A Web-Services Based Architecture for Dynamic-Service Deployment

Christos Chrysoulas[1], Evangelos Haleplidis[1], Robert Haas[2], Spyros Denazis[1,3], and Odysseas Koufopavlou[1]

[1] University of Patras, ECE Department, Patras, Greece
{cchrys,ehalep,odysseas}@ee.upatras.gr
[2] IBM Research, Zurich Research Laboratory, Rüschlikon, Switzerland
rha@zurich.ibm.com
[3] Hitachi Sophia Antipolis Lab, France
Spyros.Denazis@hitachi-eu.com

Abstract. Due to the increase in both heterogeneity and complexity in today's networking systems, there arises a demand for an architecture for network-based services, that gives flexibility and efficiency in the definition, deployment and execution of the services and at the same time, takes care of the adaptability and evolution of such services. In this paper we present an approach that applies a component model to GT4, a Web-service based Grid environment, which enables the provision of parallel applications as QoS-aware (Grid) services, whose performance characteristics may be dynamically negotiated between a client application and service providers. Our component model allows context dependencies to be explicitly expressed and dynamically managed with respect to the hosting environment, computational resources, as well as dependencies on other components. Our work can be seen as a first step towards a component-based programming-model for service–oriented infrastructures utilizing standard Web services technologies.

1 Introduction

In the recent years, Web service technology has gained more and more importance in the area of Grid Computing. The Open Grid Service Architecture [1] has motivated Grid architects to build environments based on a service-oriented architecture utilizing Web-service technology. The evolution of the Globus Toolkit 4 [2] towards the Web Service Resource Framework [3] was the outcome of that effort.

Grids are mostly built following a service-oriented architecture using Web-services technology which has not been designed to fit the idea of a component-based plug-and-play client programming framework. Services are typically discovered dynamically, using technologies like the Monitoring and Discovery System (MDS) [4] in Globus Toolkit 4, rather than created, they further do not provide means to describe

D. Hutchison et al. (Eds.): IWAN 2005, LNCS 4388, pp. 206–211, 2009.

dependencies for example in other services running outside the Grid. Web-service technology provides a versatile messaging facility but lacks an extensive component model applicable to service composition. In this paper we present a component-based architecture in order to address the above issues.

Our architecture is based not only on the Globus Toolkit 4 environment, a Grid architecture for the provision of parallel applications as Grid services over standard Web-service technology, but it also makes heavy use of a component-based architecture trying to solve the problem of creating new services and the dependencies in other services and components outside the architecture we present.

Our proposed Dynamic Service-Deployment Architecture is developing as part of the FlexiNET [5] IST research project and particularly as a sub-module of the FlexiNET Wireless Access Node (FWAN) module.

The remainder of the paper is organized as follows: Section 2 describes what FlexiNET is. Section 3 describes the architecture regarding the Dynamic Service-Deployment. A discussion on related work is given in section 4. Conclusions and future work are presented in section 5.

2 FlexiNET Architecture

As stated in section I, the DSD module is developed for the FlexiNET Project. The primary aim of the project is to define and implement a scalable and modular network architecture incorporating adequate network elements (FlexiNET Node Instances) offering roaming connection control, switching/routing control, and advanced services management/access functions to the network access points that currently only support connectivity between user terminals and network core infrastructures [6], [7], [8].

The FlexiNET network architecture consists mainly of node instances, communication buses and data repositories.

The DSD module is part of the FWAN. The FWAN architecture can be seen in Figure 1 and is based on Hitachi's distributed router. Hitachi's distributed router consists of two functional blocks, the basic and the extensible function block.

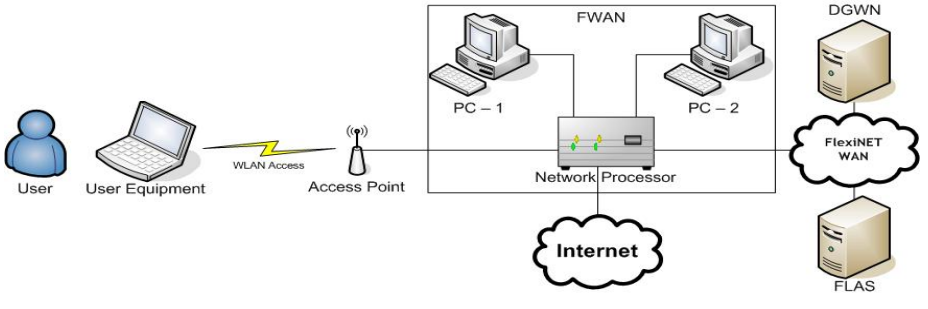

Fig. 1. FWAN architecture

The FWAN has, as a basic functional block, a network processor, and as extended functional block, two PCs. A user will access the FWAN through an access point using either a laptop or a mobile phone. The FWAN is responsible for authenticating native and roaming users through the FLAS using an AAA proxy.

