A Policy-Based Management Architecture for Flexible Service Deployment in Active Networks

Yiannis Nikolakis[1], Edgar Magaña[2], Marcin Solarski[3], Alvin Tan[4], Epifanio Salamanca[2], Joan Serrat[2], Celestin Brou[3], and Alex Galis[4]

[1] National Technical University of Athens
Heroon Polytechniou 9, 15773 Athens, Greece
ynikol@telecom.ntua.gr
[2] Universitat Politècnica de Catalunya
Dept T.S.C. Campus Nord D4 c. Jordi Girona, 1-3 c.p. 08034 - Barcelona, Spain
{emagana, epi}@nmg.upc.es, serrat@tsc.upc.es
[3] Fraunhofer Institute FOKUS
Kaiserin-Augusta-Allee 31, 10589 - Berlin, Germany
{solarski, brou}@fokus.fraunhofer.de
[4] University College London
Dept of Electronic & Electrical Engineering, Torrington Place, London WC1E7JE, United Kingdom
{atan, a.galis}@ee.ucl.ac.uk

Abstract. This paper describes a dynamic, scalable and extensible policy-based network management (PBNM) system that is fully integrated with a service provisioning architectures for active networks. The key result is network customisation according to the needs of the different service providers and its end users. Our component-based service provisioning architecture enables us to render service- and user-specific requirements, across single/multiple administrative domains, at deployment time and to dynamically map service components onto the network using the corresponding management policies. The architecture presented in this paper describes the approach undertaken by the IST-FAIN research project as well as the main issues that we encounter in developing and integrating the PBNM with the service provisioning mechanism.

1 Introduction

Active and programmable networking technology introduces innovative approaches for enabling the user to set up the network configuration and its corresponding services dynamically, according to specific requirements. The dynamic nature of active networks and its continuously emerging new services, calls for efficient management mechanisms to configure the network according to the needs of the service providers and the services themselves, as well as to dynamically extend the management functionalities to adapt to the requirements set by new services.

N. Wakamiya et al. (Eds.): IWAN 2003, LNCS 2982, pp. 240–251, 2004.

The FAIN project [1], [2] is aimed at developing a fully managed, multi-EE prototype active node to support deployment and provisioning of component-oriented services. The FAIN programmable node is described in [3]. The FAIN network management architecture consists of the Policy-Based Network Management (PBNM) [4] and the Active Service Provisioning (ASP) [5] systems, which provide higher-level management and service deployment mechanisms built on top of an active network infrastructure.
In this paper we describe the final implementation of both the PBNM and ASP systems as well as the intercommunication mechanism developed in order to offer a flexible service deployment on active networks. We further introduce new PBNM components at the network-level that were developed to address the management aspects of end-to-end service creation and deployment, i.e., Resource Manager (RM), Service Manager (SM) and the Inter Domain Manager (IDM). For these purposes, the network-level ASP was developed, which enhances our node-level ASP [5] work. We also consider the case when the network management system is requested to deploy a service in a target node outside of its domain. The FAIN services and their needs are described using technology-independent (i.e., achieved via XML implementation) service descriptors and abstractions from the actual implementation details. The service requirements are used to create a Virtual Active Network (VAN), which is dedicated to provide the service with appropriate computational and communicational resources.
The paper is structured as follows: Section 2 provides an overview of the FAIN PBNM with emphasis on the component functionality; Section 3 presents the details of the ASP framework; Section 4 gives an explanation of the architecture and interaction use cases between the PBNM and the ASP frameworks; Section 5 illustrates an example scenario that successfully demonstrates our expected functionalities, while Section 6 gives a brief review of related work. Finally, Section 7 concludes the paper.

2 Policy-Based Network Management (PBNM) Architecture

The FAIN PBNM management architecture is designed as a hierarchically distributed architecture. It consists of two levels: the network management level (NMS) and the element management level (EMS). The NMS is the entry point to the management architecture. It is the recipient of policies, resulting from the ANSP's management decisions or service level agreements (SLA) between various categories of users. The enforcement of these SLAs requires reconfiguration of the network, which is automated by means of policies sent to the NMS. Network-level policies are processed by the NMS Policy Decision Points (PDPs), which decide when policies can be enforced. When enforced, they are delivered to the NMS Policy Enforcement Points (PEPs) that map them to element level policies, which are, in turn, sent to the EMSs. EMS PDPs follow a similar procedure at the element level. Finally, the active node PEPs execute the enforcement actions on the managed network nodes [6].

