A Policy-Based Management Architecture for Active and Programmable Networks

Christos Tsarouchis and Spyros Denazis, Hitachi Europe Ltd.
Chiho Kitahara, Hitachi Ltd.
Julio Vivero, Epi Salamanca, and Edgar Magaña, Universitat Politècnica de Catalunya
Alex Galis, University College London
Juan Luis Mañas, Integrasys SA
Yannick Carlinet and Bertrand Mathieu, France Telecom
Odysseas Koufopavlou, University of Patras

Abstract

Next-generation networks must be capable of supporting a multitude of service providers that exploit an environment in which services are dynamically deployed and quickly adapted over a common heterogeneous physical infrastructure, according to varying and sometimes conflicting customer requirements. In this context, network management must become more flexible in order to cope with these emerging conditions. More specifically, new management architectures must offer service providers the freedom to manage their services according to their own policies and seamlessly extend management functionality as the only way to react to the introduction of new services. Based on a new business model that describes such an environment, we propose a policy-based management architecture that is extensible and operates in an active and programmable network. This management architecture is part of a new network architecture that was developed in the FAIN European Union research and development IST project.

In the world of networking we are experiencing a significant paradigm shift resulting in new technologies and architectures. The motivation behind this shift is the still elusive goal of rapid and autonomous service creation, deployment, activation, and management resulting from new customer and application requirements. Research activity in this area has clearly focused on the synergy of three concepts: network virtualization, open interfaces and platforms, and increasing degrees of intelligence inside the network.

Management, as a key component of a network architecture, must also be considered and designed around the same concepts. To this end, the management architecture must support the coexistence of different management strategies, facilitating customization and interoperation with different vendors' equipment. Management must also be dynamically extensible to support the deployment and operation of new services.

In this article we describe the management aspects of a new network architecture designed and implemented as part of the *Future Active IP Networks* (FAIN) European Union R&D IST project [1]. The main objective of the FAIN project is to develop an active network (AN) architecture oriented toward dynamic service deployment in heterogeneous networks. This architecture encompasses the design and implementation of active nodes that support different types of execution environment, policy-based driven network management, and a platform-independent approach to service specification and deployment. The architecture is deployed and evaluated in a pan-European testbed.

The FAIN management architecture encapsulates the three aforementioned concepts and is built in accordance with the Internet Engineering Task Force's (IETF's) policy-based management framework [2], used in an active network environment. As a consequence, it inherits the features of this enabling technology, which are then applied to this new problem space.

We briefly introduce the FAIN business model on which the management architecture is based. We give an overview of the FAIN Policy-Based Network Management (PBNM) architecture and its objectives. We describe all the architectural components and the functionality thereof, while we then provide details of its implementation. We provide our conclusions, identify some open issues, and propose future work.

The FAIN Business Model

The FAIN management architecture is the realization of the business model proposed in FAIN. Accordingly, a brief description of the most important actors and the relationships thereof of the FAIN business model is essential to understand the motivation and objectives of the FAIN management architecture.

The main actors of the business model are the active network service provider (ANSP), the service provider (SP), and the consumer (C).

The ANSP is the primary owner of the network resources and provides facilities for the deployment and operation of the active components in the network. The ANSP offers both basic network resources, of which it is the primary owner, and secure access to facilities for the deployment and operation of the active components in the network. The whole offering takes the form of a virtual network, available to potential customers such as SPs or large corporate customers, network operators may play the role of ANSP.

The SP buys network resources from the ANSP and creates services comprising active components delivered by a service component provider. It then deploys these components in the network, and offers the resulting service to Cs.

The C is the end user of the active services offered by an SP. A C may be located at the edge of the information service infrastructure (i.e., be a classical end user) or it may be an Internet application, a connection management system, or even another SP.

FAIN has focused mainly on the relationships and interactions between ANSP and SP, and SP and C with respect to service deployment and management.

The FAIN PBNM Management Architecture

The FAIN PBNM management architecture is designed as a hierarchically distributed architecture consisting of two levels (two-tiered architecture): the network management level, which encompasses the network management system (NMS), and the element management level, which encompasses the element management system (EMS).

Furthermore, the defined policies have been categorized according to the semantics of management operations, which may range from QoS operations to service-specific operations. Accordingly, policies that belong to a specific category are processed by dedicated policy decision points (PDPs) and policy enforcement points (PEPs) (Fig. 1).

The NMS is the entry point of the management architecture. It is the recipient of policies that may have been the result of network operator management decisions or service level agreements (SLAs) between ANSP and SP, or SP and C. These SLAs require reconfiguration of the network, which is automated by means of policies sent to the NMS.

Network-level policies are processed by the NMS PDPs, which decide when policies can be enforced. When enforced, they are delivered to the NMS PEPs that map them to element level policies, which are in turn sent to the EMSs. EMS PDPs perform similar processes at the element level. Finally, the AN node PEPs execute the enforcement actions at the NE.

