

# A Decision Making Framework for Dynamic Service Deployment

Christos Chrysoulas, Giorgos Kostopoulos, Evangelos Haleplidis, Robert Haas,  
Spyros Denazis and Odysseas Koufopavlou

**Abstract**— Owing to the increase in both heterogeneity and complexity in today's networking systems, the need arises for an architecture for network-based services that provides 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 Web-service-based Resource Management 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. In such a model the Resource Management, in terms of representation, allocation and management of the resources, plays a vital role regarding the efficiency of the whole Dynamic Service Deployment architecture.

**Index Terms**—Web-Services, Dynamic Service Deployment, Matchmaking, Node Model.

## I. INTRODUCTION

FEDERATED and distributed systems present new challenges to resource management. Dynamic, heterogeneous and distributively owned resource environments come across with the problem of resource representation, allocation and management. Distributed management introduces resource heterogeneity. Not only the

This work is supported by the European Union's FlexiNET Project under contract FP6-IST-1-50764.

C. Chrysoulas is with the Electrical and Computer Engineering Department of University of Patras, Rio, 26500, Greece (Tel. +30-6932-121829; fax: +30-2610-994798; e-mail: cchrys@ee.upatras.gr).

G. Kostopoulos is with the Electrical and Computer Engineering Department of University of Patras, Rio, 26500, Greece (Tel. +30-6977-533441; fax: +30-2610-994798; e-mail: gkostop@ee.upatras.gr).

E. Haleplidis is with the Electrical and Computer Engineering Department of University of Patras, Rio, 26500, Greece (Tel. +30-6974-488296; fax: +30-2610-994798; e-mail: ehalep@ee.upatras.gr).

R. Haas is with IBM Research, Zurich Research Laboratory, Rüschlikon, Switzerland (Tel: + 41 44 724 8698; fax: + 41 44 724 8578; e-mail: rha@zurich.ibm.com).

S. Denazis is with Hitachi Europe SAS, Sophia Antipolis, France (Tel. +33-48 9874100; fax: +33-68 057 4833; e-mail: Spyros.Denazis@hitachi-eu.com)

O. Koufopavlou is with the Electrical and Computer Engineering Department of University of Patras, Rio, 26500, Greece (Tel. +30-6974-005424; fax: +30-2610-994798; e-mail: odysseas@ee.upatras.gr).

set of available resources, but even the set of resource types is constantly changing [1].

Our architecture makes extensive use of a component-based architecture to try to solve the problem of creating new services and the dependencies in other services and in components outside the architecture we present [2].

Interoperability across heterogeneous platforms, in particular the need to expose all or part of our application to other applications on different platforms or on different technologies, is the driving force for developing a Web-service based architecture. Web Services combine the best aspects of component-based development and the Web. Like components, Web Services represent functionality that can be easily reused without knowing how the service is implemented. Unlike current component technologies, which are accessed via proprietary protocols, Web Services are accessed via ubiquitous Web protocols (i.e. HTTP). Any type of application can be offered as a Web service. Web Services are applicable to any type of Web environment: Internet, intranet, or extranet. Web Services support Web-based access, easy integration, and service reusability [16].

Matchmaking provides a powerful and robust solution to resource management in such dynamic and distributive environments. Matchmaking is the mean for a consumer to express constraints and preferences on a resource and for the resource to express constraints and preferences on a consumer.

Our proposed Resource Management approach is developing as part of the FlexiNET [3] IST research project, specifically as a function of the Dynamic Service Deployment (DSD) module, main component of the FlexiNET Wireless Access Node (FWAN) module.

The remainder of the paper is organized as follows: Section II describes what FlexiNET is. Section III describes the architecture regarding the dynamic service-deployment. Section IV presents the proposed matchmaking framework. Section V presents the AAA Proxy user-case scenario. A discussion on related work is given in Section VI. Conclusions and future work are presented in Section VII.