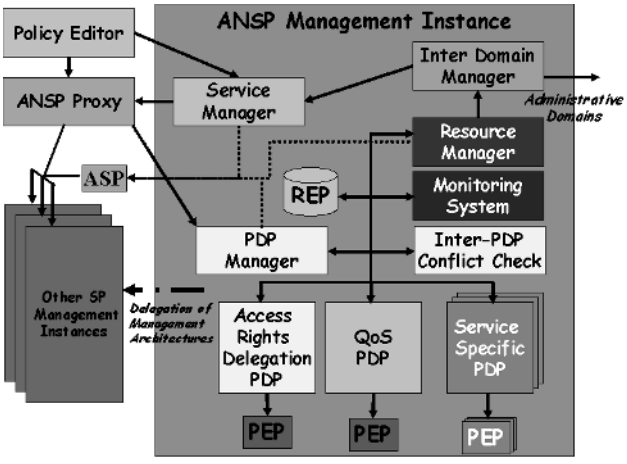

Fig. 1. FAIN management instances and their components

The defined policies are categorised according to the semantics of management operations, which may range from basic configuration and fault management operations to service-specific operations. Policies that belong to a specific category are processed by dedicated PDPs and PEPs. The three main actors of the FAIN Enterprise Model, the ANSP, SP, and the Customer, may request their own (virtual) management architecture through which they are able to manage the resources allocated to the Virtual Environments (VE) of their virtual network. The VE is an abstraction used only for the purpose of partitioning the resources of the active node so that different communities of users can stay isolated from each other.

This allows each actor to select and deploy his own management architecture freely, in order to manage its allocated resources, which can be policy or non-policy based. This model corresponds to a 'management instance' associated with the particular service provider.

The main components of the FAIN PBNM are illustrated in the Figure 1. Service Manager, Resource Manager and the Inter Domain Manager (IDM), are summarised in the table below. They have been developed in order to support service deployment, decision- making with regards to resources control, and inter-domain communication respectively. A detailed description of all components is given in [7].

3 Active Service Provisioning (ASP) Architecture

One of the key results of the FAIN project is an architecture for service provisioning called Active Service Provisioning (ASP). Like the PBNM, it is a two-layered system that comprises the network level and node level. An overview of its architecture is depicted in Figure 2 and has been described in [6] in detail. The network level functionality is responsible for finding the target nodes for deploy-

Table 1. Description of FAIN PBNM components as depicted in Figure 1

PBNM Component	Description
Service Manager	This component is responsible for setting up a VAN in response to a service request. It receives, as an input, the SLA that has been agreed between this domain and the one that requests it. It uses this information together with the topological requirements imposed by a service, which it retrieves from the ASP, to generate the relevant policies for the service. When a VAN is successfully created, the service manager instructs the ASP to trigger the deployment of the requested service.
Resource Manager	The Resource Manager maintains information about the nodes and links of the system and can compute possible end-to-end routes for a given service, based on the network topology and resource information obtained by the Monitoring System. It co-operates with the ASP system, in order to address service-specific requirements.
Inter-Domain Manager	It is in charge of implementing end-to-end negotiation of service deployment into separate active nodes that belong to different administrative domains, managed by different organisations.

ing a service by matching the network level service requirements against the actual network capabilities, coordination of the deployment process at the node level (i.e., Network ASP Manager), and providing a service code retrieval infrastructure (i.e., Service Registry where the service descriptors are kept and Service Repository where the code modules are kept). At the node level, the ASP identifies necessary service components by resolving node-level component dependencies and coordinates component installation, activation and pre-configuration in the execution environments located on the node. Active services that can be deployed using the ASP system have to conform to the FAIN service model presented in [5]. The work described in this paper extends the one presented in [6] by discussing the network-level deployment process. In this section, the deployment requirements are categorized taking into account their scope (network and node level) as well their lifetime characteristic (static and dynamic requirements).

3.1 Service Description

The FAIN service model used is component-based [5]. It allows structuring services using self-contained software units of deployment, management and computation, called service components. A service component may be associated with a code module that can be downloaded from the service repository and installed on an active node in an execution environment. This code module is executable in a particular execution environment and may have a number of dependencies/requirements on the resources needed for its execution.
The principal composition pattern is a component hierarchy, where service components may be recursively composed of sub-components. An example service conforming to the FAIN model is depicted in Figure 3.

