

Towards a Resource Management and Service Deployment Framework

Christos Chrysoulas^{1,*}, Evangelos Haleplidis¹, Giorgos Kostopoulos¹,
Robert Haas² and Odysseas Koufopavlou¹

¹ *Electrical and Computer Engineering Department of University of Patras, Rio, 26500, Greece*

² *IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland*

SUMMARY

Owing to the increase in both heterogeneity and complexity in today's networking systems, the need arises for new network-based services architectures. They must provide flexibility and efficiency in the definition, deployment and execution of the services and, at the same time, handle the adaptability and evolution of such services. In this paper we present an approach that applies a Web-service-based resource management framework. It enables the provision of parallel applications as QoS-aware applications, 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 entire dynamic service deployment architecture.

1. INTRODUCTION

NETWORK and service management fields nowadays find themselves at crossroads with middleware technologies, new network architectures and emerging research directions. Middleware technologies like web services have reached maturity and enjoy wide deployment and adoption. Network architectures and infrastructures built for different purposes are well 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, increasing the challenges even further.

This activity has coincided with the end of an era in network and service management during which vast experience has been accumulated and numerous lessons have been learned. It is 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.

*Correspondence to: Christos Chrysoulas, Electrical and Computer Engineering Department of University of Patras, Rio, 26500, Greece (E-mail: cchrys@ee.upatras.gr)

This produces speculation and activity about redefining/reassessing the initial requirements that drove the developments in network and service management in the past and about the “shape” of management when projected into the future.

The most prominent decentralized management approaches are based on distributed object technologies such as CORBA (Common Object Request Broker Architecture) [1] and Java RMI (Remote Method Invocation) [2]. 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 that offer management capabilities over the Web. Such approaches are based on the Common Information Model (CIM), which encompass a set of common management operations [3].

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

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, such as AAA (Authentication, Authorization, and Accounting), monitoring and SLA (Service-Level Agreement) support, are treated in homogeneous way in a heterogeneous environment.

Network Architectures with Distinct Separation of Concerns

The most representative example is the separation of control from the forwarding plane [4], which allows the two to evolve separately. The binding element between the two is a set of open interfaces that abstract functionality and allow access to vendor-independent functionalities and resources.

Distributed Network Node Architectures

Individual network nodes and other devices are clustered together 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, the injection of new functionality into the network.

In such a 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. Common denominator is the management functionality that injects services and components on demand and configures the network end-to-end. Unless we address these challenges with

sufficient and complete technical solutions first, it is difficult to see how new research initiatives, e.g. autonomic networking, can be brought to a successful outcome.

This has been the primary motivation of our research presented in this paper, which touches upon these issues by exploring potential solutions on the service deployment and network configuration within a network architecture, called FlexiNET, that bears the aforementioned architectural characteristics.

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

Our research has been carried out as part of the FlexiNET [5] European Union IST research project, and we have developed the Dynamic Service Deployment (DSD) functionality, hosted by the FlexiNET Wireless Access Node (FWAN), one of the key architectural components of the FlexiNET architecture.

The remainder of the paper is organized as follows: An overview of related work is presented in Section 2. Section 3 describes what Web services really are. Section 4 briefly describes the FlexiNET architecture. Section 5 presents an instance of a distributed node architecture that functions as a distributed router exploiting the DSD functionality and benefits from the open configuration capability. Section 6 focuses on the design of the DSD architecture in FlexiNET, while Section 7 proposes a matchmaking approach that is used in the context of resource management during service deployment. Section 8 discusses the dynamic configuration over Web Services. Section 9 introduces an Authorization, Authentication and Accounting (AAA) scenario that demonstrates the use of the DSD and configuration functionality. Section 10 gives the experimental results. Conclusions and an outlook on future work are presented in Section 11.

2. DISCUSSION OF RELATED WORK

Agent systems such as ACL [6] and RETSINA [7] 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 solution proposed here uses an XML representation of the available, needed resources. XML is simple and lightweight, enabling an 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 [8], LDAP [9], MDS [10], UDDI [22]). Such systems differ in various dimensions, such as their description syntax (e.g., MIBs [8], relations [12], LDAP objects [9]), query language (e.g., SQL [12], XQuery [13], LDAP query [9]), and the techniques used to publish and aggregate service descriptions (e.g., soft state vs. stateful servers). However, all these systems 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 be enforced only after candidate services have been selected by the incoming customer, which can result in a wasted effort.

Distributed component models, such as CORBA [14] or DCOM [15], are widely used, mainly in the context of commercial applications. The Common Component Architecture developed within the CCA Forum [16] defines a component model for high-performance computing based on interface definitions.

XCAT3 [17] is a distributed framework that accesses Grid services, e.g. OGSi [18], based on CCA mechanisms. It uses XSOAP [19] for communication and can use GRAM [20] for remote component instantiation. The Vienna Grid Environment (VGE) [21] is a Web-service-oriented environment for supporting high performance computing applications. The VGE has been realized based on state-of-the-art Grid and Web service technologies, Java and XML. The Globus Toolkit 4 [20] 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. Our model clearly addresses the above issue regarding the dynamic deployment of new services.

3. WEB SERVICES

3.1 Web-service Technology

Interoperability across heterogeneous platforms, in particular the need to expose all or part of a particular application to other applications on different platforms or on different technologies, is the driving force for developing a Web-service architecture.

Web services standardize the messages that entities in a distributed system must exchange to perform various operations. At the lowest level, this standardization concerns the protocol used to transport messages (typically HTTP), message encoding (SOAP), and interface description (WSDL). A client interacts with a Web service by sending a SOAP message; the client may subsequently receive response message(s) in reply. At higher levels, other specifications define conventions for securing message exchanges (e.g., WS-Security), for management (e.g., WSDM), and for higher-level functions, such as discovery and choreography [22]. Figure 1 presents an overview of these different component technologies.

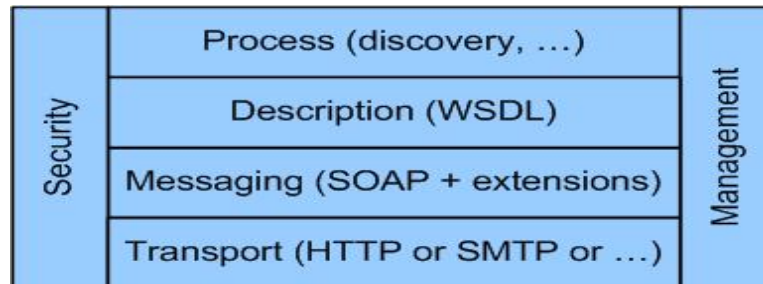


Figure 1. An abstract view of the various specifications that define the Web-service architecture.

