A POLICY-BASED MANAGEMENT ARCHITECTURE FOR ACTIVE AND PROGRAMMABLE NETWORKS

Christos Tsarouchis[1,7], Chiho Kitahara[2], Spyros Denazis[1], Julio Vivero[3], Epi Salamanca[3], Edgar Magana[3], Alex Galis[4], Juan Luis Manas[5], Yannick Carlinet[6], Bertrand Mathieu[6], Odysseas Koufopavlou[7]

[1]Hitachi Europe Ltd, [2]Hitachi Japan Ltd., [3]Universitat Politecnica de Catalunya, [4]University College of London
[5]Integrasys, [6]France Telecom, [7]University of Patras

Abstract

There has been a significant shift in the networking and management paradigm resulting in new technologies and network architectures proposed. The motivation behind the shift has been the yet elusive rapid and autonomic service creation, deployment, activation and management combined with context customisation and customer personalisation. However, there is a faint picture that has just started to reveal itself in the sense that it allows us to discern all those characteristics that are necessary to understand new requirements and possible directions. Those characteristics come in the form of concepts of network virtualisation, openness, and intelligence, which have introduced a whole new range of problems and complexity to the way that management is performed in this new network environment. To this end, network management must support co-existence of different management strategies to allow customisation, dynamic extensibility of its functionality, cope with scalability issues, and facilitate service deployment. Meeting these requirements we need to employ and combine new enabling technologies that build on and extend existing architectures. In this paper, we describe the management part of a new network architecture, which has been designed and implemented as part of the FAIN European Union research and development IST project [1]. The FAIN management architecture encapsulates the three aforementioned concepts while it attempts to provide solutions to satisfy these requirements. In particular, we build upon the IETF's Policy-based management framework, which we have used it in an active network environment, and as such it inherits the properties of this enabling technology that can be applied to this new problem space.

1 Introduction

During the last six years there has been a proliferation of new concepts, architectures and technologies that they have contributed to a networking paradigm shift. The motivation behind the shift has been the yet elusive rapid and autonomic service creation, deployment, activation and management combined with context customisation and customer personalization. Such motivation can be traced in different organizations, fora, and research activities as well as market forces. More specifically, IETF's Diffserv [2], which superseded the earlier formed Intserv, attempted to address the problems of communication services creation, deployment and scalability. This was achieved by means of a new service model (Diffserv) and its corresponding mechanisms that implement this model. Such mechanisms come in the form of signalling protocols and new network element functionalities that allowed the reservation of network resources and use thereof according to customer requirements.

More recently, policy-based networking has attracted significant industry interest [3]. Presently, it is promoted by several network equipment vendors in the form of fora like DMTF [4] or is standardised within the IETF Policy working group [5].

Policy-Based Network Management (PBNM) offers a more flexible, customisable management solution allowing each router/switch to be configured on the fly, for a specific application tailored for a consumer. This approach opened a new window of opportunity to operators as it enables them to homogeneously perform their network management tasks, raise the level of interoperability across different vendors' equipment thereby creating a new range of different service products.

However, aiming at rapid service creation and deployment resulted in the development of active technologies that traditionally come from the fields of software and computer engineering like programming languages, object oriented and distributed programming, and operating systems.

Adding to this, the ever-increasing demands for interoperability and pushing intelligence and functionality inside the network, gave rise to a distinct research field known as Programmable or Active Networks mainly represented by the Opensig [6] and

Active Networks communities [7]. Recently, first sightings of programmable networks concepts have also appeared in IETF in the form of the ForCES group requirements and framework [8].

Taking into account all these activities, we can identify a viewpoint that has just started to reveal itself in the sense that it allows us to discern all those characteristics that are necessary to understand new requirements and possible directions. These characteristics come in the form of concepts of a network as a service-enabling platform, which is open, intelligent and offers virtualisation facilities. The management for such a new network environment has to take into account a whole new range of problems and complexity.

To this end, network management must support co-existence of different management strategies to allow customisation, dynamic extensibility of its functionality, cope with scalability issues and facilitate service creation and deployment. To meet these requirements we need to employ and combine new enabling technologies that build on and extend existing architectures.

In this paper, we describe the management part of a new network architecture, which has been designed and implemented within the Future Active IP Networks (FAIN) project [1]. The FAIN management architecture encapsulates the three aforementioned concepts while it attempts to provide solutions to satisfy these requirements. In particular, we build upon the IETF's Policy-based management framework, which we have used it in an active network environment, and as such it inherits the properties of this enabling technology that can be applied to this new problem space.