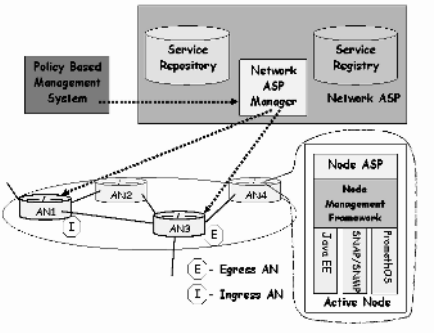

Fig. 2. The ASP overall architecture and relations to other FAIN components

From the deployment perspective, an active service, composed of a set of components, can be divided into a number of co-located component groups, each of which is to be deployed on a single host. Thus, deploying a compound active service means identifying groups of co-located components and deploying each of such groups on a node matching the deployment requirements of the component group. The deployment requirements of a service can be divided into two categories:

- the network level deployment requirements describing the requirements shared by all subcomponents of a top component; and
- the node level deployment requirements of each component within the group. These requirements include the execution environment and node resources needed for execution and they are specified with the help of node level service descriptor as described in [5].

A FAIN active service is described with a network-level service descriptor that specifies the 'top' service components which are abstract components (in the sense they do not have any code directly associated) grouping all service components to be deployed on the same node. Deploying a service on a network level means finding a mapping of the service top components onto the available active node in the network so that the network level deployment requirements are fulfilled by the selected nodes. The network level descriptor specifies the following items:

- the top service components, including references to node level descriptors of the subcomponents; and
- the static topological constraints of every top service component, which include partial information on the requested node location. In the current implementation, we differentiate node roles (ingress, intermediate and egress with respect to domain borders and the main service packet flow) and locations relative to other nodes, whose location is determined otherwise, for instance.

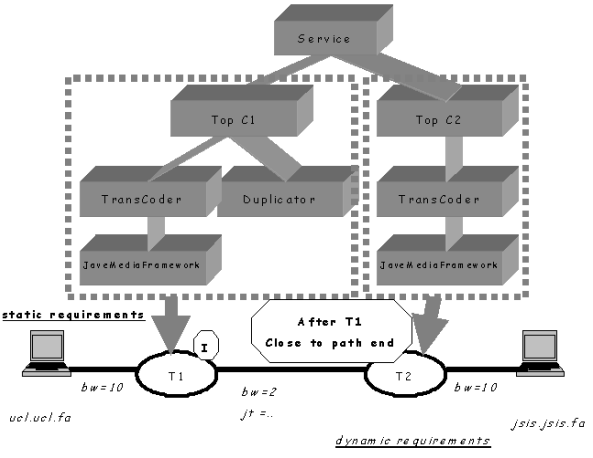

Fig. 3. Example description of a distributed FAIN service - component topology and its network deployment requirements

The network level requirements specified above are of static nature and do not change whenever the service is deployed. However, they are specified so that a number of physical network topologies may address them and thus do not allow fully determining the mapping of top components onto the physical network. To complete this mapping, additional requirements are specified. They can usually be determined only at deployment time and are called dynamic requirements because of this. It is the service deployer that specifies them when requesting a service deployment. These additional deployment requirements include QoS of the virtual links, for instance, the jitter or the bandwidth as well as the edge constraints on the mapping of the logical service topology onto the real network, like physical nodes/networks on which or close to which given top components have to be placed.

4 Integrated System Operation

Having designed and developed the PBNM and the ASP systems as individual entities, we further seek to obtain synergistic benefits by integrating both systems to establish a seamless, fully managed service provisioning mechanism.
Once an SLA between ANSP and SP has been agreed upon, the following steps are carried out when providing a service. Firstly, the SP (or a service itself) requests for a service to be provided with the given edge constraints and the PBNM, with the support of the ASP, computes a mapping of service components to the nodes of the active network. It considers the SP's constraints, the deployment requirements of the service and the actual resources in the active

