producers and consumers to subscribe specific classes of events. As a result, not only consumers and producers are decoupled, but also, consumers receive only events of the classes that they have subscribed.

The rest of this work is organized as follows. Section 2 discusses some motivations and related works. Section 3 describes the proposed model for handling structured classes of events and the architecture of the asynchronous communication broker, which supports such a model. In section 4 a case study is reported. Finally, section 5 concludes the work.

2. Motivations and related works

Current Grid architectures do not take into account the mobile computing environment since mobile devices have not been considered as valid resources or interfaces in Grid community. However, in the last few years mobile devices have substantially been enhanced and have got a large diffusion in the market. As a consequence, they can no longer be discriminated by Grid community. Differently, they can be effectively incorporated into the Grid either as service consumer or as service provider [4]. Such new potentialities have been attracting the interest of several researchers. We cite some remarkable works that face some aspects related to combining Grid and Mobile computing [5,6,7].

In [5] new challenges stemming from implicit requirements are outlined. In particular, authors emphasize some characteristics of mobile devices and their impact in the Grid environment.

In [6] a new scheduling algorithm for Mobile Grid environments have been proposed. Such a algorithm takes into account the intermittent connectivity of mobile nodes, which are interfaced with the Grid by specific proxy components.

In [7] mobile devices are considered as active resources for the Grid. In particular, authors propose an architecture for deploying Grid services on mobile nodes by making them active actors in the Grid.

In addition to this, in pervasive grid environments mobile users can dynamically enter in a physical site and exit by leaving pending computations. This calls for advanced session management mechanisms. Moreover, every time a mobile user enters in a site, the environment has to locate him and to provide him with the list of services available at that location. Such a list may also depend on hardware characteristics of the mobile device.

In both fields, pervasive computing and grid computing, specific needs have been faced by developing middleware frameworks. Most of them are event oriented middleware infrastructures, which rely on event channels in order to enabling components to communicate each other. For examples, we report GAIA [8] and the Globus Toolkit [9].

GAIA is a middleware infrastructure built at the Department of Computer Science at the University of Illinois. It defines the concepts of physical space and active space. A physical space is an area that contains objects like devices and resources. An active space is a physical space enriched by a software infrastructure that changes its behavior depending on context and enables interactivity among users and the environment. The GAIA kernel includes an event manager that receives and dispatches events in the active space. It is important to note that with this approach a consumer subscribes the channel and receives every event dispatched by it. Contrary, in order to restrict the set of events received by a consumer, distinct channels should be available. In other words, GAIA doesn't support direct mechanisms for classifying and handling hierarchies of events.

The Globus Toolkit is the most diffused middleware infrastructure for grid applications. Latest version of the Globus Toolkit implements part of the WSNotification specifications [10,11]. Such specifications define an interaction model based on the publish/subscribe paradigm; i.e., a producer publishes a topic (class of events) and consumers can subscribe it in order to get notification of events of that class. By this way, a direct link among producers and consumers is established. As a consequence, consumers and producers are not decoupled, but consumers have to know which are the producers of any class of events. Other problems arise when events of a class can be produced by more sources: consumers have to know and subscribe every possible producer. In addition to this, it is not possible to handle hierarchies of classes of events. WSNotification specifications have been enhanced by the WSBrokeredNotification [12] specifications that introduce the

concept of broker as mediator between producers and consumers. This allows to decouple producers and consumers. However, such specifications are not supported by the Globus Toolkit yet and, more important, they do not take into account hierarchies of classes of events.

In this work, we present a model that extends the one proposed by the WSBrokeredNotification specifications and implemented it as a Grid Service that is part of a middleware we are developing for Pervasive Grid environments. The service is compliant to the OGSA standard [13] and can be deployed over the Globus Toolkit 3.2 platforms, or later.

3. The Asynchronous Communication Broker

A model for defining and handling structured classes of events

In a complex environment like a pervasive grid, events can be grouped in classes, which can also be related each other and constitute hierarchies.

To clarify such a concept, figure 1 can help understanding. In the part a of the figure, events are classified and represented by sets as follows:

- Class A = B U C U D U (E_A1; E_A2)= (E_A1; E_A2; E_B1; E_C1; E_C2; E_D1)
- Class B = C U (E_B1) = (E_B1; E_C1; E_C2)
- Class C = (E_C1; E_C2)
- Class D = (E_D1)

An alternative representation is proposed in the part b of the figure. This is more interesting for our aim because it allows to focus on hierarchical properties of the classes of the events. In particular, for an example, a consumer that subscribes for class D will receive only the event E_D1. Differently, a consumer that subscribes for class B will receive every events E_B1, E_C1, and E_C2.