The use of this *policy control configuration model* [2] and its use in a hierarchically distributed management architecture combines the benefits of management automation with reduction of management traffic and distribution of tasks.

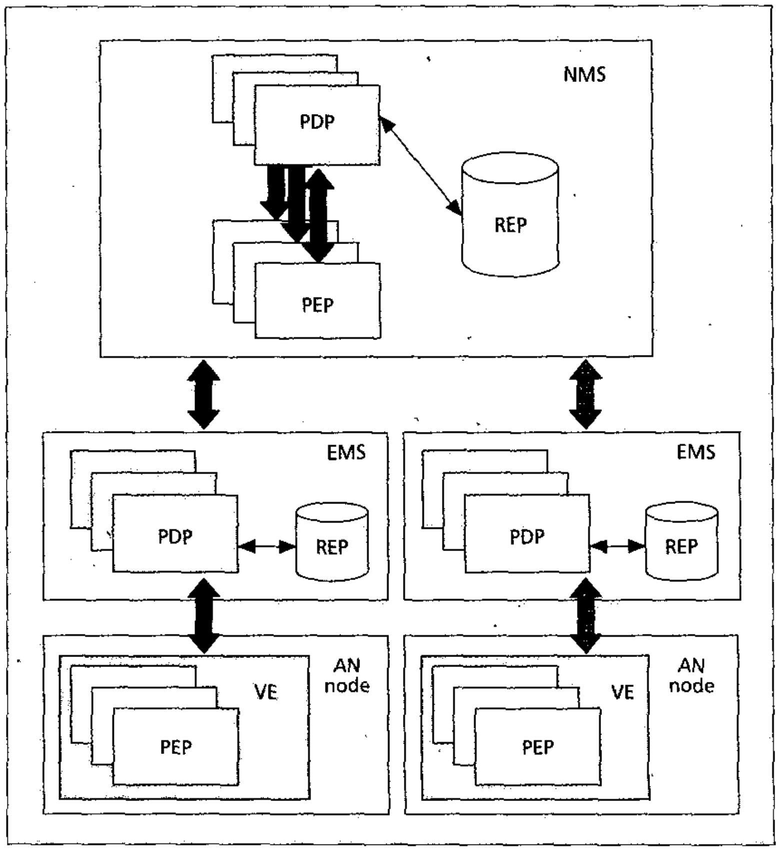
As the FAIN management architecture is based on the FAIN business model, the relationship among the three main actors (ANSP, SP, and C) is projected directly onto the architecture. Accordingly, each of these actors may request and get its own (virtual) management architecture through which it is enabled to manage the resources allocated to the virtual environments (VEs) of its virtual network (Fig. 1).

In this way, each actor is free to select and deploy its own model of managing the resources (its own management architecture), which can be centralized, hierarchical, policy- or non-policy-based. The complexity of the virtual network and the types of service deployed in it dictate the particular choice of management architecture by its owner. In addition, different management architectures simultaneously coexist in the same physical network infrastructure as they may be deployed by different actors. To this end, we create an environment capable of accommodating opposing requirements, an accomplishment beyond the capabilities of the traditional approach of monolithic architectures.

Our model extends the Tempest approach [3] to the management plane; it was the first to advocate the simultaneous support of (virtual) control architectures for asynchronous transfer mode (ATM) networks.

It also extends the scope of management by delegation (MbD) [4] as it allows delegation of the network management responsibility to a third party (e.g., an SP) that can be deployed and hosted in a separate physical location from the NMS of the owner of the network (e.g., the ANSP).

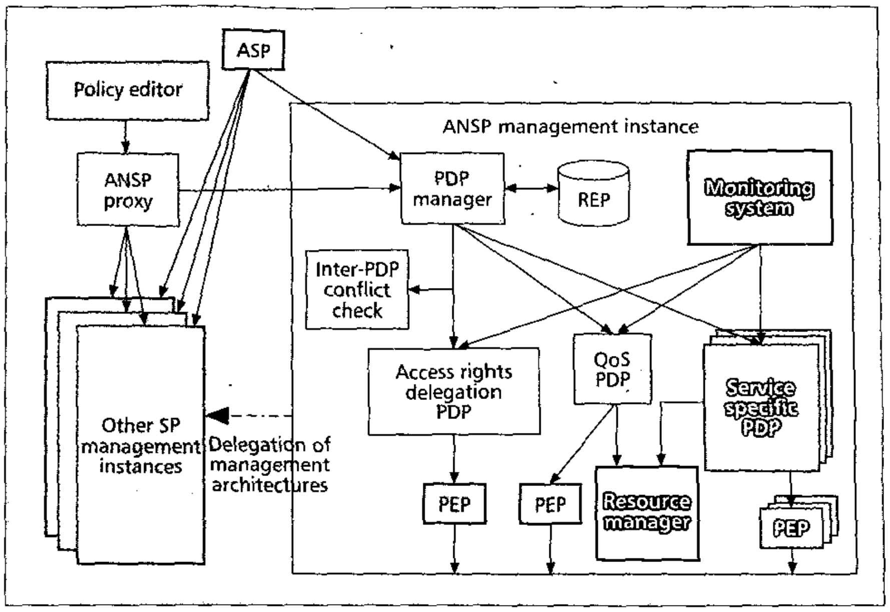
Figure 2 illustrates the above discussion. Starting with the management architecture of the network operator, the ANSP, it instantiates and registers a new management instance (MI), which is delegated to one of its customers (i.e., the SP). This management instance will host the SP's management architecture. The SP has the option to buy from the ANSP an instance of the ANSP's architecture, in our case a policy-based one. To this end, the network management architecture developed by the ANSP not only is used for managing the network elements (NEs) but becomes a commodity, thus creating another important source of income for the ANSP.