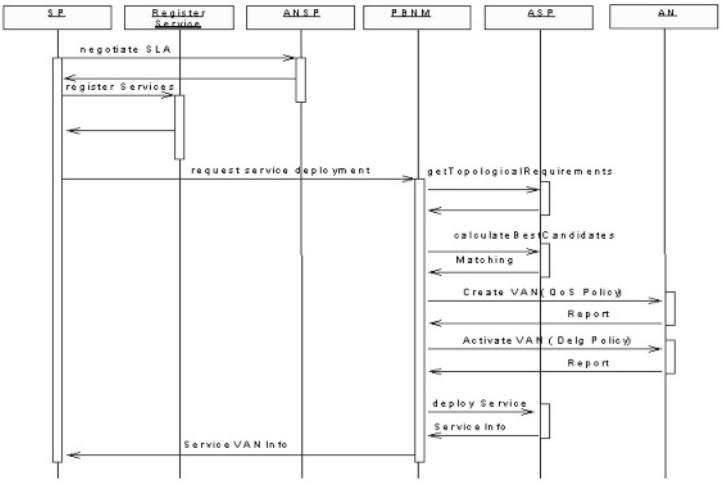

Fig. 4. System operation process

network. Secondly, once the PBNM has obtained the target nodes that make up the VAN, our management system creates a VAN that meets the service requirements. Thirdly, the PBNM requests the ASP to deploy the service in the activated VAN. Figure 4, shows the phases mentioned before, while details of these steps are described in the following subsections.

4.1 Service Requirements Matching

One of the core activities of our system is mapping the components of a given service to the available nodes in the network. The Network ASP Manager decodes the service topological requirements from the service network-level service descriptor, and the edge constraints from the service deployer. The RM receives the topological requirements and checks the resource availability along the possible routes between the two given service endpoints. Resources for a particular VAN are allocated in a 'hard' manner, so that the full requested amount of computational and communication resources must be available on all nodes and links of a route. If multiple paths satisfy the resource requirements, the RM submits the list of the candidate mappings, along with resource information to the Network ASP Manager.

The network-level ASP performs the requirements matching process on the node-level. It validates the deployability of the service according to each of the candidate mappings one after another, by checking that every service component can be installed on a pertinent node ASP for each identified node in a candidate VAN. The node-level determines whether a node-level deployment is successful by resolving the technological requirements of the service components. This

includes checking the availability of the execution environment type and computational resources needed by the service components. As a result of this mapping, the Network ASP returns a selected candidate VAN with service components assigned to the nodes that belong to this VAN.
We have adopted this two-phase process, in order to maintain a clear separation of tasks and responsibilities between the management and the service provisioning frameworks. An alternative solution would be to place all the necessary functionality in only one component, namely the Resource Manager, which would be able to consider both resource and service-specific requirements in one pass. In this way, however, the Resource Manager should also be able to access and interpret service-related information, required for the mapping of service components. Effectively, this would result into duplicating parts of the service provisioning functionality inside the management system and mixing up the scope of the two separate frameworks. The drawback in our approach is that for each candidate path, the corresponding resource information must be transmitted from the Resource Manager to Network ASP.

4.2 Virtual Active Network Creation

A VAN is specified by three sets of requirements: the QoS parameters (e.g., the required bandwidth), the computational parameters (e.g., the amount of memory) and the specific service requirements (e.g., the VE types).
The VAN creation process is started when an SP triggers the deployment of a service. The SP uses the corresponding GUI offered by the Policy Editor or by directly contacting the Service Manager (SM). The network-level management system (via the SM) receives the request and translates it into a set of policies, a network-level QoS policy that creates a VAN by allocating resources to it and a network-level Delegation policy that activates the VAN and delegates management of the allocated resources through the creation of a new management instance for this particular SP.

Service Resources Reservation. The determination of the VAN topology is done through the process described in section 4.1. The information about the new VAN is sent to the QoS PDP that integrates this information with other VAN requirements in a structure that is forwarded to the QoS PEP. Next, the QoS PDP forwards the decision to its corresponding PEP. The QoS PEP transforms the request into a set of appropriate element-level QoS policies (one policy for each of the active nodes that constitute the VAN) and it sends the policy to the EMS of the established nodes. Once the QoS policy is enforced, the EMS calls the reservation method within the active node, thus ending the reservation process. The internal active node interactions that guarantee the reservation of resources, are not within the scope of this paper, but can be found in more detail in [3].