## II. FLEXINET ARCHITECTURE

The proposed Resource Management approach is part of the DSD module which 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. The primary aim is not to replace or enhance existing networking infrastructures, but to offer a **value-added complementary** network architecture, addressing service access de-centralisation and separation of data, service logic and control from the pure transport network. Moving and/or concentrating interconnectivity (switching/routing), intelligence, dynamic service deployment, and service-management processes towards the edges, is vital. This would enable core networks to be treated as backbone resources being deployed by and interacting with network services and applications [4], [5], and [6].

The FlexiNET network architecture consists mainly of node instances, communication buses and data repositories, as depicted in Figure 1.

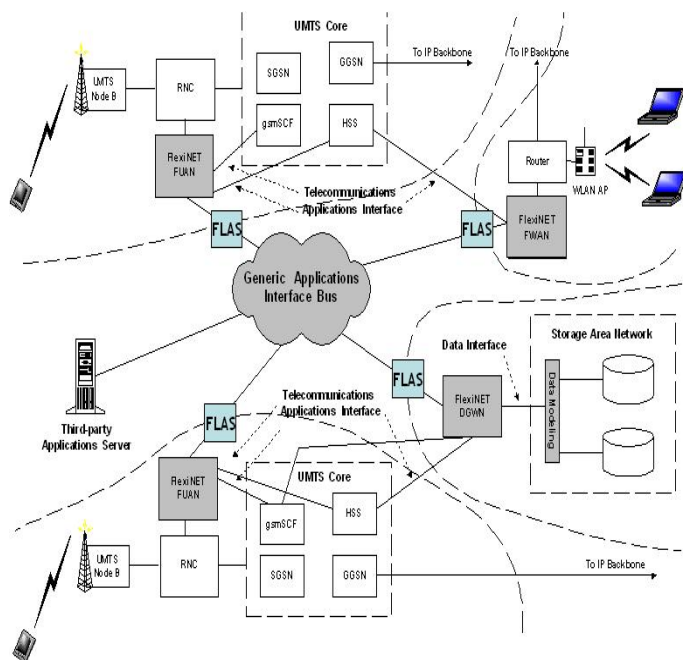


Figure 1. FlexiNET Architecture

The FlexiNET UMTS Access Node (FUAN) provides to the FlexiNET interfaces, functions such as switching/routing control, access to applications data & service logic, etc. The FUAN complements existing access nodes (RNC, BSC) of UMTS networks.

The FlexiNET WLAN Access Node (FWAN) acts as both a services access-gateway (user authentication, service authorization, service discovery, etc.), and connection gateway between WLAN infrastructures and the FlexiNET WAN. FWAN has to achieve user and service-roaming capabilities between different providers and service programmability using dynamic service deployment. Service programmability functions will be provided using the Hitachi distributed-router architecture [7], which will provide dynamic service deployment utilising the ForCES protocol [5].

The FlexiNET Data Gateway Node (DGWN) acts as the gateway between a generic Storage Area Network (SAN) infrastructure and the FlexiNET network. It is used by other

FlexiNET instances for accessing subscriber (profile, location, etc) and applications data needed for service execution. It provides a Generic Data Interface that other FlexiNET elements may use to access the stored data in the SANs.

The Generic Applications Interface Bus is used for the implementation of dynamic application-related functions and the communication of information flows pertaining to the execution of application and service logic, including a framework allowing service registration, discovery and binding.

The FlexiNET Applications Server (FLAS) is the physical entity, which hosts the logic of the applications that the FlexiNET network architecture provides. These services are called from other entities remotely and executed locally. Using the DGWN they are in position to retrieve specific information needed for services execution.

The DSD module is part of the FWAN, but the functionality can be added in every agent element of the FlexiNET architecture, e.g. FUAN. The FWAN architecture can be seen in Figure 2 and is based on Hitachi's distributed router. Hitachi's distributed router consists of two functional blocks, the basic and the extensible function block. The basic function block processes the general packet-forwarding functions which are common to all services, e.g. it classifies received packets, handles routing table retrieval, packet switching, and node management. The extensible function block processes the packet-forwarding functions for specific protocols and services, such as packet filtering for firewalls, layer-7 processing for content-switching, address translation, encryption, etc.