■ Figure 1. *The hierarchical FAIN management architecture.*

Furthermore, the ability of the ANSP to generate and support multiple management domains may create additional business opportunities. For example, the ANSP may build an operations sytems service (OSS) hosting facility for SPs to instantiate their own management architectures. In this way, the ANSP may sell both its expertise in running and operating an OSS as well as the architecture and its corresponding implementation.

In contrast, the SP does not need to build its management architecture from scratch but can customize an existing one according to the services it intends to run. Alternatively, the SP may deploy its own management architecture using the OSS hosting facility provided by the ANSP, thus reducing the cost of managing the network.



☐ Figure 2. *FAIN management instances and their components.*

In FAIN we have focused on and experimented with the automated instantiation of management architectures using as a blueprint the PBNM system of the ANSP to instantiate another management system for the SP. Note also that this instantiation relationship can be recursive in the sense that the SP may further delegate its own instances to a C.

Finally, the architecture of the MI used by the ANSP has been designed in such a way that it is dynamically extensible in terms of its functionality, as a result of using AN technology.

The ANSP's management architecture can be extended in two distinct ways:

- Deployment of a whole new pair of PDP/PEPs that implement new management functionality
- Extension of the inner functionality of existing PDP/PEPs

The former is triggered by the PDP manager, whereas the latter is achieved by the PDPs themselves. The execution of the extension, fetching and deploying the requested functionality, is the responsibility of FAIN's active service provisioning (ASP) system [5].

One important assumption underlying the previously described virtual management architectures is that well established open interfaces and protocols have to be provided by the NEs. This may seem from the outset to be a demanding condition, but there is convincing evidence of a strong push toward ubiquitous open interfaces. Initiatives like IEEE P1520 and lately the IETF ForCES working group serve as proof of such claims. Furthermore, the programmable and active networks paradigm also relies on similar assumptions [6].

The FAIN PBNM Management Components Description

We proceed now to present the details of the FAIN policy-based management architecture. Following the FAIN business model, the first instance of the management architecture that is created is that of the ANSP. As the NMS and EMSs of the ANSP instance have similar functionality and components, we focus on the NMS and, wherever applicable, note the differences between them.

Policy Editor

The policy editor exists only at the network level. It offers a GUI and a toolset in the form of templates and wizards for the composition of policies. These are generic enough to accommodate different types of policy, thus exploiting the architecture's extension capabilities.

ANSP Proxy

Policies originating at the policy editor are sent to the ANSP proxy. This has been introduced to enhance the security of the ANSP and/or its customers, the SPs. It provides authentication and authorization of the incoming requests (policies) and finds the MIs to which the policies must be forwarded.

Inter-PDP Conflict Check

Conflicts may appear when a policy arriving at one PDP clashes with a contradicting policy processed by another PDP. Conflicts [7] may be distinguished between *syntactic* conflicts that can be resolved with policy syntax analysis and *semantic* conflicts that are more difficult to track and resolve.

The inter-PDP conflict check component was introduced in order to process complex policies that capture inter-PDP semantics in a hierarchical manner, thus reducing the risk for semantic conflicts.

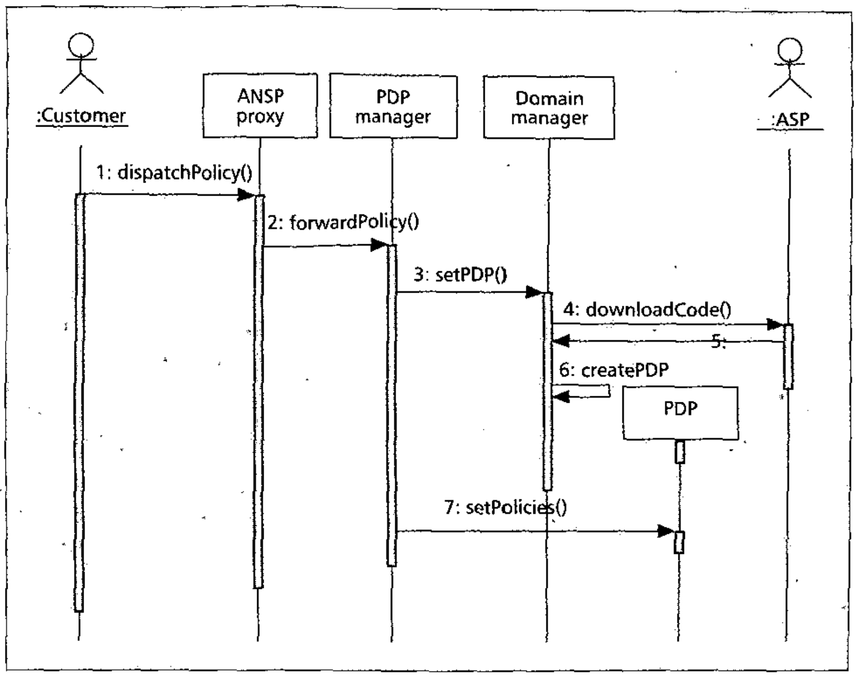
The description of the conflict-checking algorithm is currently work in progress in the FAIN project and is therefore not included in this article.