3.2 Web-service Implementation

From the client's perspective, a Web service is simply a network-accessible entity that processes SOAP messages. To simplify service implementation, it is common for a Web-service implementation to distinguish the following:

- The hosting environment, i.e., the logic used to receive a SOAP message and to identify and invoke the appropriate code to handle the message, and potentially also to provide related administration functions.
- The Web-service implementation, i.e., the code that handles the message.

The SOAP message is typically transported via HTTP, thus an “HTTP engine” or “Web server” is required and a “SOAP engine” needed to process the SOAP messages. Figure 2 illustrates the various components.

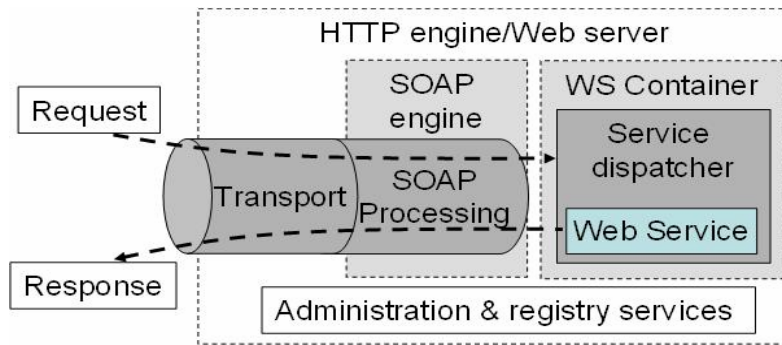


Figure 2. High-level view of the functional components of the Web service.

3.3 XML-based Architecture

Several Extensible Markup Language (XML)-based network management techniques have recently emerged in an effort to overcome the limitations of SNMP management in configuration operations. As XML documents can have much higher-level semantics than SNMP, they are more appropriate for bulk configuration operations. Exporting XML interfaces for network management has also other advantages because

- XML is easy to generate, parse and process, supports sophisticated data structuring, and can therefore handle complex organizations of management information;
- XML DTD (Document Type Definitions) and XML Schemas specify and validate the structure of XML documents, thus freeing developers of network management applications from tedious tasks; and
- XML comes with numerous W3C technologies supporting a rapid development of network management applications (e.g., Extensible Stylesheet Transformations (XSLT), which transform XML documents to other XML formats, and XPath/XQuery for discovering XML elements that are subject to criteria).

XML management systems with SOAP interfaces constitute an emerging architecture for WS-based network management. This architecture leverages the advantages of XML systems with respect to the programmability and cost-effectiveness of the management operations. The drawbacks of XML-based architectures lie in the processing overhead incurred by the parsing and construction of XML messages and files, which can lead to performance penalties. XML gateways tend to increase the response time for network management operations. Although this is a disadvantage of XML management per se, it becomes less important when SOAP interfaces are used. This is because the fact that XML gateways deal with XML documents eliminates the additional overhead imposed by SOAP. SOAP constitutes a more natural interface for XML management systems that already handle XML messages. This is a significant incentive for using SOAP in XML systems. Note that in any case SOAP interfaces are very likely to proliferate because SOAP is an open standard with a growing community of developers and vendors supporting it [23].

4. FLEXINET ARCHITECTURE

The proposed DSD and configuration approaches have been developed within the context of 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) that offer roaming connection control, switching/routing control, and advanced services management/access functions to the network access points by taking advantage of the separation of control from the transport plane and the separation (creation) of the data plane from applications and services. A main objective 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.

An innovation introduced by the FlexiNET architecture, which represents an additional dimension in the separation of concerns mentioned earlier, is the concept of the separation of data placed in a network-wide logical database taking the form of what is termed the *data plane* in FlexiNET. Such a data plane hosts a wide variety of data ranging from user profiles, service profiles, service component repositories to large configuration data – a key requirement of internet network operators as explained in RFC 3535. To this end, the data in the data plane may become accessible by different services and applications, including management ones that use them, as well as different operators or third parties belonging to the same federation.

Furthermore, the FlexiNET node instances mainly reside at the edges of the network to move and/or concentrate interconnectivity (switching/routing), intelligence, dynamic service deployment, and service-management processes towards or at the edges. This would enable core networks to be kept simple and be treated as backbone resources being deployed by and interacting with network services and applications [5], [24].

The FlexiNET network architecture (shaded in grey in Figure 3) mainly consists of these node instances and data repositories connected by means of (open) communication buses.

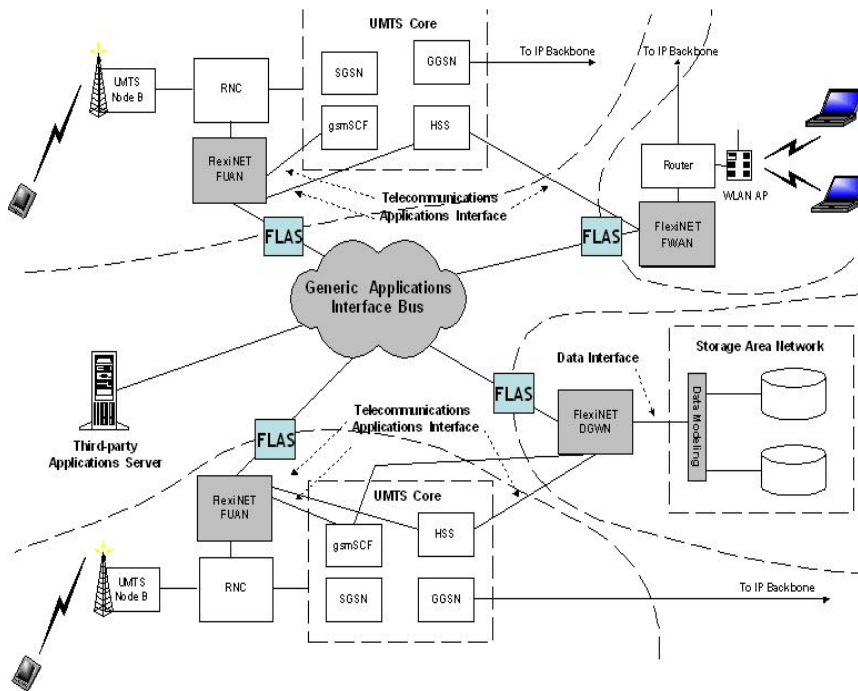


Figure 3. FlexiNET architecture.