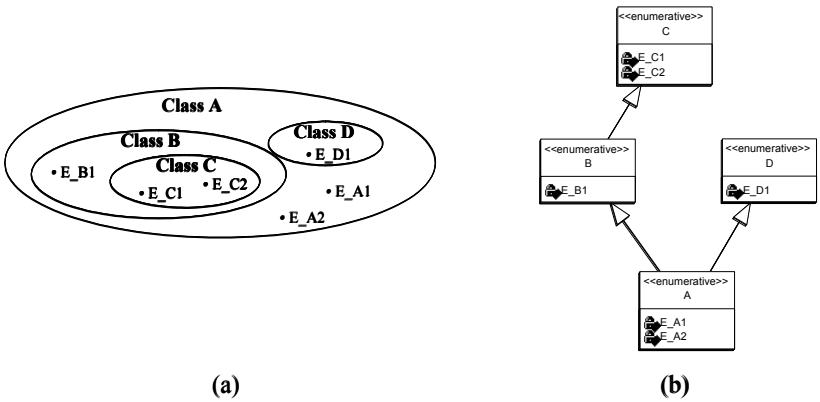

Figure 1 – Representations of hierarchies of classes of events

Architecture of the service

The Asynchronous communication service handles hierarchies of classes of events. In particular, it implements and extends (with the model described previously) the WSBrokeredNotification specifications. Figure 2 shows the following services functionalities:

- Publish Topic – This functionality allows producers to publish topics, that is classes of events. When a new topic is published, it can extend other already published topics. By this way, hierarchies of classes can be defined.
- Unpublish Topic – This functionality is invoked by a producer to communicate that it will no longer be a producer for that class of events. If there isn't any other producer for the topic, the service remove the class of events from the list of available topics.
- Notify Event – This functionality is required by the producer to communicate a change of state, i.e. an event, to the Broker. Consequently, the Broker dispatches the event to every consumer that has subscribed the class which contains the event, or a class hierarchically related to that one.

- Verify Topic – This functionality enables producers and consumers to verify weather a topic is active or not.
- Browse Topics – This functionality enables producers and consumers to browse classes of events and hierarchies of classes.
- Subscribe Topic – This functionality allows consumers to subscribe a class of events. When a class is subscribed, the consumer automatically subscribes every class hierarchically upper.
- Unsubscribe Topic – This functionality is invoked by a consumer to unsubscribe a class (and every other hierarchically upper) of events.

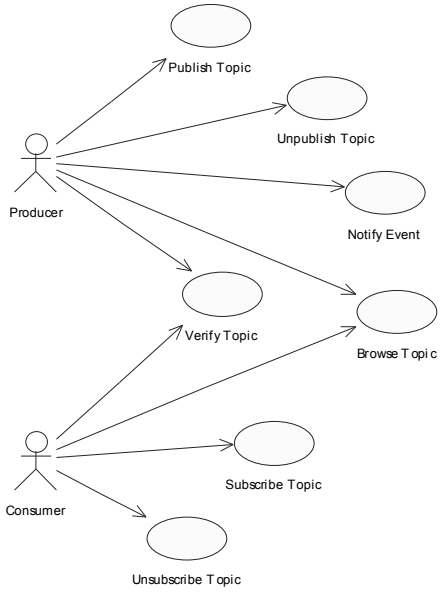

Figure 2 – Service use cases

The internal architecture of the service is shown in figure 3.

In order to become a producer, a service active in the environment has to publish a topic and to indicate the related hierarchy (if any). If the topic is not already present in the list of available topics, it is inserted in by the *CommunicationChannel*. After that, the *CommunicationChannel* creates the *WaitForEvents* thread, which is in

charge of receiving events from the producer. When an event is received, the *CommunicationChannel* creates one *NotifyEvent* thread for each consumer that has subscribed that class, or any other hierarchically lower.

The service has been developed as a Grid Service and deployed over a Globus Toolkit 4.0 platform. It is an OGSA compliant component.

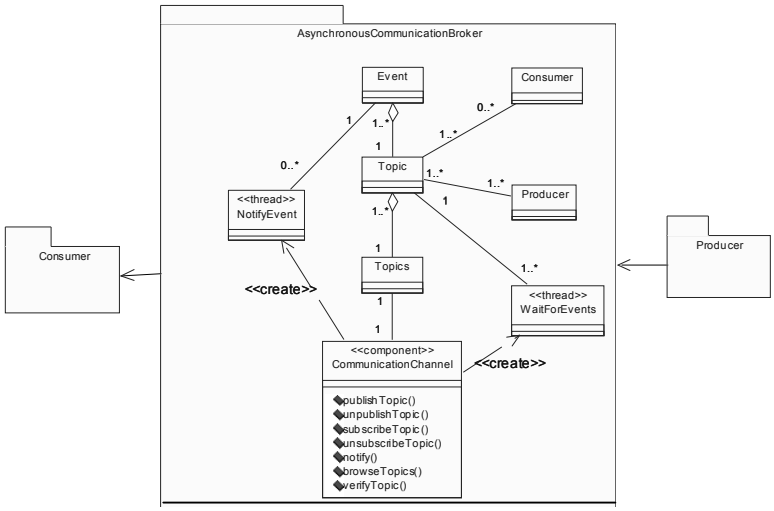

Figure 3 – Service architecture

4. An applicative scenario

This section presents an application scenario in which some basis middleware services interacts each other for supporting a pervasive grid environment. In particular, middleware services are:

- *LocationService* – This service locates mobile devices in the environment. It is in charge of communicating to the environment i) incoming devices, ii) location changes for active mobile devices, and iii) outcoming devices.

