DYNAMIC DEPLOYMENT OF SEMANTIC-BASED SERVICES IN A HIGHLY DISTRIBUTED ENVIRONMENT

Christos E. Chrysoulas and Odysseas Koufopavlou
Electrical and Computer Engineering Department of University of Patras, Rio, 26500, Greece

Keywords: Semantic Service Discovery, Semantic Web Services, Node Model, Dynamic Service Deployment.

Abstract: Today's Networking Systems tend to increase in both heterogeneity and complexity. So there arises the need for an architecture for network-based services that provides flexibility and efficiency in the definition, deployment and execution of the services In this paper we present an approach that applies a Semantic-based Service deployment framework, which enables the provision of parallel applications as QoS-aware, 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, and dependencies on other components.

1 INTRODUCTION

Network and service management fields find themselves nowadays at crossroads with middleware technologies, new network architectures and emerging research directions. Middleware technologies like web services have reached maturity enjoying wide deployment and adoption. Network architectures and infrastructures built for different purposes are on their way towards IP convergence giving rise to new integrated and more complex architectures. Finally, recent ambitious research directions like autonomic computing and communications have already made a dynamic appearance in the networking community raising the challenges even higher.

This activity has coincided with the end of a period in network and service management during which vast experience and lessons learned have been accumulated based on what constitutes the past state of the art in telecommunications and in data networks, realized by many as CORBA-based distributed management platforms and SNMP-based platforms, respectively.

This encounter in the making already produces speculation and activity about redefining/reassessing the initial requirements that drove the developments in network and service management during the past period and about the "shape" of management when projected into the future.

The most prominent decentralized management approaches are based on distributed object technologies as CORBA (Common Object Request Broker Architecture) (Object Management Group, 2006) and Java RMI (Remote Method Invocation) (Jae-Oh L, 2000). According to these paradigms, information can be gathered from any location based on invocations of distributed remote objects on the target network element. The afore- mentioned distributed object technologies allow management operations to be performed on simple service-oriented APIs. On the downside they are resource expensive for large object populations, resulting in suboptimal object retrieval.

Apart from decentralized approaches based on agents or distributed object technologies, there are also approaches offering management capabilities over the Web. Approaches that are based on the Common Information Model (CIM), which encompass a set of common management operations (Distributed Management Task Force, 2006).

As network infrastructure is shifting towards service-centric networks a number of architectural characteristics are likely to influence management operations and functionality and dictate the specific choices of technologies that realize thereof. To our view, three of such characteristics are going to play a crucial role in the coming years:

E. Chrysoulas C. and Koufopavlou O. (2007).
DYNAMIC DEPLOYMENT OF SEMANTIC-BASED SERVICES IN A HIGHLY DISTRIBUTED ENVIRONMENT.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Web Interfaces and Applications*, pages 416-419
DOI: 10.5220/0001278504160419

1. Federated Network Architectures

In an effort to provide seamless end-to-end connectivity that meets customer demands, networks/service providers have started forming federations of networks wherein a number of operations like AAA, monitoring and SLA support are treated in homogeneous way in a heterogeneous environment.

2. Network architectures with distinct separation of concerns

The most representative example is the separation of control from the forwarding plane (Yang L et al, 2005), which allows the two to evolve separately of each other. The binding element between the two is a set of open interfaces that abstract functionality and allow access to functionality and resources that are vendor independent.

3. Distributed Network Node Architectures

Individual network nodes and other devices are clustered together in order to form more complex and extensible distributed architectures that operate as one integrated node. Such constellations provide the means of adding resources as needed and foster dynamic service deployment, namely, injecting new functionality in the network.

In such context, management faces a number of challenges originating from the increasing complexity and size of networks, the heterogeneity of devices and technologies that must coexist and the high degrees of flexibility required in services. This has been primarily the motivation of our research presented in this paper, which touches upon these issues by exploring potential solutions on the service deployment and resource management within a network architecture.

We have based our designs on web services as the 'de-facto' omnipresent standard technology in networks with high integration capability and one of the most promising approaches to future management technologies.

The remainder of the paper is organized as follows: Section 2 describes in depth the proposed architecture regarding the dynamic service deployment and a user-case scenario of the deployment of a new service is also presented. Conclusions and future work are presented in Section 3.