More specifically, the node instances take the form of two types of access nodes, namely, FUANs and FWANs:

The UMTS access node, called FlexiNET UMTS Access Node (FUAN) hosts functions such as switching/routing control, access to applications data and service logic, etc.. It 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 internet. FWAN aims at achieving user- and service-roaming capabilities between different providers and service programmability using dynamic service deployment and open interfaces. Service programmability functions will be provided by implying a distributed network node architecture based on Hitachi's distributed-router architecture [25].

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 to access subscriber (profile, location, etc) and application data needed for service execution. It provides a Generic Data Interface that other FlexiNET elements may use to access the data stored 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 that enables service registration, discovery and binding. DGWN, SAN and the Data Interface represent the components of the data plane.

The FlexiNET Applications Server (FLAS) is the physical entity that hosts the logic of a variety of applications and services, and exploits the capabilities of the FlexiNET network architecture. These services

are called remotely from other entities and executed locally. Using the DGWN, they are able to retrieve specific information needed for service execution.

5. DISTRIBUTED ROUTER ARCHITECTURE

The idea behind the distributed router architecture as proposed by Hitachi [25] is the separation of basic forwarding and packet delivery functionality from the service/control logic that may reside in access nodes in order to foster service creation by means of service deployment capabilities. As depicted in Figure 4, the architecture consists of two blocks, the Basic and the Extended block. Each functional block may in turn be composed of several modules. This is the architecture adopted to build the FWAN FlexiNET node instance.

The Basic function block has packet-processing modules that classify received packets, retrieve routing tables and reroute or terminate packet flows to extensible function block modules for further processing.

The Extended function block has service modules that handle the processing for service-specific functions and control and management modules that process the routing protocols, management-agent functions, etc. These modules are inserted and removed as required for performance or for service provision by the node.

The two blocks are loosely connected by a generic physical interface such as a Gigabit Ethernet network interface, and communicate with each other by exchanging packets that include control messages across this interface.

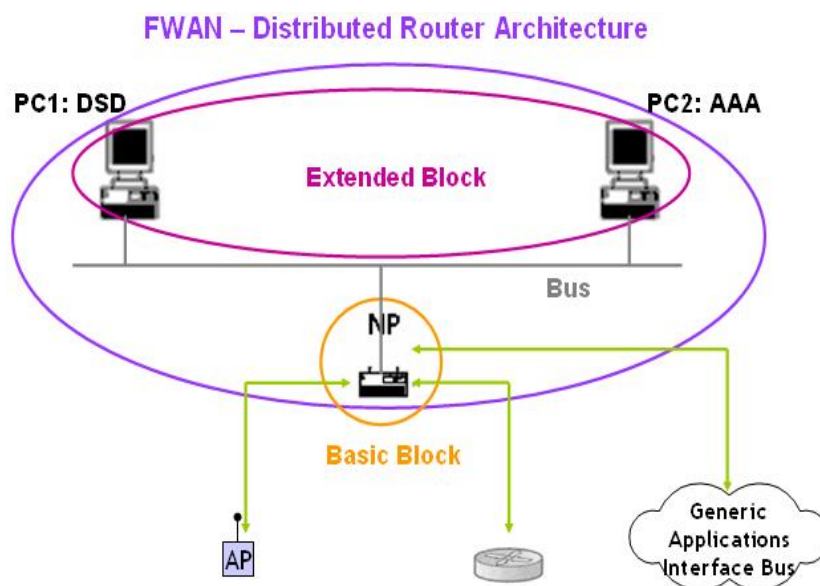


Figure 4. A distributed network node architecture.

The advantage of this architecture is that new modules may be dynamically added to the extensible block as needed. As a result, flexibility and scalability are greatly improved because the modules to be added form a pool of resources and are capable of hosting new functionality in the form of new services and components thereof. This feature will be exploited when we describe the DSD architecture.

Figure 5 shows the internal configuration of the node.

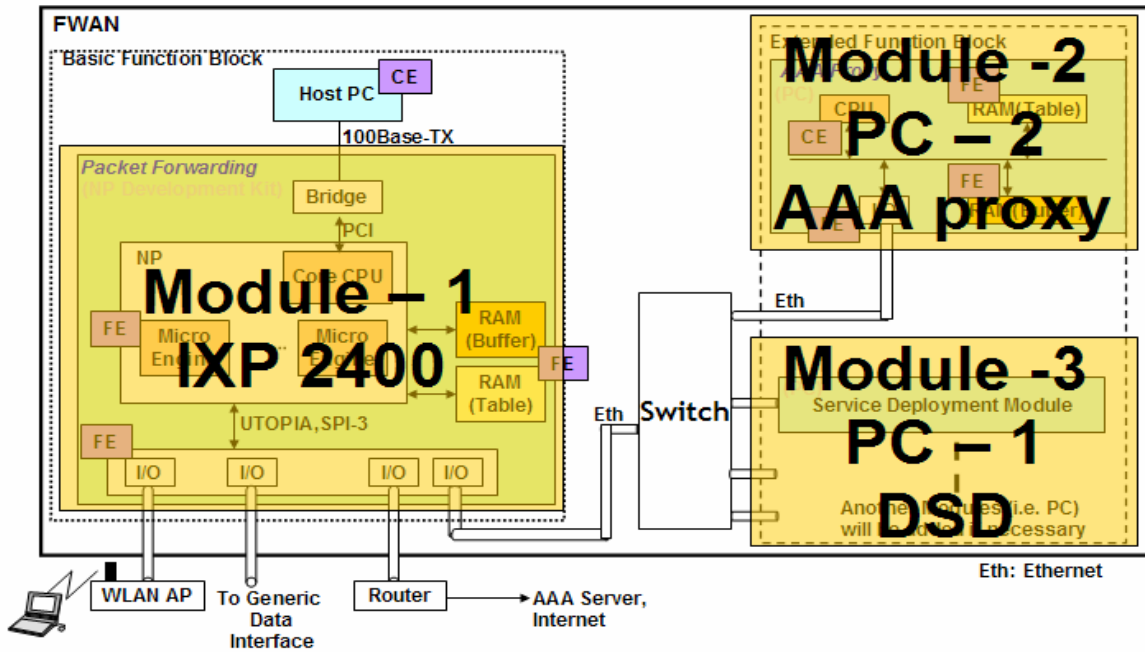


Figure 5. Internal configuration of the node.