The Dynamic Service Deployment Module (DSD) must be deployed on the FWAN before boot-up. The Bootstrap Process is responsible for booting up the FWAN with the AAA proxy module. In order to accomplish its task, it reads from a static configuration file which is stored in a local of the Bootstrap Process, followed by a series of commands which will be sent to the DSD Module in order to create the FWAN's node fundamental functionalities. The Bootstrap Process mainly will trigger the install of the AAA proxy through the DSD module.

The AAA Proxy Module is forwarding the Authentication packets to the FLAS Server, encapsulates the EAP Packets [9] into XML messages that are passed over Web Services, and the opposite, in order to authenticate and authorize the user. The AAA proxy service is deployed in the FWAN at boot-up time. It is stored in a local directory and deployed by the DSD module. The code will be requested from the DGWN through Web services.

On boot-up the DSD module is requested by the Boot-up process to deploy the AAA proxy module. The DSD module retrieves through the DGWN the AAA proxy service code and deploys it on any of the two PCs based on specific algorithms. Also based upon the user profiles, the DSD module will deploy a Quality of Service Module (QoS), which is responsible for providing QoS to specific users. The required configuration of the network processor will be handled by the ForCEG module which receives Web Service requests from the AAA Proxy and the QoS Module and translates them into ForCES protocol messages [10].

3 DSD Architecture

3.1 DSD Definition

By Dynamic Service Deployment we are referring to a sequence of steps that must be taken in order to deploy a service on demand. The necessary steps regarding the service deployment refer to service code retrieval, code installing destination according to matchmaking algorithms, and service deployment. The matchmaking algorithms provide the most efficient use of system resources by examining the available resources of the FWAN with the required resources of the service to be deployed.

3.2 Proposed DSD Architecture

The following figure depicts the current proposed DSD architecture. As can be deduced from the figure the DSD is the sum of the following sub-components:

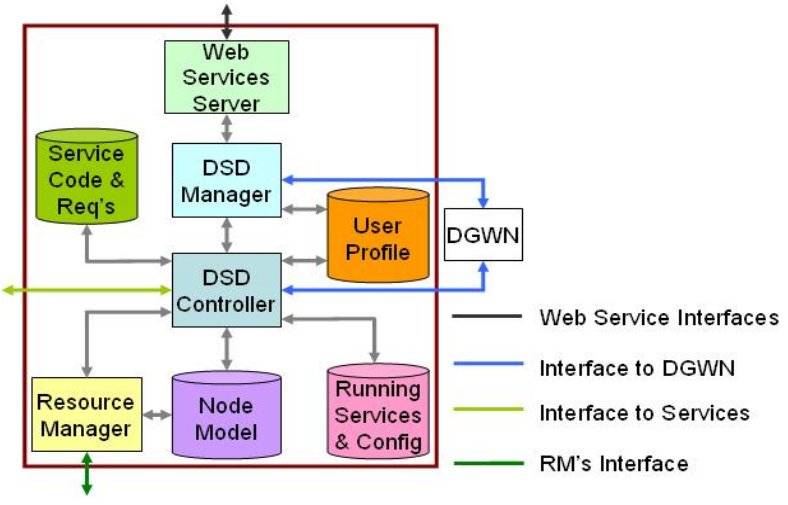

Fig. 2. DSD Architecture

Web Services Server
The Web Services Server sub-component hosts the interfaces with the AAA Proxy and the Bootstrap Process. At this stage only these two processes will interact with the DSD Module. This server is responsible for exchanging messages between the DSD Module and the AAA Proxy Module and the Bootstrap Process. The Web Services Server sub-component has the necessary functionalities necessary to register a Web Service in a UDDI directory. This component also is capable of finding other Web Service Interfaces.

DSD Manager
The DSD Manager sub-component has two functions depending on whether the user's profile is required:

- In the case of the AAA Proxy communicates with the DGWN, the DSD Manager must download the user profile, in order to find, which services must be deployed , and provides the request to the DSD Controller.
- In the case of Bootstrap Process, the DSD Manager passes the bootstrap services required for deployment to the DSD Controller

The DSD Manager is responsible to check if a user has terminated the connection and undo the user's personal configuration.