2 DYNAMIC DEPLOYMENT OF SERVICES

2.1 Dynamic Service Deployment Definition

Dynamic Service Deployment refers to a sequence of steps that must be taken in order to deploy a service on demand. The necessary steps regarding the service deployment are 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 our system and comparing them with the resources required by the service to be deployed.

2.2 Proposed DSD Architecture

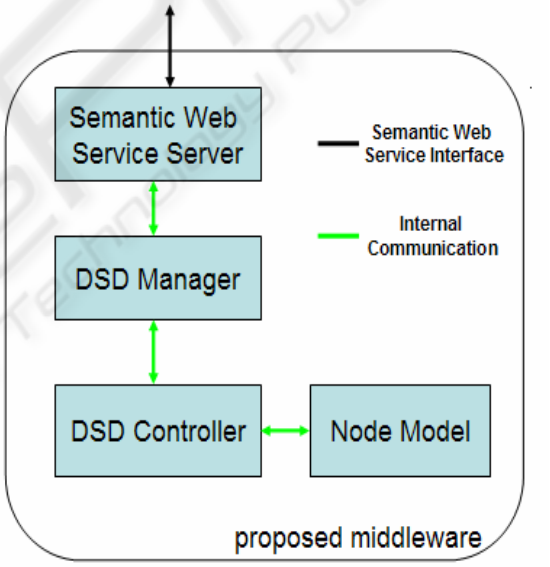

Figure 1: Internal architecture of Dynamic Service Deployment service.

Figure 1 describes the internal architecture of the Dynamic Service Deployment service:

- **Semantic Web-Service Server:** The Web-services server sub-component hosts the interfaces needed for the communication of the whole DSD Module with the External environment. The Web-service server sub-component has the functionalities needed 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 a user profile is required:
  - 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 a bootstrap process, the DSD manager passes the bootstrap services required for deployment to the DSD controller.

  The DSD manager is responsible for checking whether a user has terminated the connection, and for undoing the user's personal configuration.
- ***DSD Controller:*** The DSD controller sub-component has the following duties: it receives the service request from the DSD Manager, communicates with the proper database in order to download the service code and the service requirements, retrieves the available resources from the node model, performs the matchmaking algorithm in order to find the most suitable resources, and finally deploys the service. The DSD controller is responsible for the services in three dimensions: download, deploy, and configure.
- **Node Model:** The node model is responsible for keeping all information regarding our system. It provides a complete view of the system, and contains information on available and used physical resources, as well as data on running services.

2.3 Semantic Description of Services

The first step to semantically deploy services is to have a description of services.

The ontology chosen for our service description is based on standard device property descriptions and tries to be generic and to cover also implicit information left out of those classifications.

Figure 2 is a part of this ontology and illustrates that a service publishes its description and can be provided by different kind of devices:

Our service ontology is independent of any service model and implementations and can be applied to EJBs (Monson-Haefel, 2000), CORBA Components, Fractal components (Bruneton et al, 2006), OSGiTM bundles/services (OSGI Alliance, 2005) or Web services (Newcomer, 2002).

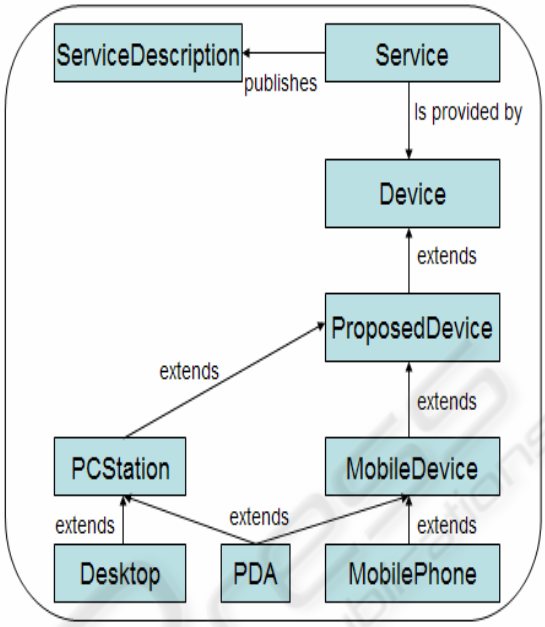

Figure 2: Proposed ontology.