6. DSD ARCHITECTURE

The DSD architecture performs two major tasks, namely, (i) the decision where to deploy the services upon reception of a service request and (ii) the execution of a service-deployment decision.

The former task is based on a matchmaking algorithm that implements different strategies. For the matchmaking algorithm to run, it first needs to be fully aware of the node environment in which deployment is going to take place. This includes the amount of resources, the capabilities as open interfaces, and the composition (operating systems, programming environments supported, etc.) of each of the modules in the distributed environment. Second, it must be able to understand the service requirements in terms of resources as well as code modules and location thereof that require deployment. These two sets of information are compared to determine which module meets the service deployment criteria in the most efficient manner.

This comparison task comprises all the mechanisms and interfaces necessary to carry out the deployment by contacting the code server, downloading, loading and running the code, and finally configuring the local networking environment according not only to the service requirements but also user profile that stores the signed SLA of the customer.

Resource Manager

The Resource Manager component discovers and monitors the resources. With the help of the resource manager Interface, it collects information from all the modules of the distributed router according to the data definitions captured in the node model. It also communicates with the DSD controller to feed the matchmaking algorithm with the current state of the node model.

Node Model

The Node Model is responsible for keeping all information for every module that comprises the distributed router so that a complete view thereof is maintained. This information includes availability and usage of physical resources, as well as data on running services. The node model is an essential part of the proposed DSD architecture and its functionality as it is equivalent of information models like MIBs or CIM, but customised for the purpose of service deployment [26]. More details on the structure and the information captured by the Node Model will be provided later in the text.

User Profile

The User Profile is a data storage facility where the downloaded user profile is stored so as to keep track of the active users.

Service Code and Requirements

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

Running-services and Configuration

Finally, the Running-services and Configuration data storage facility is responsible for storing data about running services and their current configuration.

5.2 Node Model Structure

We have approached the modeling of the node information relevant to deployment as a four-layered tree structure that is aligned with the structure of the distributed router architecture as the latter is the actual environment in which deployment takes place (Figure 7). In moving from the root towards the leaves, each level abstracts that part of information that has been associated with a specific component of the distributed architecture. In addition, the component contains information regarding the parent node at the preceding level as well as children nodes at the next lower level.

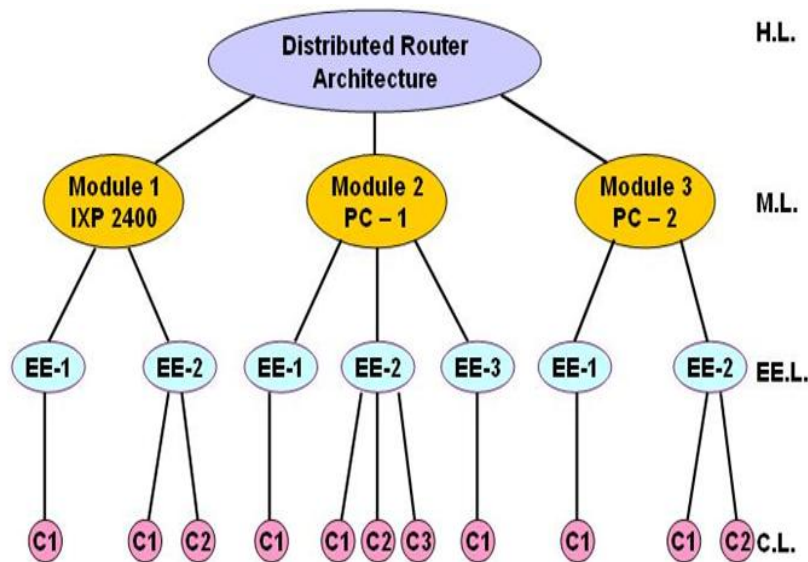


Figure 7. Node Model Structure.

The High Level (H.L.) (root) corresponds with the complete Distributed Router. As explained above, a distributed router is in reality expected to be a collection of off-the-shelf physical modules interconnected by means of a high-speed bus and managed to appear as one physical box with the functionality of a standard router. It is this kind of information that is provided by this level and allows any enquiring entity to discover the current topology of the distributed network node.

The next level represents the Module Level (M.L.) associated with information pertaining to a wide range of possible physical modules. Examples of modules are entire PCs, ASICs, or special-purpose embedded systems such as network processors boards.

The third level is the Execution Environment Layer (E.E.L.), namely, the necessary programming environments in which the components run. These programming environments are hosted by specific modules. For instance, a Java Virtual Machine hosts service components implemented in Java.

Finally, the Component Level (C.L.) contains information for the actual service components to be deployed. Components are usually blocks of software running in the execution environment and performing specific functions as part of the service they realize.

The Node Model is described in XML and makes provisions for accommodating new modules as they are being dynamically added to the router architecture. To this end, whenever a new module joins the distributed router it automatically sends an XML file to the resource manager describing itself. The resource manager, in turn, updates the node model. Similar operations take place whenever new components are being deployed or states change. In this way, the node model is always kept up-to-date, thus allowing the matchmaking algorithm to take decisions that are as accurate as possible.

7. MATCHMAKING

In this section we describe the fundamental processes and components of our matchmaking approach and how it exploits the node model.

Matchmaking provides a powerful and robust solution to resource management in such dynamic and distributed environments. Matchmaking is the means for a consumer to express constraints and preferences on a resource and vice versa. It is particularly important in the context of dynamic service deployment, where the discovery process and the deployment of new services need to be automated.

The underlying ideas of the matchmaking paradigm are intuitive and very simple. *We use a rather simple matchmaking algorithm as our aim is to test the functionality of the proposed architecture and not the performance of matchmaking algorithm itself.* The algorithm we have chosen to implement to best serve the above need is the One-at-a-time-with-Best-Fit. According to it, only one service at a time will be deployed in each module by using the Best-Fit algorithm. Best fit means that the module upon which the service will be deployed will have utilized the least percentage of its resources remaining before the service was deployed, based on the grading of each different resource.