PDP Manager

The PDP manager receives policies and dispatches them to the appropriate PDPs. If the corresponding PDP is not installed, the PDP manager requests its download and installation from the ASP [5] framework developed by the FAIN project, thereby extending the management functionality of the system when needed.

The sequence diagram in Fig. 3 illustrates the first method of extending the management architecture mentioned earlier. The domain manager, a subcomponent of the PDP manager, is responsible for interacting with the ASP and instantiating the new PDP. Once the new PDP is deployed, the PDP manager forwards the policy to it.

The PDP manager also acts as a finite state machine for coordinating the whole policy installation procedure. An example of this coordination is when an SP requests the instantiation of a new virtual network and its corresponding MI. As a result, the PDP manager receives two different types of policy, the QoS policy and the access rights delegation policy. It then first installs the QoS policy, an action that requires admission control; only if there are resources available does it attempt to install the delegation policy. When both installations are completed successfully, it instantiates the new MI (Fig. 3) and hands it over to the SP. The entity inside the PDP manager responsible for the instantiation of the new MI is the domain manager.



■ Figure 3. *Dynamic installation of a PDP.*

Different Types of PDPs

Our architecture accommodates different types of PDP, each making decisions that apply to a specific context: QoS PDP, delegation of access rights PDP, and service-specific PDPs. They all perform conflict checks that are meaningful within their decision context (intra-PDP). In order to reach a decision, they also interact with other components that assist the PDPs in making a decision (e.g., a resource manager for admission control).

In addition, each PDP contains at least two types of component: the condition and action interpreters. These components provide action and condition processing logic for the policy types handled by the PDP. Each PDP has at least one instance of each type, but they can be dynamically extended to accommodate more interpreters capable of processing new actions and conditions conveyed by the policies. This represents the second extensibility method mentioned earlier.

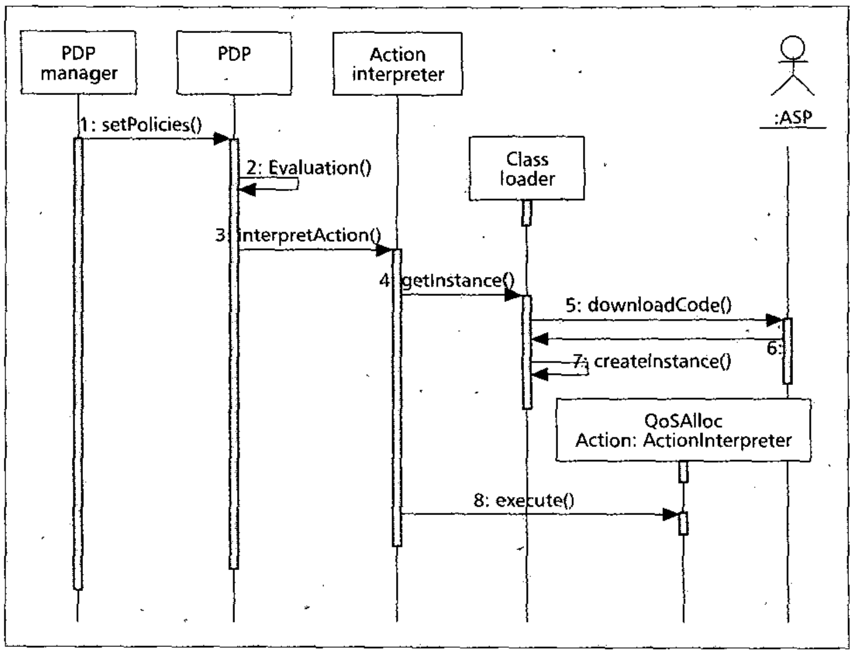
Figure 4 illustrates the sequence of events that take place when a new action interpreter is deployed. A generic action interpreter, acting as manager, becomes the recipient of all action requests carried by the policy. If there is an appropriate action interpreter already deployed in the PDP, the generic action interpreter forwards the request for further processing. Otherwise, it contacts the ASP in order to retrieve the action interpreter capable of processing the particular request.

In what follows we look more closely at the specific types of PDP used in the FAIN management architecture.