Our service description is also independent of any description languages and can be written in different standard description languages such as WSDL (Newcomer, 2002), UDDI/XML (Newcomer, 2002) or OWL (McGuinness, van Harmelen, 2004).

2.4 Semantic Description of Deployment

The second step for the semantic deployment of services is to define the description of the deployment itself.

As we have seen in the previous section, each service has its own semantic description. In the same way, each execution platform has its own semantic description. The important description that we add is the semantic deployment description. This description is attached to the platform and links the different services to the platforms they are running.

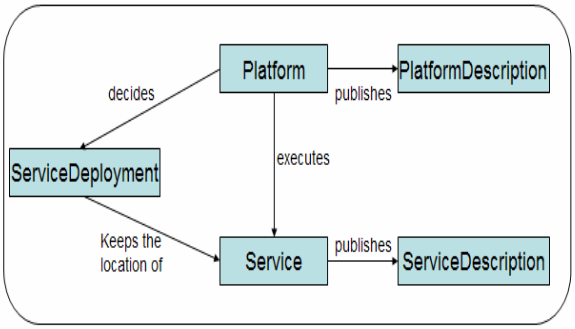

Figure 3: Proposed platform ontology.

This semantic deployment description can be instantiate in various different ways that we have also described in an ontology:

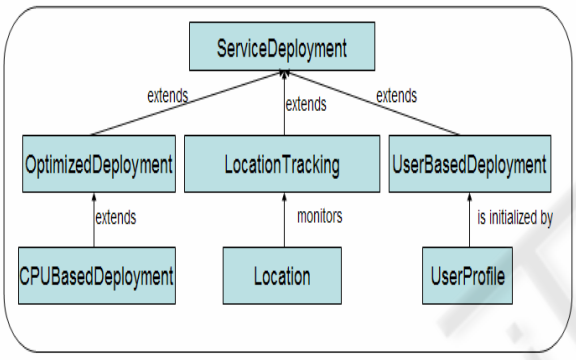

Figure 4: ServiceDeployment ontology.

For instance, figure 4 shows that the deployment can be optimized according to resources constraints (such as CPU, memory, etc), or can apply migrations to follow a user or can be customized by the user.

3 CONCLUSIONS AND FUTURE WORK

In this paper, we presented an architecture for the semantic deployment of services in a highly distributed environment. This architecture, based on a middleware distributed on each device, specially includes a Deployment Service interacting with discovery and interoperability middleware services. This Deployment Service takes into account the semantic description of services and the semantic description of deployment itself to apply a semantically and timely local deployment strategy.

We presented an architecture that adds a dynamic perspective to a Web-service-based infrastructure. Our component-based model addresses the issue of dynamic deployment of new services in a highly distributed environment and the way these address each other in that environment. We expect that this work is not only relevant to the Grid community but also to the Web-services and the network communities as we not only addressed concerns related to Grid computing but also discussed architectural issues regarding Web-service configuration and deployment.

As future work we plan to provide a more sophisticated model for service deployment and selection based on QoS properties.

REFERENCES

Object Management Group, 2006. The *Common Object Request Brocer: Architecture and Specification* (3.0.2). http://www.omg.org.

Jae-Oh, L., 2000. Enabling Network Management Using Java Technologies. *IEEE Communication Magazine 2000*; January; 38(1):116-123.

Distributed Management Task Force, 2006. Common Information Model (CIM), http://www.dmtf.org/standards/cim/.

Yang, L., Halpern, J., Gopal, R., DeKok, A., Haraszti, Z., Steven Blake, S., 2005. ForCES Forwarding Element Model. IETF draft, work in progress, <draft-ietf-forces-model-04.txt >.

Monson-Haefel, R., 2000. Entreprise JavaBeans. O'Reilly & Associates.

Bruneton, E., Coupaye, T., M. Leclercq, M., V. Quéma, V., Stefani, J., 2006. *The Fractal Component Model and Its Support in Java*. Software Practice and Experience, special issue on Experiences with Auto-adaptive and Reconfigurable Systems. 36(11-12).

OSGI Alliance, 2005. About the OSGI service platform. Technical report, OSGI Alliance (2005) revision 4.1.

Newcomer, E., 2002. Understanding Web Services, Addison-Wesley, Pearson Education. Boston, 3rd edition.

McGuinness, D., van Harmelen, F., 2004. OWL Web Ontology Language Overview, W3C.