To describe the resources of our system best it will be wise to decide the priority of each resource and weigh it with a multiplication factor. Accordingly, following list gives the grade for each resource:

- Memory = * 6
- Disk space = * 5
- CPU cycles = * 4
- Bandwidth = * 3
- Threads = * 2

```
<?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 8. Agent advertisement at the module level.

As depicted in Figure 8, a module keeps all the necessary information, in terms of resources, of the entire distributed router environment. For this purpose, the Node Model was designed and implemented.

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 are issued by the DGWN as XML. The Matchmaker (a subcomponent of the DSD Controller) runs the matchmaking algorithm and finds compatible pairs in terms of needed and available resources. It is then the role of the DSD module to deploy the service to the agent that best fits the needs of the service to be deployed.

The very first task is to determine which module is capable of deploying the service.

```
<?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 9. Service request.

This Manager-Agent system model presents a communication framework in which standardization affects only the way in which management information is modeled and carried across systems, deliberately leaving aspects of their internal structure unspecified. This achieves a highly optimized implementation of the architecture already designed.

An example of advertisement of a PC's (agent's) resources is given in Figure 8, and an example of service requirements in terms of CPU cycles, threads and memory is shown in Figure 9. XML is used to describe all the information needed.

8. DYNAMIC CONFIGURATION OVER FORCES

Once the service has been deployed, the next step is to configure FWAN. The FWAN is configured using a Web-Services Gateway called ForCES Gateway (ForCEG) [27]. ForCEG is a middleware between the Forwarding and the Control Plane. It translates control-plane XML requests into forwarding-plane configuration using the ForCES protocol.

The Forwarding Plane, from the ForCES model [4] point of view, has been modeled using Logical Function Blocks (LFBs), i.e., blocks encapsulating fine-grained operations of the forwarding plane. Each

LFB is responsible for completing only a single, specific task, such as reducing the TTL field of an IPv4 packet. Concealing the details of the ForCES model reduces the cost of designing new Control Plane applications.

To conceal the details of the ForCES model, higher-layer functions such as Firewall, Routing, QoS-related, etc., can be introduced, and associations can be made between these functions and LFBs. The ForCEG can advertise the higher-layer functions in a WSDL [28] file to Control Plane Processes, which we call Main Control Programs (MCPs). The ForCEG then can recognize which LFBs the MCP needs to modify. An MCP “targets” a higher-layer function, provides the necessary attributes according to the “target” field, and inserts an operation (such as SET or GET). The ForCEG is then able to construct the necessary ForCES messages to be sent to the appropriate LFBs.

Figure 10 depicts the “Target” concept. The ForCES Forwarding Element (FE) Start/Termination Point is a necessary component of the ForCES protocol [11] because it is the source and the destination of ForCES messages. As the QoS software module needs to set up classifier rules, it “targets” a QoS-related function, provides the necessary classification rules, and issues a SET command.

Once the message has been received from the MCP, the required information needs to be extracted by a parser and checked for consistency by a control module. A translator module inside the ForCEG will translate the attributes of the message, and a ForCES Control Element (CE) Start/Termination Point will transmit the ForCES packet to the Forwarding Element, as described in [28]. Figure 11 depicts the ForCEG architecture as described above.

In the WSDL file, an MCP can acquire data regarding the URL for calling the API, the higher-layer functions available, and an XML schema that defines which data types each of these higher-layer functions requires. Making ForCEG web-service-enabled allows all supported operations to be advertised and discovered through one URL.

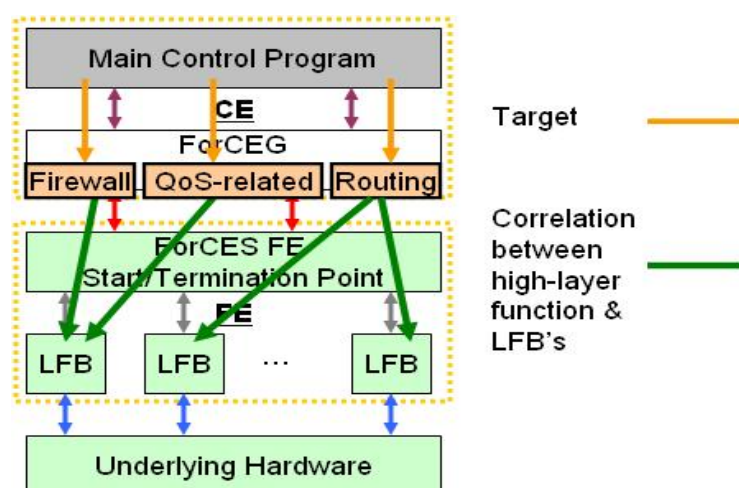


Figure 10. “Target” concept.

In addition, the ForCEG is able to dynamically download other mappings between higher-layer functions and LFBs from an external source, and re-publish them to the UDDI registry. Such a dynamic configuration adds additional programmability capabilities to the ForCEG.

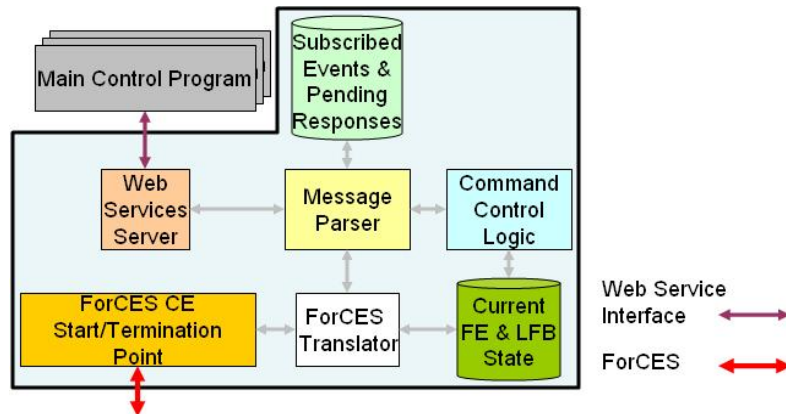


Figure 11. ForCEG architecture.

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

We have applied the DSD functionality to dynamically deploy the AAA Proxy service. Figure 12 shows the AAA Proxy architecture.

The AAA proxy module forwards the authentication packets to the FLAS Server, and encapsulates the EAP packets [29] 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.