QoS PDP — The QoS PDP is responsible for analyzing QoS policies. Specifically, it:
- Decides when a policy should be enforced
- Forwards decisions to PEP components in order to be enforced
- Accepts requests that come from PEPs
- Controls the policy-validity period in order to uninstall expired policies

To realize this functionality the QoS PDP interacts with the monitoring system and resource manager components.

Access Rights Delegation PDP — By access rights delegation policies [8] we mean those policies that specify to what extent an actor is permitted to access network resources through the control interfaces provided. In this way, by controlling access to resources, the operations on them are also controlled, eventually restricting the capabilities of services that are deployed.



□ Figure 4. *Dynamic installation of an action interpreter.*

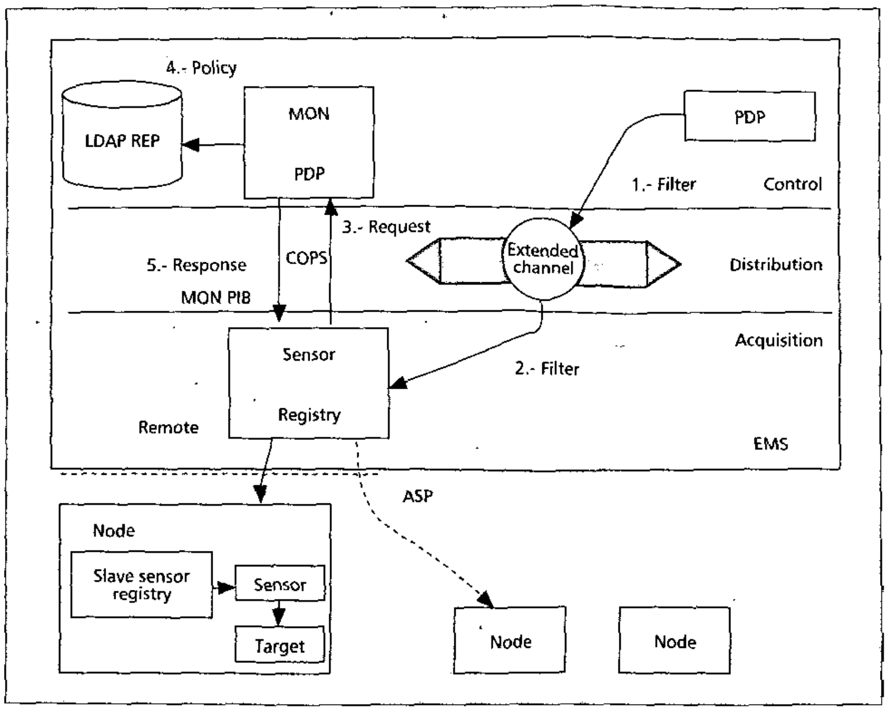


Figure 5. *The FAIN monitoring system.*

In FAIN, the delegation PDP is used by the ANSP to determine what operations the SP's services are allowed to carry out on those network resources assigned to the SP as part of its virtual network creation.

Delegation of access rights policies involves the (re)configuration of the security components of those nodes that form the topology of the SP virtual network. Every request to use a particular interface is checked by the security component. Access is only granted to authorized entities.

Other Service-Specific PDPs — The FAIN management architecture is designed to accommodate new PDPs that participate in service-specific decisions. These service-specific PDPs may be deployed on demand, using the same extensibility mechanisms described earlier.

Policy Enforcement Points (PEPs)

Each type of PDP has its own PEP counterpart. Network-level policies are translated by the network-level PEP into element-level policies, and then sent to the corresponding element PDPs that reside in the EMSs. This translation should take into account *topological information* about the location of the EMS PDPs. In other words, policy translation should be associated with information on where policies should be distributed.

Similarly, element-level PEPs enforce the element-level policies sent by the EMS PDPs by mapping them onto the correct FAIN node open interfaces. The use of open interfaces allows all PEPs across the network to share the same view of nodes' control interfaces, making them node (platform) independent.

Monitoring System

Policy decisions rely on both local and global network state information. While PEPs are the primary source for gathering device-specific data, a monitoring system is required in order to construct and maintain an overall picture of the network state. In an active network like FAIN, where new modules, service components and resource abstractions are constantly added, and multiple players run their own (virtual) networks competing for resources with each other, the task of maintaining a real-time picture of the network is very challenging.

The FAIN monitoring system is composed of three layers, introduced in order to distinguish the different types of monitoring operation (Fig. 5). The acquisition layer collects and processes data coming from entities residing in the active nodes through resource abstraction interfaces. The distribution layer supports efficient delivery of such information to the PDPs through an extended notification channel. Finally, the policy-based control layer makes decisions that pertain to the way the monitoring operations are carried out.

All three layers apply a set of strategies that aim to extend the functionality of the monitoring system thereby adapting to network changes. Whenever a PDP needs to monitor a new target, it submits an event subscription filter to register its interest. The distribution layer passes this filter to the acquisition layer, where a sensor registry analyzes it in order to identify the appropriate sensor to attach to the target. If no such sensor is previously available, it contacts the monitoring PDP, which examines the monitoring policies and returns a response with the appropriate configuration data.