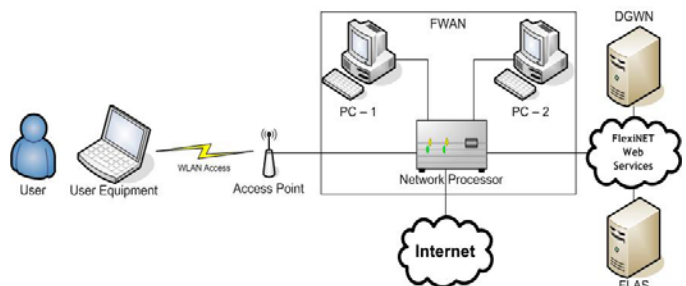


Figure 2. FWAN architecture

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

### III. DSD ARCHITECTURE

#### A. DSD 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 the FWAN and comparing them with the resources required by the service to be deployed.

### B. Proposed DSD Architecture

Figure 3 depicts the current, proposed DSD architecture. As can be deduced from the Figure, the DSD is the sum of the following sub-components:

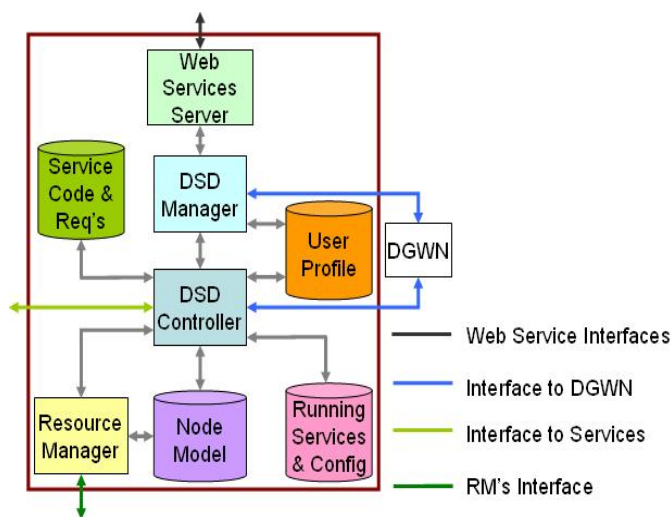


Figure 3. DSD Architecture

- Web Services Server
- DSD Manager
- DSD Controller
- Resource Manager
- Node Model
- User Profile
- Service Code and Requirements
- Running Services and Configuration

Interested readers are referred to [2] for further details.

The Dynamic Service Deployment module (DSD) must be deployed on the FWAN during boot-up. The bootstrap process is responsible for booting up the FWAN with the AAA proxy module. To accomplish this task, it reads from a static configuration file which is stored in a local copy 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 will mainly trigger the installation of the AAA proxy through the DSD module.

## IV. MATCHMAKING

In this section we briefly describe the fundamental processes and components of our matchmaking framework.

Matchmaking provides a powerful and robust solution to resource management in such dynamic and distributed environments. Matchmaking is the mean for a consumer to express constraints and preferences on a resource and for the resource to express constraints and preferences on a consumer.

The underlying ideas of the matchmaking paradigm are intuitive and very simple. We use a rather simple

matchmaking algorithm since our aim is to test the functionality of the proposed architecture not the matchmaking algorithm itself. The algorithm we choose to implement in order to best serve the above need is the One-at-a-time-with-Best-Fit. In simple words, only one service will be deployed in each Module at a time by using the Best-Fit algorithm.

A module is needed to keep all the necessary information, in terms of resources, of the whole FWAN. For that purpose the Node Model was designed and implemented. This model is based upon hierarchy. The implemented architecture allows the model to be more generic and to be structured more logically top to bottom. The proposed architecture is depicted in Figure 4.