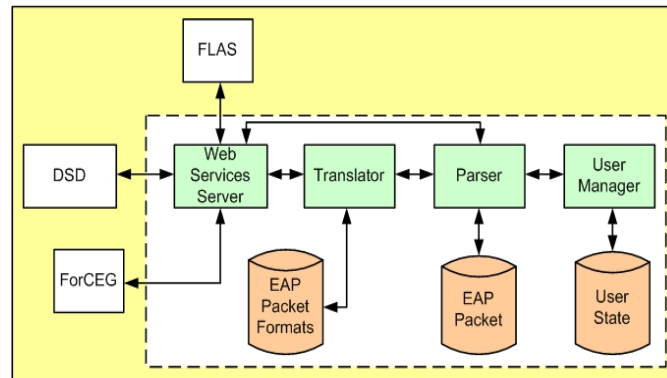


Figure 12. AAA proxy architecture.

At boot-up, the boot-up process request the DSD module to deploy the AAA proxy module. The DSD module retrieves the AAA proxy service code through the DGWN and deploys it on one of the two PCs based on specific algorithms. In addition, based on 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 [28], [29].

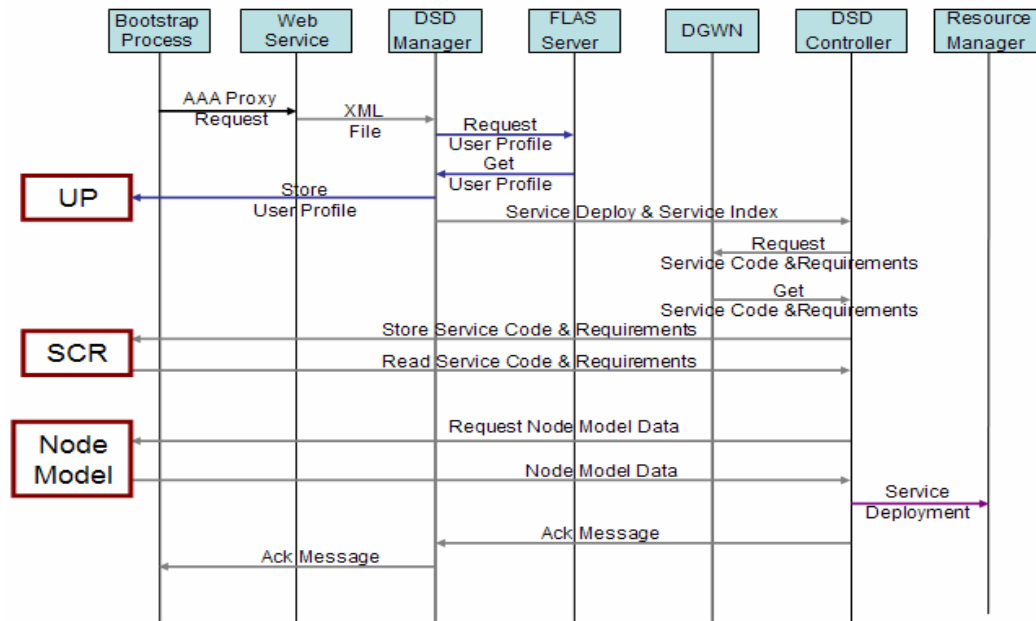


Figure 13. Message exchange during the deployment of the AAA Proxy.

The sequence of events for the deployment of the AAA proxy is illustrated in Figure 13.

- 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 FLAS Server, via a Web service, to download the User Profile.
- The DSD controller communicates with the DGWN, via a Web service, to download the code and the requirements for deploying 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 saying that the AAA Proxy has been deployed to the DSD manager, which in turn sends the acknowledgement to the bootstrap process.

10. EXPERIMENTAL RESULTS

Dynamic Service Deployment can be split into two time parameters: the time to download the new service and the time to install the new service. The service we tested for the DSD is the AAA Proxy, which is 21kB long. The AAA Proxy is downloaded from the DGWN. Table 1 shows measurements for DSD over Web Services.

Table 1: Dynamic Service Deployment over Web Services.

	Mean [sec]	Standard deviation [sec]:
Code download	2.8	0.34
Code deployment	0.8	0.09
Code configuration	1.3	0.10

Total	4.9	0.36
-------	-----	------

Figure 14 is a graphical representation of the results that occurred during the tests we performed.

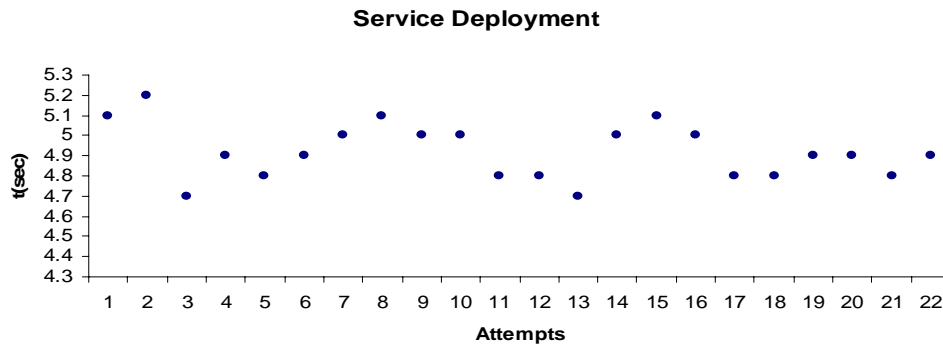


Figure 14. Dynamic Service Deployment.

Any delays are due to the fact that XML systems must in any case parse, process, and construct XML documents. According to the results FWAN is capable of deploying 12 services per minute.

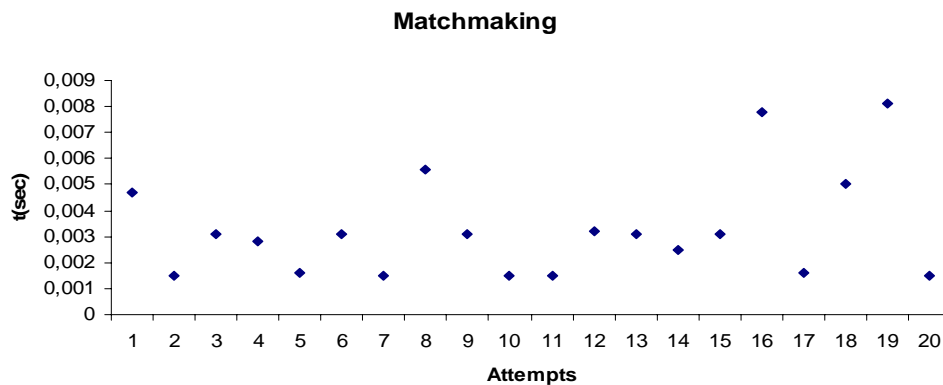


Figure 15. Matchmaking.