The monitoring system, with the assistance of the ASP, deploys the requested sensor and attaches it to the corresponding target. Finally, in accordance with its configuration, the sensor attaches itself to the notification channel and sends the information to the interested PDPs.

This approach allows the monitoring system to dynamically extend itself by seamlessly integrating new monitoring components, or by enhancing existing monitoring methods through new configuration policies.

Management System Implementation and Evaluation

Overview of the Implementation based on a Scenario

In the service scenario we have selected for implementation and experimentation, an SP offers customers (end users) a WebTV service (Fig. 6). The WebTV SP multicasts a video program over the Internet that can be viewed by end users, irrespective of their terminal transcoding capabilities.

Although multicast technology is usually used with video applications, the scenario chosen in FAIN (WebTV) does not use multicast as a transport protocol. We have made this choice in order to demonstrate that multicast can be deployed as an active service as opposed to a protocol-driven approach. Indeed, we have developed an active service, named *duplicator*, to offer the multicast functionality in the active network where and when it is needed.

The WebTV SP requests the ANSP to set up an active virtual private network (AVPN) wherein it can deploy services that may be further adapted to customer requirements. Customers then subscribe to this WebTV service by directly contacting a WebTV SP server.

In this context, one of its customers uses a terminal that is not capable of correctly displaying the video stream broadcast by the WebTV SP due to a format mismatch. To adapt to this customer requirement, the WebTV-SP selectively deploys the compatible audio/video transcoder in the network so that the video stream received by the customer's device has the appropriate format.

We now describe in detail the scenario interactions pertaining to the management operations. As a result of the SLA agreed between the ANSP and the WebTV-SP, policies are sent to the ANSP's MI. The instantiation of the WebTV SP's MI is the successful outcome of a two-phase process coordinated by the ANSP's PDP manager: creation of all VEs that constitute the WebTV SP virtual network by reserving the necessary resources, followed by activation of the VEs (i.e., use of the reserved resources). The first phase is triggered by the QoS policy, and involves the ANSP's QoS PDP, while the second is triggered by the delegation policy and involves the ANSP's delegation of access rights PDP. Upon completion, instantiation of the WebTV SP's MI in all the appropriate NMS and EMS stations is carried out by the ANSP according to a configuration policy; then the ANSP yields control of the MI to the WebTV SP.

The monitoring system may initiate reconfiguration when, for instance, the access bandwidth drops dramatically and the end user needs different transcoding of the video stream.

Implementation in the FAIN Testbed and Initial Trials

The prototype implementation is deployed on the pan-European FAIN testbed, an overlay network connecting 10 different sites. Initial trials have focused mainly on functional evaluation of our management system, and in particular on the creation and usage of MIs and their extensibility features.

The interfaces are defined in Interface Definition Language (IDL) and implemented in Java. We have used CORBA as our middleware technology as implemented by the open source package OpenORB [9]. The PDP manager, ANSP proxy, PDPs, and PEPs are instantiated as CORBA objects and then registered using the CORBA naming service. With this approach we show the seamless nature of performing management even if the EMSs and NMS are physically located at different places.

In particular, CORBA services play a fundamental role in supporting our distributed architecture, as in the CORBA notification service, which binds the monitoring modules to the rest of the system. Additionally, the CORBA dynamic invocation interface and interface repository are the underlying technologies used to perform dynamic invocations and analyze returned values.

Finally, the event model is defined based on the Common Information Model (CIM) [10] in order to unify all information models used in the management architecture. In our implementation, the QoS PDP subscribes to events that are triggered by excess resource usage in active nodes, whereas the delegation PDP subscribes to events that are triggered by malicious accesses against active nodes. Following the IETF PCIM model [11] allows us to create new properties by extending existing classes of the model, thereby supporting extended QoS functions and newly introduced delegation functions in the FAIN framework. For our prototype implementation, we have created, using XML, two kinds of policy, a QoS policy and a delegation policy, that control VE configuration.

Initial Performance Trials

We have also carried out initial performance trials with the objective of assessing the viability of the new features of our management architecture. Accordingly, we have focused on creating MIs on behalf of different SPs, and on the two types of extensibility engineered in these MIs: extending their capabilities by dynamically deploying new PDPs and extending already deployed PDPs.

For this purpose, we have used one PC with an Intel 1.5 GHz Pentium IV and 500 MB of RAM. This PC runs not only the EMS but also the rest of the active node functionality the EMS manages. The first MI instance, created as part of the bootstrap procedure, was the ANSP's MI that was subsequently used to instantiate the SP's MIs.