DSD Controller
The DSD Controller sub-component is assigned to receive the service request from the DSD Manager, to communicate with the DGWN in order to download the service code and the service requirements, to retrieve from the Node Model the available resources, to perform the Matchmaking Algorithm in order to find the most suitable resources, and finally to deploy the service. The DSD Controller is responsible for the Services, in 3 dimensions: Download, Deploy, and Configure.

Resource Manager
The Resource Manager sub-component is assigned to do the discovery and monitoring of the resources. It collects information, with the help of the Resource Manager Interface, from all the components of the Node model, and also from the DSD Controller. All the collected information is available to the rest of the sub-components through the WebMDS Interface it provides. Only the necessary information is passed to the Node Model.

Node Model
The Node Model is responsible for keeping all the information about FWAN. It provides us with a complete view regarding the FWAN. The node model contains information regarding physical resources available and used, and data about running services.

User Profile
User Profile is the data–storage where the downloaded User Profile is stored. It is responsible for keeping the User Profile.

Service code and Requirements
The Service Code and Requirements data-storage is responsible for storing the downloaded code and the requirements (in terms of physical resources) that describe a service.

Running Services and Configuration
The Running Services and Configuration data-storage is responsible for storing data about running services and their current configuration.

4 Discussion of Related Work

Distributed component models such as Cobra [11], DCOM [12] are widely used mostly in the context of commercial applications. The Common Component Architecture developed within the CCA forum [13] defines a component model for high-performance computing based on interface definitions. XCAT3 [14] is a distributed framework that accesses Grid services e.g. OGSI [1] based on CCA mechanisms. It uses XSOAP for communication and can use GRAM [2] for remote component instantiation. Vienna Grid Environment [15] is a Web service oriented environment for supporting High Performance Computing applications. Vienna Grid Environment (VGE) has been realized based on state-of-the-art Grid and Web Services technologies, Java and XML. Globus Toolkit 4 [2] is an environment that mostly deals with the discovery of services and resources in a distributed environment rather than the deployment of the services themselves.

5 Conclusion and Future Work

We presented an architecture that adds a dynamic perspective to Web service based Grid Infrastructure. Our component-based model is addressing the issue regarding the

dynamic deployment of new services in a distributed environment and the way they address themselves in that environment. We expect this work to be not only relevant to the Grid community but also to the Web service and the Network communities as we did not only address concerns related to Grid computing but also discussed architectural issues regarding Web service configuration and deployment. Our implementation of the model is still in prototype stage which requires further refinement and analysis. For future work we plan to provide a more sophisticated model for service deployment and selection based on QoS properties.

References

1. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Globus Project (2002), http://www.globus.org/research/papers/ogsa.pdf
2. The Globus Alliance, http://www.globus.org
3. The Web Service Resource Framework, http://www.globus.org/wsrf/
4. The Globus Alliance, http://www.globus.org/toolkit/docs/development/3.9.4/info/wsmds.html
5. FP6-IST1 507646 FlexiNET Technical Annex
6. FP6-IST1 507646 FlexiNET D21 Requirement, Scenarios and Initial FlexiNET Architecture
7. FP6-IST1 507646 FlexiNET D22 Final FlexiNET Network Architecture and Specifications
8. Aladros, R.L., Kavadias, C.D., Tombros, S., Denazis, S., Kostopoulos, G., Soler, J., Haas, R., Dessiniotis, C., Winter, E.: FlexiNET: Flexible Network Architecture for Enhanced Access Network Services and Applications. In: IST Mobile & Wireless Communications Summit 2005, Dresden (2005)
9. RFC 3748: Extensible Authentication Protocol (EAP) (June 2004)
10. Haleplidis, E., Haas, R., Denazis, S., Koufopavlou, O.: A Web Service- and ForCES-based Programmable Router Architecture. In: IWAN 2005, France (2005)
11. CORBA Component Model, v3.0, OMG, http://www.omg.org/technology/documents/formal/components.htm
12. COM Component Object Model Technologies, Microsoft, http://www.microsoft.com/com/default.mspx
13. The CCA Forum, http://cca-forum.org/
14. Krishnan, S., Gannon, D.: XCAT3: A Framework for CCA Components as OGSA Services. In: Proceedings of the Ninth International Workshop on High-Level Parallel Programming Models and Supportive Environments, pp. 90–97 (April 2004)
15. Benkner, S., Brandic, I., Engelbrecht, G., Schmidt, R.: VGE - A Service-Oriented Environment for On-Demand Supercomputing. In: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (Grid 2004), Pittsburgh, PA, USA (November 2004)