The matchmaking algorithm performs its procedure in a mean time of 3 msec (standard deviation is 2 msec) as shown in Figure 15.

Once a service has been deployed it does not need to be re-deployed, except for redundancy and load-balancing needs. Also, if an already installed service is requested to be re-deployed, the code will already be stored locally. In that case the time needed for the requested service to be deployed decreases to 2.1 sec, and FWAN is capable of deploying 28 services per minute.

The average time needed for a new service to be deployed according to the proposed architecture is slightly better than results that from other frameworks proposed. For example, the average time for the deployment of a new service in the Highly Available Dynamic Deployment Architecture (HAND) infrastructure [30] is 6.85 sec, so the HAND infrastructure is capable of deploying up to 9 services per minute. There is other on-going work, for example from Grid world [31] or from Autonomic Computing [32], but these efforts seem to move in a rather theoretical base, so there is no direct way to compare any experimental results.

We have also conducted experiments to see how long it takes for a new module to be fully added to our system, that is, how much time it takes to update the Node Model by adding all the new information of the new module(s) in terms of resources (available or no) and jobs it (they) can carry out.

According to our trials, it takes an average time of 40 msec (standard deviation is 0.8 msec) for a new module to be added and be fully operational. The average time for three new modules to be added and be fully operational is 54 msec (standard deviation is 1.3 msec); for 10 modules the average time required is 85 msec (standard deviation is 1.5 msec) (Figure 16). By fully operative we mean that the system is in a position to accept the deployment of a new service.

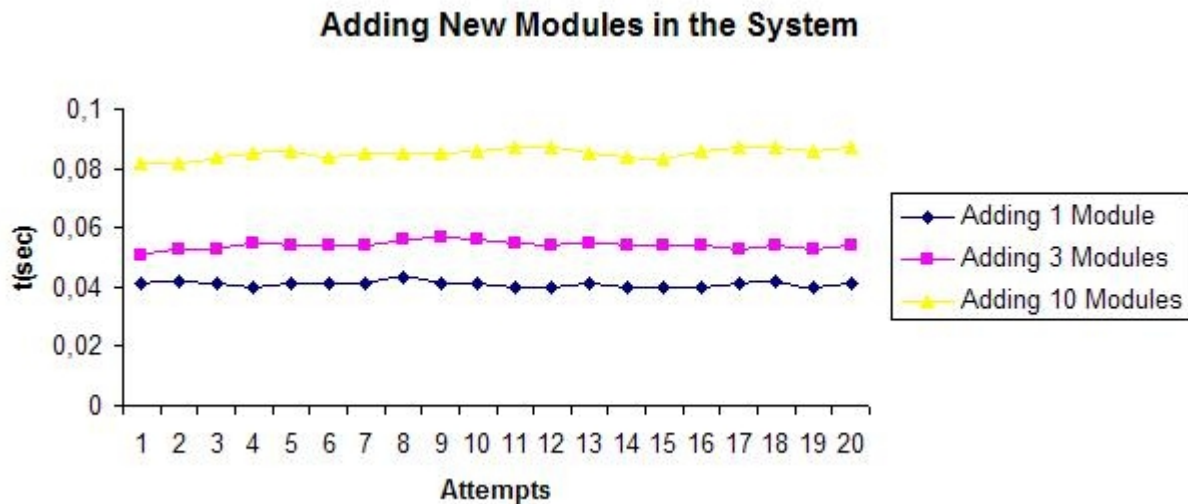


Figure 16. Adding new modules.

Taking into account these measurements it becomes clear that the time needed for a new module to be fully operational is 0.04 sec, whereas for three new modules it is 0.051 sec and for ten new modules 0.085 sec. These really tiny time differences — the difference between one and 10 new modules is only 0.045 sec — allows us to say that our distributed router approach offers significant benefits in terms of system scalability and performance.

11. CONCLUSIONS AND FUTURE WORK

Dynamic, heterogeneous and distributively-owned resource environments present unique challenges for resource representation, allocation and management. In this paper, we have introduced a resource management and a service-deployment 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 FWAN node.

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

Our component-based model addresses the issue of dynamic deployment of new services in a distributed environment and how these services 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 because we

not only addressed concerns related to Grid computing but also discussed architectural issues regarding Web-service configuration and deployment.

This paper has revealed the issues raised by Distributed Router modeling and implementation. We have presented a modular node architecture that achieves scalability of performance, functional flexibility, and reliability. Scalability is achieved by adding new modules having identical interfaces in an Extensible function block. New services can then easily be added by inserting modules that have the appropriate functionality. Another significant aspect of our model is that the failure in one module will not affect any of the other modules because they operate independently.

The implementation of the model regarding service deployment has now been realized, even though it requires further refinement and analysis. As future work we plan to provide a more sophisticated model for service deployment and selection based on QoS properties.

A significant goal of our future work is to incorporate preferences into the matchmaking algorithm. In this way more sophisticated algorithms will be able to cope with the possibility of more demanding services.

The effectiveness and usability of the architecture proposed have been successfully tested and verified within the FlexiNET IST project.

ACKNOWLEDGMENT

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

REFERENCES

1. Object Management Group. The *Common Object Request Broker: Architecture and Specification* (2.0 Ed.), July 1995. <http://www.omg.org/docs/formal/99-10-07.pdf> [9 August 2007].
2. Jae-Oh L. Enabling Network Management Using Java Technologies. *IEEE Communication Magazine* 2000; January; 38(1):116-123.
3. Distributed Management Task Force. Common Information Model (CIM), <http://www.dmtf.org/standards/cim/> [7 September 2006].
4. Halpern J, Delegates E. ForCES Forwarding Element Model. IETF ForCES working group draft, work in progress, October 2006 (draft-ietf-forces-model-07.txt).
5. Lopez-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*, 13-19 June, Dresden, Germany.
6. <http://www.w3.org/2001/04/20-ACLs> [January 2006].
7. <http://www.cs.cmu.edu/~softagents/retsina.html> [January 2006].
8. Stallings W. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, 3rd Edn., Addison-Wesley, Reading, MA, 1998.
9. Howes TA, Smith MC, Good GS. *Understanding and Deploying LDAP Directory Services*. 2nd Edn., Addison-Wesley Professional, 2003 .
10. Globus, MDS document <http://www.globus.org/toolkit/mds/> [9 August 2007].
11. Doria A, Haas, R, Hadi Salim J, Khosravi H, Wang W M. ForCES Protocol Specification. IETF ForCES working group draft, work in progress, July 2007 (draft-ietf-forces-protocol-11.txt).
12. Garsia-Molina H, Ullman JD, Widom J. *Database Systems: The Complete Book*. Prentice Hall, Upper Saddle River, NJ, 2002.
13. Scott B, Chamberlin D, Fernandez M, Florescu D, Robie J, Simeon J. XQuery 1.0: An XML Query Language. W3C, 2002.
14. CORBA Component Model, v3.0, OMG. <http://www.omg.org/technology/documents/formal/components.htm> [9 August 2007].