During the first trial, the instantiation of the first SP's MI took a total of 10,637 ms to carry out all phases, while the rest of the trials gave an average time of 8942 ms. The reason for the difference between times is that in the first trial the PDPs responsible for processing the corresponding policies were not initially present but were deployed on demand upon arrival of the policies; during subsequent trials the PDPs were already in place. Deploying the new PDPs and returning their references took 212 ms and 176 ms for the QoS and delegation PDPs, respectively. Finally, further extending the functionality of these PDPs took an average of 21.3 ms.

In a real-life scenario we do not expect requests for MI creations to occur frequently; nor do we expect a large number of SPs requesting MIs. Furthermore, these MIs are going to operate for long periods commensurate with the lifetime of the SP virtual network. We also consider PDP deployment or extension to be important features of which the SPs would make full use. In this context, these feasibility trials are encouraging, and suggest that continued investigation and enhancement of our management architecture will be worthwhile.

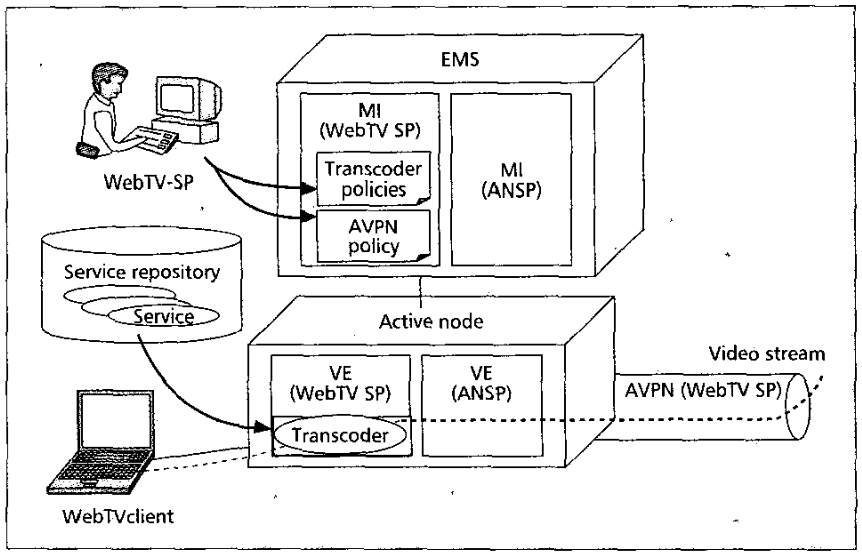


Figure 6. *The WebTV scenario.*

Conclusions

In this article we propose and describe a hierarchically distributed policy-based network management architecture that is an important result of the FAIN project. We have applied policies as a way of managing active networks, and used active technologies and mechanisms to extend the management architecture by dynamically deploying additional PDPs and PEPs.

Although PDPs and PEPs can be deployed on demand, they must comply with the expected (standardized) interface and be registered in the ASP system. Also, our management architecture supports an extension mechanism of a finer granularity by dynamically adding new functionality (policy action and condition interpreters) into already existing PDPs/PEPs.

We have used different types of PDPs and PEPs as a means of differentiating groups of policies and facilitating policy decision making according to a specific context.

Based on a new business model that advocates the deployment of virtual networks on top of the same network infrastructure, we have extended the concept of management by delegation through allowing multiple management architectures to be instantiated and to function independently of each other. This was enabled by the use of the FAIN active node and its open interface.

Finally, we have mostly focused on implementing and experimenting with the configuration model for policy control. We consider the outsourcing model to be equally important. According to this model, control protocols must be policy-aware in order to convey policy information that is necessary for the PDPs to make a decision. In addition, the PDPs need to interact with the PEPs; therefore, additional semantics must be built into the protocol to enable communication with a particular PEP. Building protocols with these properties is also one of the aims of our next stage of research.

Acknowledgments

This article describes work undertaken and in progress in the context of FAIN IST 10561, a three-year project (2000-2003) partially funded by the Commission of the European Union. The authors would like to acknowledge all FAIN partners. Finally, the authors would like to express their thanks to the reviewers for their valuable comments.

References

[1] FAIN Project: http://www.ist-fain.org

[2] K. Chan *et al.*, "COPS Usage of Policy Provisioning," IETF RFC 3084, Mar. 2001.

[3] J. E. van der Merwe *et al.*, "The Tempest — A Practical Framework for Network Programmability," *IEEE Network*, vol. 12, no. 3, May/June 1998, pp. 20–28; http://www.research.att.com/~kobus/ docs/tempest_small.ps.

[4] G. Goldszmidt and Y. Yemini, "Distributed Management by Delegating Mobile Agents," *15th Int'l. Conf. Distrib. Comp. Sys.*, Vancouver, B.C., Canada, June 1995; http://www.cs.columbia. edu/~german/papers/icdcs95.ps.Z

[5] "Specification of Revised Case Study Systems," FAIN Proj. del. 5, http://www.ist-fain.org/deliverables/del5/d5.pdf.

[6] L. Peterson, Ed., "Node OS Interface Specification," AN Node OS Working Group, Nov. 30, 2001; http://www.cs.princeton.edu/nsg/papers /nodeos-02.ps

[7] M. Sloman and E. Lupu "Policy Specification for Programmable Networks" *Proc. IWAN '99*, 1999.