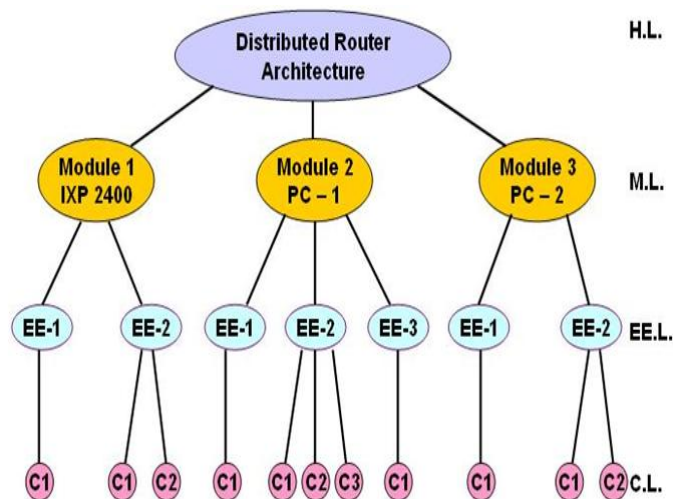


Figure 4. Node Model Architecture

The Node Model was designed in such a way to easily allow new Modules to be added. The Node Model has plug-n-play capabilities. For example, as soon as a new module is connected to the LAN automatically sends an XML file describing itself to the Node Model. Then by using usual parsing techniques (mainly DOM parsing) the newcomer's Module information is added to the main XML file.

The proposed architecture is a four layered architecture. In High Level (H.L.) there is the Distributed Router. A distributed router is a collection of off-the-shelf physical boxes interconnected in a LAN and managed to appear as one physical box with the abilities of a standard router. In Module Level (M.L.) there are the Modules. As a module we define stand-alone off-the-shelf equipment that has processing capabilities. Example of modules are PC's or ASIC's. In the Execution Environment Layer (E.E.L.) lays the platform on which the components run. In the Component Level (C.L.) lay the components. Components are blocks of software running on an execution environment and perform specific functions. If we use the ForCES model a component can be an LFB [9]. The Node Model is used to describe Hitachi's distributed router, but can be used to describe any similar system.

Agents describe their capabilities using XML and advertise themselves by sending messages to the DSD Module. These messages contain descriptive information about the agent. The

service requirements come from the DGWN as XML. The Matchmaker runs the matchmaking algorithm and finds compatible pairs in terms of needed and available resources. It is then DSD's module burden to run the service to the agent that best fits the needs of the "want to be deployed" service.

Example advertisement of a PC's (agent's) resources can be seen in Figure 5. Example of service requirements in terms of CPU Cycles, Threads and Memory can be seen in Figure 6. XML is used to describe all the needed information.

```
<?xml version="1.0" encoding="UTF-8"?>
<Module>
  <ID>XX</ID>
  <IP_Address>150.140.xxx.xxx</IP_Address>
  <OS>Fedora Core 2</OS>
  <Resources Resource_type="Maximum">
    <ID ID="1">
      <Type>Thread</Type>
      <Value>12</Value>
    </ID>
    <ID ID="2">
      <Type>Memory</Type>
      <Value>512</Value>
    </ID>
    <ID ID="3">
      <Type>CPU Cycles</Type>
      <Value>1000</Value>
    </ID>
  </Resources>
  <Resources Resource_type="Available">
    <ID ID="1">
      <Type>Thread</Type>
      <Value>10</Value>
    </ID>
    <ID ID="2">
      <Type>Memory</Type>
      <Value>328</Value>
    </ID>
    <ID ID="3">
      <Type>CPU Cycles</Type>
      <Value>500</Value>
    </ID>
  </Resources>
</Module>
```