15. COM Component Object Model Technologies, Microsoft, <http://www.microsoft.com/com/default.mspx> [9 August 2007].
16. The CCA Forum. <http://cca-forum.org/> [9 August 2007].
17. 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*, April 2004, pp. 90-97.
- 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. Available at <http://www.globus.org/alliance/publications/papers/ogsa.pdf> [13 September 2007].
18. papers/ogsa.pdf [13 September 2007].
19. XSOAP toolkit. <http://www.extreme.indiana.edu/xgws/xsoap> [9 August 2007].
20. The Globus Alliance. <http://www.globus.org> [9 August 2007].
21. 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. pp. 11-18.
22. Newcomer E. *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Addison-Wesley, Boston, 2002.
23. Soldatos J, Alexopoulos D. Web Services-based Network Management: Approaches and the WSNET System. *International Journal of Network Management*, 2007, vol.17, pp.33-50.
24. Chrysoulas C, Kostopoulos G, Haleplidis E, Haas R, Denazis S, Koufopavlou O. A Decision Making Framework for Dynamic Service deployment. *15th IST Mobile & Wireless Communications Summit 2006*, 4-8 June, Myconos, Greece.
25. Hirata T, Mimura I. Flexible Service Creation Node Architecture and its Implementation. *IEEE Computer Communications Workshop*, 2003, pp. 166-171.
26. Chrysoulas C, Haleplidis E, Kostopoulos G, Denazis S, Koufopavlou O. A Distributed Router's Modeling and Implementation. In *Proceedings of 5th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2006*, Patras, Greece, pp. 288-292.
27. Haleplidis E, Haas R, Denazis S, Koufopavlou O. A Web Service- and ForCES-based Programmable Router Architecture. IWAN2005, 21-23 November, Sophia Antipolis, Nice, France.
28. Booth D, Canyang Kevin Liu. Web Services Description Language (WSDL) Version 2.0 Part 0: Primer. W3C Working Draft, May 2005, <http://www.w3.org/TR/2005/WD-wsdl20-primer-20050510/> [9 August 2007].
29. Kostopoulos G, Kavadias C, Chrysoulas C, Denazis S, Koufopavlou O. Security in Wireless Networks: The FlexiNET Approach. In *Proceedings of the 5th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2006*, Patras, Greece, pp. 606-610.
30. Qi L, Jin H, Foster I, Gawor J. HAND: Highly Available Dynamic Deployment Infrastructure for Globus Toolkit 4. In *Proceedings of the 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing, PDP '07*, 7-9 Feb. 2007, Naples, Italy, pp. 155-162.
31. Weissman JB, Kim S, England D. A Framework for Dynamic Service Adaptation in Grid: Next Generation Software Program Progress Report. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 3-8 April 2005, Denver, CO.
32. Diaz G, Achir N, Chen K. Modeling Data to Management Dynamic Services Deployment in Autonomic Networks. In *Proceedings of the 2nd IEEE International Conference on Information and Communication Technologies, 2006. ICTTA '06*, 24-28 April 2006, Damascus, Syria, Vol. 2, pp. 3416- 3420.

AUTHOR'S BIOGRAPHIES

Christos Chrysoulas received his Diploma in Electrical & Computer Engineering from the Electrical & Computer Engineering Dept., University of Patras, Greece in 2003. Since then he is working as a Researcher Engineer, in the Department of Electrical and Computer Engineering of the University of Patras. His research interests include Computer Networks, High Performance Communication Subsystems Architecture and Implementation, Wireless Networks, New Generation Networks Architectures, Resource Management and Dynamic Service Deployment in New Generation Networks and Communication Networks, Grid Architecture, Semantics Christos Chrysoulas has published more than 10 technical papers in these areas. He has also participated as Senior Engineer in European Research Projects.

Evangelos Haleplidis was born at Athens in July 1979. He graduated from the University of Patras, Department of Electrical and Computer Engineering in November 2002 with specialization in hardware design (VLSI). He started his Ph.D. in September 2003 in the same department. The focus of his Ph.D. is in open interfaces regarding network programmability. He participated on the successful EU IST project Flexinet.

Giorgos Kostopoulos received his Diploma in Electrical & Computer Engineering from the Electrical & Computer Engineering Dept., University of Patras, Greece in 2003. Since then he is working as a Researcher Engineer, in the Department of Electrical and Computer Engineering of the University of Patras. His research interests include Security in Wireless Networks, New Generation Networks Architectures, Security Management in New Generation Networks and Communication Networks. Giorgos Kostopoulos has published more than 15 technical papers and book chapters in these areas. He has also participated as Senior Engineer in European Research Projects.

Dr. Robert Haas is a networking expert with IBM Research since 1996 and he is now project leader active in the field of storage resiliency and security. He contributed to the invention and implementation of advanced functions for the IBM Network Processor, and completed his PhD in 2003 at ETH Zurich on automated service deployment in programmable networks. He is active with the IETF (Internet Engineering Task Force) and currently co-author of the ForCES (Forwarding and Control Element Separation) protocol specification and author of the corresponding MIB specification.

Odysseas Koufopavlou received the Diploma of Electrical Engineering in 1983 and the Ph.D. degree in Electrical Engineering in 1990, both from University of Patras, Greece. From 1990 to 1994 he was at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. He is currently Professor with the Department of Electrical and Computer Engineering, University of Patras. His research interests include computer networks, high performance communication subsystems architecture and implementation, VLSI low power design, and VLSI crypto systems. Dr. Koufopavlou has published more than 150 technical papers and received patents and inventions in these areas. He has participated as coordinator or partner in many Greek and European R&D programmes. He served as general chairman for the IEEE ICECS'1999.