[8] N. Damianou *et al.*, "The Ponder Specification Language" *Wksp. Policies for Distrib. Sys. and Nets.*, HP Labs Bristol, UK, 29–31 Jan. 2001; http://www. doc.ic.ac.uk/~mss/Papers/Ponder-Policy01V5.pdf

[9] Open Source Project OpenORB; http://openorb.sourceforge.net/

[10] Distributed Management Task Force, "DMTF Technologies: CIM Schema V. 2.6," Feb. 2002.

[11] B. Moore *et al.*, "Policy Core Information Model — Version 1 Specification," IETF Policy Working Group, RFC 3060, Feb. 2001.

Biographies

CHRISTOS TSAROUCHIS (christos.tsarouchis@hitachi-eu.com, tsarouchis@ee.upatras.gr) received his B.Sc. in electrical and computer engineering from the University of Patras, Greece, in 2000 and is a Ph.D. student. Since 2000 he is a researcher at the Information Technology Laboratory, Hitachi Europe Ltd., Cambridge, England. His research interests include policy-based network management and programmable networking.

CHIHO KITAHARA (chiho.kitahara@hitachi-eu.com) graduated from Chiba University and entered the System Development Laboratory, Hitachi Ltd., Japan. She was involved in R&D of communication middleware systems, particularly for dynamic QoS control mechanisms in industry. She moved to Hitachi Europe Ltd. in 2000 and has been involved in the IST FAIN project. Currently she is interested in service and QoS management in network systems.

SPYROS DENAZIS (spyros.denazis@hitachi-eu.com) is working as a senior research engineer in the Information Technology Laboratory, Hitachi Europe Ltd., Cambridge. He is currently acting as a work package leader in the European IST project FAIN. His areas of interest encompass programmable networks, router architectures, and dynamic service deployment. He received his Ph.D. in computer science in 1993 from the University of Bradford, United Kingdom, and his B.Sc. in mathematics in 1987 from the University of Ioannina, Greece.

JULIO VIVERO MILLOR (julio@tsc.upc.es) received his M.Sc. in telecommunications engineering from the Polytechnic University of Catalonia in 1999. Since then, he has participated in several IST projects in the area of network management. Currently, he is preparing his Ph.D. His research interests are focused on the area of network management. In particular, he focuses on management of programmable networks and policy-based management.

EPIFANIO SALAMANCA CUADRADO (epi@nmg.upc.es) finished his graduate classes in telecommunications engineering at the Polytechnic University of Catalonia in 1998. In 2000 he joined the Network Management Group, a research group in the Department of Signal Theory and Communications. Since then he has collaborated in several IST projects in the area of network management. His current interests are in management of programmable networks and policy-based network management.

EDGAR MAGAÑA (emagana@nmg.upc.es) and received his M.Sc. in computational engineering from the Polytechnic National Institute of Mexico in December 2001. In February 2002 he joined the Network Management Group, where he has participated in projects on network management, programmable networks, and active grids. He is in the first year of his Ph.D., where he is focusing on management of active grid networks based on policies.

ALEX GALIS (a.galis@ee.ucl.ac.uk) is a visiting professor at University College London. He has co-authored and published more than 100 articles and reports on various networking topics, including network and service management, intelligent services, programmable and active networks, and grid systems; and two books on network management: *Deploying and Managing IP over WDM Networks* (Artech House, 2003) and *Multi-Domain Communication Management* (CRC Press, July 2000).

JUAN LUIS MAÑAS (juan.manas@integrasys.es) received his B.Sc and M.Sc. degrees in electrical engineering (1995 and 2000) from the Polytechnical University of Madrid, Spain. He is currently working as a researcher in the Telecom Division at Integrasys S.A., where his work focuses on monitoring and management architectures for next-generation networks. His research interests include adaptive network management and high-performance network monitoring technologies.

YANNICK CARLINET (yannick.carlinet@francetelecom.com) graduated in 1999 from the Ecole Nationale Supérieure des Mines, Nancy, France. He then contributed to the DARPA program on active networks until 2001 in a public research laboratory (NIST) in the Washington, DC area. Since 2001 he has been an employee of France Telecom R&D, Lannion, France, working on programmable networks and gateways, ad hoc networks, and QoS.

BERTRAND MATHIEU (mathieu2@rd.francetelecom.com) graduated in 1994 from the Institut des Sciences, Toulon, France. Since then he has been an employee of France Telecom R&D, Lannion. His current research interests are related to programmable networks, content adaptation, and QoS.

ODYSSEAS KOUFOPAVLOY [M] (odysseas@ee.upatras.gr) is an associate professor with the Department of Electrical and Computer Engineering at the University of Patras, Greece. His research interests include very large scale integration, low power design, VLSI cryptosystems, and high-performance communications subsystems architecture and implementation. He received his diploma and Ph.D. in electrical engineering from the University of Patras.