- *PersonalAgentService* – This component is in charge of interfacing with a mobile device and handling its requests.
- *SessionManagerService* – This service handles sessions for mobile devices. It creates a *PersonalAgent* as soon as a new device comes in the environment. When a mobile device changes its location, the *SessionManagerService* updates the list of services, which depends on the user location, available for that device. Finally, when a mobile device exits the environment, the *SessionManagerService* destroys its *PersonalAgent*.
- *DeviceService* – This service handles the list of mobile devices active in the environment.

For this scenario, three classes of events can be defined:

- Class MOBILITY = (NEW_LOCATION)
- Class PRESENCE = (NEW_DEVICE, DEVICE_HAS_LEFT)
- Class LOCALIZATION = MOBILITY U PRESENCE;

As shown in figure 4, the *LocationService* publishes the topic LOCALIZATION, that is it can produce events like NEW_LOCATION, NEW_DEVICE, and DEVICE_HAS_LEFT. The *SessionManagerService* subscribes such a topic, whereas the *PersonalAgentService* and the *DeviceService* respectively subscribe the MOBILITY and the PRESENCE topics.

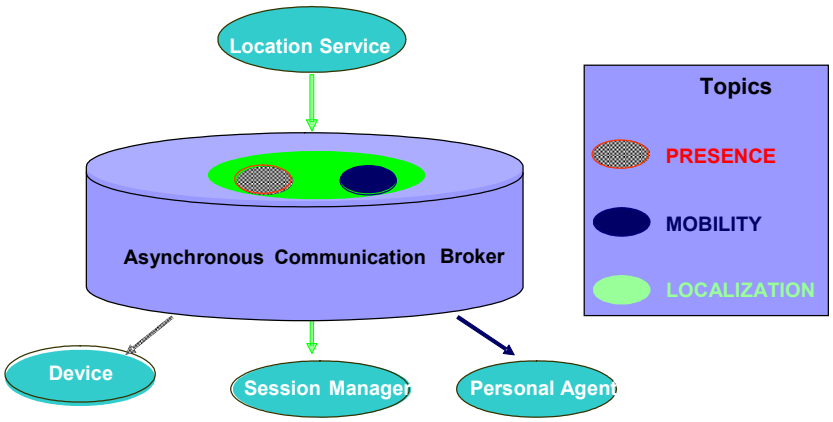

Figure 4 – Applicative scenario

5. Conclusions

This work has described an asynchronous communication service for pervasive grid environments. The service implements a communication model that enables to handle hierarchies of events. The models that we propose is demonstrating to effectively handle events and hierarchies of classes of events in pervasive grid environments.

References

[1] I. Foster, C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure". Morgan Kaufmann, 1999.

[2] D. Saha and A. Murkrjee, "Pervasive Computing: A Paradigm for the 21st Century", IEEE Computer, March 2003.

[3] V. Hingne, A. Joshi, T. Finin, H. Kargupta, E. Houstis, "Towards a Pervasive Grid", International Parallel and Distributed Processing Symposium, IPDPS 2003.

[4] B. Clarke and M. Humphrey, "Beyond the 'Device as Portal': Meeting the Requirements of Wireless and Mobile Devices in the Legion of Grid Computing System", International Parallel and Distributed Processing Symposium, IPDPS 2002.

[5] T. Phan, L. Huang and C. Dulan, "Challenge: Integrating Mobile Devices Into the Computational Grid", International Conference on Mobile Computing and Networking, MobiCom 2002.

[6] S. M. Park, Y. B. Ko and J. H. Kim, "Disconnected Operation Service in Mobile Grid Computing", International Conference on Service Oriented Computing, ICSOC 2003.

[7] D. C. Chu and M. Humphrey, "Mobile OGSI.NET: Grid Computing on Mobile Devices", International Workshop on Grid Computing, GRID 2004.

[8] C. Hess, M. Roman, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrsted, "A Middleware Infrastructure for

Active Spaces", IEEE Pervasive Computing, October-December 2002.

[9] http://www.globus.org

[10] http://www.oasis-open.org/committees/download.php/13488/wsn-ws-base_notification-1.3-spec-pr-01.pdf

[11] http://docs.oasis-open.org/wsn/2004/06/wsn-WS-Topics-1.2-draft-01.pdf

[12] http://www.oasis-open.org/committees/download.php/13485/wsn-ws-brokered_notification-1.3-spec-pr-01.pdf

[13] I. Foster, C. Kesselman, J. Nick, S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.