Service Resources Activation. After the enforcement of the QoS policies has successfully terminated in all nodes, the PDP Manager starts processing the

delegation policies by forwarding them to the delegation PDP to be evaluated. A network level policy is translated into two element level delegation policies: one for assigning access rights to the VE and activating it (i.e., VE activation), and the other for creating the management instance (MI) inside the EMS so that the SP can manage its resources (i.e., MI creation).
A policy set is created and submitted to the respective EMSs. The policies are forwarded to the appropriate management instance, where they are processed sequentially. The first delegation policy causes the assignment of a security profile to the created VE, to be afterwards used for its activation. Since this policy must be enforced immediately, it is forwarded to the delegation PEP running inside the privileged VE in the active node. The delegation PEP enforces the policy that activates the new VE using the API offered by the active node.

Service Deployment. When the SM is made known of the correct activation of the network resources, it will ask the NetASP to perform the corresponding service deployment, indicating the kind of service required and the VAN identifier for the related customer. The network-level ASP iterates through the nodes of the final VAN and contacts the node level ASP on each of these nodes. The node ASP may perform the installation and configuration of the service components identified during the mapping process. To perform the actual installation, the ASP makes use of the node management framework, which provides a unified CORBA-based platform for different types of execution-environment. Through this interface, it is possible to access the EE capabilities and isolate the deployment and management code. The deployment process ends after all service components have been deployed on the target nodes.

4.3 Inter-domain Management

The role of the inter-domain manager (IDM) comes in when ANSPs need to propagate their SPs' traffic across each other's administrative domain. A successful negotiation will lead to a request to create a VAN between the two domains. As such, ANSPs need to derive contracts with other ANSPs to encompass the geographical spread of their targeted client base. For example, in Figure 5, the administrator in Domain 1 needs to establish a relationship with Domain 2 in order to reach its customers.
When the network management system is requested to deploy a service that involves installation of the code in a target node outside of its domain the IDM contacts a corresponding IDM of the other domain in order to negotiate the service deployment in one or more of their active nodes and lets the SP of the first domain to use them. The role of this component is highlighted for reservation of computational and communicational resources for service deployment, particularly as requests occur between different administrative entities.
In our design we have established a repository that maps a destination address to an IDM, i.e., a 'many-to-one' relationship. As such, an ANSP must register a list of destination addresses within its domain on a repository that has well-

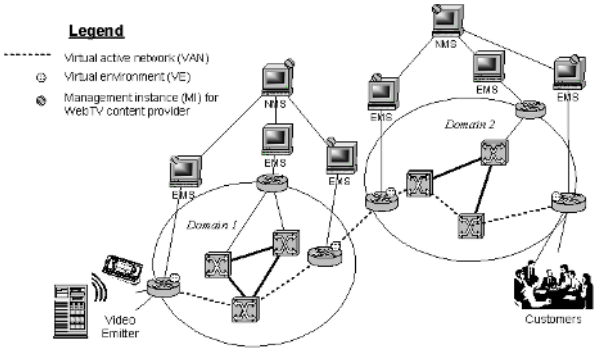

Fig. 5. A view of multiple domains, using the WebTV scenario as an example

known address, so that this repository can provide a discovery mechanism for mapping destination addresses to their respective IDM.

5 Example Scenario

A demonstration of the overall architecture has been carried out over the distributed FAIN testbed, which interconnects several different sites across Europe, using IP tunnels over the public Internet.

According to the demo scenario, an SP is offering a WebTV service, emitting a video stream in MPEG format. Two customers subscribe to the service. The first one uses a device with limited bandwidth, which is not in position to receive the video stream as it is emitted by the WebTV application and requires a H323 format as input. For this reason a transcoder service will have to be installed in the active node placed closest to the customer, to convert the video stream to a format understandable to the receiver. The second customer connecting to the WebTV service does not have the limitations of the first one. However the service is not using a multicast protocol and thus a duplicator service will have to be installed after the video source, which will replicate the multimedia data into a new flow, targeted to the second client.

As mentioned before, the first client has bandwidth limitation and for this reason, it is necessary that the PBNM knows the QoS parameters of the user by QoS policies. As the customers are not expert users, they are classified in three QoS categories, i.e., Gold, Silver and Bronze. These categories involve several QoS parameters:"if User_Category = Silver then Priority = 5 and Bandwidth = 50 end" , etc.

When the PBNM decides which QoS parameters apply, it transforms the high-level policies into element-level policies; the new policies are distributed to the

targeted active nodes (where the service will be installed) and the VAN is created. After that, the delegation phase willbe executed and the VAN is prepared to receive service components. For this reason the management system requests the ASP to deploy servicecomponents, as we have explained in the earlier section. Finally the customers have a service-integrated VAN, which supports the transcoderand duplicator services in order to receive the corresponding video stream from WebTV server. The complete WebTV service scenario is explained in [8].