Figure 5. Agent Advertisement

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceModel>
  <OS>Fedora Core 2</OS>
  <Resources Resource_type="needed">
    <ID ID="1">
      <Type>Thread</Type>
      <Value>2</Value>
    </ID>
    <ID ID="2">
      <Type>Memory</Type>
      <Value>0</Value>
    </ID>
    <ID ID="3">
      <Type>CPU Cycles</Type>
      <Value>500</Value>
    </ID>
    <ID ID="4">
      <Type>Diskspace</Type>
      <Value>0</Value>
    </ID>
    <ID ID="5">
      <Type>Bandwidth</Type>
      <Value>0</Value>
    </ID>
  </Resources>
</ServiceModel>
```

Figure 6. Service Request

## V. USER-CASE SCENARIO: DEPLOYMENT OF THE AAA PROXY COMPONENT

We have applied the DSD functionality to dynamically deploy the AAA Proxy. In Figure 7 the AAA Proxy architecture is being presented.

The AAA proxy module forwards the authentication packets to the FLAS Server, encapsulates the EAP packets [8] into XML messages that are passed over Web services and vice versa, 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.

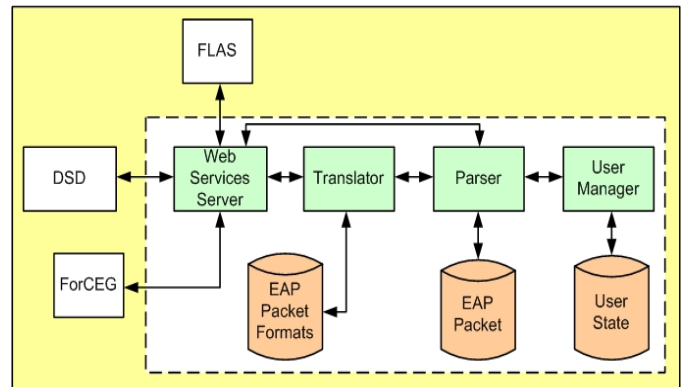


Figure 7. AAA Proxy Architecture

The AAA proxy module forwards the authentication packets to the FLAS Server, encapsulates the EAP packets [8] into XML messages that are passed over Web services and vice versa, 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 the AAA proxy service code through the DGWN and deploys it on any of the two PCs based on specific algorithms. In addition, based upon the user profiles, the DSD module will deploy a quality of service module (QoS) that 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 [9].

More details can be found in [10].

The sequence of events that takes place for the deployment of the AAA proxy can be seen in Figure 8.

- The DSD manager receives a "Deploy AAA proxy" command through the Web service interface.
- The DSD manager understands that this command has been issued by the bootstrap process and communicates with the DGWN, via a Web service, to download the code and the requirements for deploying of the AAA proxy.
- The DSD controller locates the module where the AAA proxy should reside.

- The DSD controller deploys the AAA proxy.
- The DSD controller sends an acknowledgement to the DSD manager which in turn sends the acknowledgement to the bootstrap process saying that the AAA Proxy has been deployed.

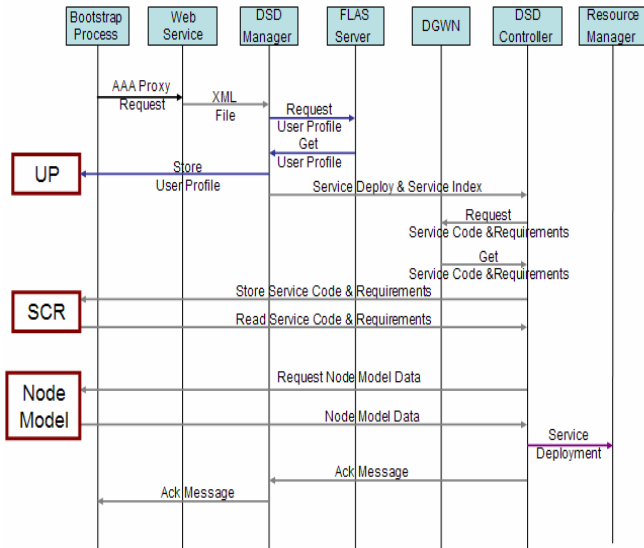


Figure 8. Message exchange during the deployment of the AAA Proxy

## VI. DISCUSSION OF RELATED WORK

The concept of matchmaking is not new since the topic is widely studied for agent systems. Agent systems such as ACL [11] and RETSINA [12] employ powerful advertisement languages with inferencing capabilities so that behavioral specifications of agents can be described. In contrast to the knowledge-based representations used in these systems, the proposed solution uses an XML representation of the available, needed resources. XML is simple and lightweight allowing efficient and robust implementation.

Much effort has been devoted to developing information systems for publishing, aggregating and supporting queries against collections of service descriptors (SNMP [13], LDAP [14], MDS [15], UDDI [16]). Such systems differ in various dimensions, such as their description syntax (e.g., MIBs [13], relations [17], LDAP objects [14]), query language (e.g., SQL [17], XQuery, LDAP query [14]), and the techniques used to publish and aggregate service descriptions (e.g., soft state vs. stateful servers). However these systems all have in common that the service agent-consumer interaction is asymmetric: the agent-published description of its resources (properties) is queried by the consumer to identify candidates prior to generating requests for service. Such strategies require that service provider policy to be enforced only after candidate services are selected by the incoming customer, which can result in wasted effort.

## VII. CONCLUSION AND FUTURE WORK

Dynamic, heterogeneous and distributively owned resource environments present unique challenges to the problem of

resource representation, allocation and management. In this paper, we have introduced a resource management system supported by a simple matchmaking algorithm to address the problem of heterogeneous resource co-allocation, motivated by the intent of solving a real problem encountered during the designing and testing of the FlexiNET's FWAN node.

Our contribution is in designing an adequate Resource Management system and accompanying implementation of the Resource Manager itself and a matchmaking algorithm, proving the feasibility of the proposed solution.

A significant goal of our future work is to incorporate preferences into the matchmaking algorithm. In this way a more sophisticated algorithm(s) will occur to cope with the possibility of more demanding services.

## ACKNOWLEDGMENT

This work is supported by the European Union's FlexiNET Project under contract FP6-IST-1-50764.

## REFERENCES

- [1] Raman, R., Livny, M., Solomon, M., "Policy Driven Heterogeneous Resource Co-Allocation with Gangmatching," *hpdc*, p. 80, 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12 '03), 2003.
- [2] Chrysoulas, C., Haleplidis, E., Haas, R., Denazis, S., Koufopavlou, O., "Applying a Web-Service-Based model to Dynamic Service-Deployment", International Conference on Intelligent Agents, Web Technology and Internet Commerce - IAWTIC'2005, Vienna.
- [3] FP6-IST1 507646 FlexiNET Technical Annex.
- [4] FP6-IST1 507646 FlexiNET D21 "Requirement, Scenarios and Initial FlexiNET Architecture".
- [5] FP6-IST1 507646 FlexiNET D22 "Final FlexiNET Network Architecture and Specifications".
- [6] Aladros, R., Kavadias, C., 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", IST Mobile & Wireless Communications Summit 2005, Dresden.
- [7] Hirata, T., and Mimura, I., "Flexible Service Creation Node Architecture and its Implementation", IEEE Computer Communications Workshop, 2003.
- [8] RFC 3748: Extensible Authentication Protocol (EAP), June 2004
- [9] Haleplidis, E., Haas, R., Denazis, S., Koufopavlou, O., "A Web Service- and ForCES-based Programmable Router Architecture", IWAN2005, France.
- [10] Kostopoulos, G., Haleplidis, E., Lampropoulos, K., Denazis, S., Koufopavlou, O., "A Web Service based Wireless Authentication Scheme", 12<sup>th</sup> European Wireless Conference, EW2006, Greece.
- [11] <http://www.w3.org/2001/04/20-ACLs>, January 2006.
- [12] <http://www.cs.cmu.edu/~softagents/retsina.html>, January 2006.
- [13] Stalling, W., "SNMP, SNMPv2, SNMPv3, and RMON", 1 and 2, 3<sup>rd</sup> edn., Addison-Wesley, Reading, Mass., xv, 619 pp.
- [14] Howes, T., Howes, T.A., Smith, M.C. and G.S., "Understanding and Deploying LDAP Directory Services", 2<sup>nd</sup> edn., Addison-Wesley Professional, 2003, 608 pp.
- [15] Globus, MDS document <http://www.globus.org/toolkit/mds/>.
- [16] Newcomer, E., "Understanding Web Services: XML, WSDL, SOAP, and UDDI", Addison-Wesley, Boston
- [17] Garsia-Molina, H., Ullman, J.D., and Widom, J., "Database Systems: the complete book", Prentice Hall, Upper Saddle River, NJ, 2002, xxvii, 1119 pp.