6 Related Work

The VAN [9] framework introduces the concept of the Virtual Active Network, which can be established on-demand for a particular customer, offering a separate service management interface. Our architecture can be seen as an extension of this work.
The Tempest [10] project provides a programmable network infrastructure, with inherent support for virtual private networks. However Tempest is restricted to the partition of resources and lacks service deployment capabilities.
The Darwin [11] project uses the notion of virtual meshes to describe the virtual networks that encompass the network and computational resources allocated and managed to meet the needs of service providers or applications, providing strong resource control mechanisms. In our approach we have also adopted the use of a component-based service model, which provides greater flexibility in the deployment of new services.
The autonomic service deployment in programmable networks [12] allows deploying component-based services in large-scale networks. The deployment process is executed in two phases, corresponding to the network and node level deployment in our approach. Unlike our approach, the interactions between deployment and management frameworks are not considered.

7 Conclusions

In this paper we have described the approach adopted by the FAIN project for the flexible support of service provisioning, according to the customer-specific requirements. In our view, network management and service deployment are closely related and the corresponding support systems should interoperate when deploying a service. As such, we integrated these two aspects by means of incorporating the PBNM and the ASP systems. The management system is responsible for setting up a suitable network topology and reserving the resources required for the smooth operation of a service, also considering an interdomain situation. The service provisioning system deals with the service-specific requirements and is involved in the process of mapping the service to the network, while it also deploys the appropriate service components to the selected target nodes. The architecture presented has been developed and tested in the framework of the FAIN project. The future work will focus on providing improved support

for service reconfigurability. Deployment algorithms and more optimised target environment selection algorithms will also be investigated.

Acknowledgements. This paper describes work undertaken and in progress in the context of the FAIN - IST 10561, a 3-year project from 2000 to 2003, which is partially funded by the Commission of the European Union. The authors would like to acknowledge all FAIN partners for the fruitful discussions that have taken place within this project.

References

1. Galis, A., B. Plattner, J.M. Smith, S. Denazis, E. Moeller, H. Guo, C. Klein, J. Serrat, J. Laarhuis, G.T. Karetsos, C. Todd: "A Flexible IP Active Networks Architecture" International Working Conference on Active Networks (IWAN2000), 16-18 October 2000, Tokyo, Japan
2. FAIN Project WWW Server http://www.ist-fain.org.
3. FAIN Deliverable D7 "Final Active Node Architecture and Design" May 2003 - http://www.ist-fain.org
4. Salamanca, E., E. Magaña, J. Vivero, A. Galis, B. Mathieu, Y. Carlinet, O. Koufopavlou, C. Tsarouchis, C. Kitahara, S Denazis and J. L. Mañas: "A Policy-Based Management Architecture for Active and Programmable Networks" IEEE Network Magazine, Special issue on Network Management of Multiservice Multimedia IP-Based Networks, May 2003, Vol. 17 No.3
5. Solarski, M., M. Bossardt, T. Becker: "Component-based Deployment and Management of Services in Active Networks" IWAN 2002, Zürich, CH, Dec. 2002
6. FAIN Deliverable D8 "Final Specification of Case Study Systems" May 2003 - http://www.ist-fain.org
7. Galis, A., S. Denazis, C. Klein, C. Brou (eds.): "Programmable Networks and their Management" Artech House Books (www.artechhouse.com), ISBN: 1-58053-745-6 (commission for publication in 4th Quarter 2003)
8. FAIN Deliverable D9 "Final Specification of FAIN Scenarios" May 2003 - http://www.ist-fain.org
9. Brunner, M., R. Stadler: "Service Management in Multi-Party Active Networks" IEEE Communications Magazine, Vol. 38(3), 2000
10. van der Merwe, J.E., S. Rooney, I.M. Leslie, S.A. Crosby: "The Tempest - A Practical Framework for Network Programmability" IEEE Network, Vol 12, Number 3, 1998
11. Gao, J., P. Steenkiste, E. Takahashi, and A. Fisher: "A Programmable Router Architecture Supporting Control Plane Extensibility" IEEE Communications Magazine, March 2000
12. Haas, R., P. Droz, B. Stiller: "Autonomic service deployment in networks" IBM Systems Journal, Vol. 42, No. 1, 2003