More specifically, in section 2 we provide a brief state of the art on network and policy-based management adopting the viewpoint of emerging network architectures and the impact thereof on network management. This was actually our starting point in FAIN in order to analyse, design and implement our policy-based network management architecture (PBNM). In section 3, we briefly introduce the FAIN project. The introduction only includes those aspects of the project that are relevant to our paper. In particular, we present the FAIN Business Model that the PBNM architecture realizes while the FAIN Active node reference architecture provides the network context within which our policy-based management architecture has been defined and implemented. In section 4, we give an overview of the PBNM architecture and its objectives. In section 5, we describe all architectural components and the functionality thereof, while in section 6, we provide details of its implementation. We conclude in section 7 including some open issues and future work..

2 Network Management and Emerging Network Architectures

Network management, either of telecommunication or data networks, has long been argued along the manager-agent model [9] and deals with three fundamental aspects: a) *functionality* grouped according to five areas, namely, Fault, Configuration, Accounting, Performance and Security (FCAPS), b) *information modelling* by which network and network element resources are identified and abstracted in a way that underpins specific operational semantics, and c) the *communication method* among managers and agents.

Realising and implementing this model gave rise to a series of different network management architectures each one being the result of limitations of the previous but most notably because of emerging network architectures and the new demands imposed on their management. The latter was the combined effect of rapidly changing customer requirements, enabling technologies, and new market forces. Such management architectures have been categorised into centralised, hierarchical and distributed while a hybrid between hierarchical and distributed is also possible [10], [11].

Although one of the original intentions of making the management architecture hierarchical was the need reducing the management data flows from the agent to the central manager, the management application was still running in the manager and away from the Network Element (NE) while treating the agent as a mere implementation of the communication protocol unable to make any decisions. Management by Delegation (MbD), introduced first as a concept in [12], was conceived in an attempt to transfer the management logic from the central management system closer to the managed entity. This results in alleviating the management burden from the central management system.

We note here that the term "*delegation*" has been heavily used in the literature with different meanings. For instance, this term has been also interpreted in the context of Policy-based management. In [13], it is used to describe the transfer of access rights between subjects by means of delegation policies. In this paper we use both terms while we further extend its meaning. However, we have made an effort so that every time we refer to this term, this is done in a specific context that defines its meaning, thereby avoiding confusion.

Management by Delegation (MbD) addressed the aforementioned problem in the context of an SNMP compliant network although the principles are more generic and applicable to other management paradigms. MbD advocates that for a number of tasks some part of the application logic could be dispatched to the NE, in the form of delegated agents, which then could run in what is called an *elastic server* [14]. This elastic server is an execution environment that provides a number of generic services ranging from operating system services such as thread and communication support, to application specific services such as SNMP support. Using these services the delegated agents can then interact with the NE environment and carry out specific management tasks, thus, reducing the processing load of the manager and the network from the traffic due to management.

MbD is in fact a computational paradigm based on mobile agents that has been applied to decentralise and automate network management tasks. As such (computational paradigm) can be seen as a predecessor of the Active Networks paradigm and the elastic server as the equivalent of an execution environment [7].

Active and Programmable Networks research communities have created a new network paradigm that investigates what is known as the *degree of programmability* [15], [16], which is the trade-off between network flexibility (including extensibility), security, interoperability (including portability) and performance. Three new important concepts have emerged that can be considered as the architectural foundations of the new emerging network & NE architectures [17], [18]:

- Execution Environment
- Open Interfaces
- Network Virtualisation

Each one of these concepts has been thoroughly investigated and researched, producing innovative results and technologies. As such a review and elaboration on relevant research efforts are considered outside the scope of this paper, it suffices to say here that the concept of open interfaces [19] and network virtualisation [20], [21], [22] give rise to new challenges in the network management, while it opens new doors of opportunity by making network management more flexible and intelligent a direct aftermath of inheriting the properties of activeness and programmability. The use and the support for open interfaces is essential, in order to achieve interoperability. Recent efforts in the IEEE P1520 [19], IETF ForCES [23] and MSF forum [24] have been made towards standardized interfaces in order to provide interoperable solutions to the largely diverse environments of today.

In contrast, the same concepts may be viewed as an attempt to meet new customer requirements and the projection thereof to network management. The following requirements are considered [25] as being increasingly consistent across different market segments ranging from Small to Medium Enterprises (SMEs) to large enterprises as well as service providers' Operations Support System (OSS) environments:

- Ease of use and implementation
- Ease of integration
- Dynamic adaptability
- Scalability
- Reliability
- Cost/value

We believe that programmability and activeness can play an important role in the ease of integration and dynamic adaptability whereas network virtualisation in creating more scalable and building cost/value systems. This claim has been one of the objectives of the FAIN project.

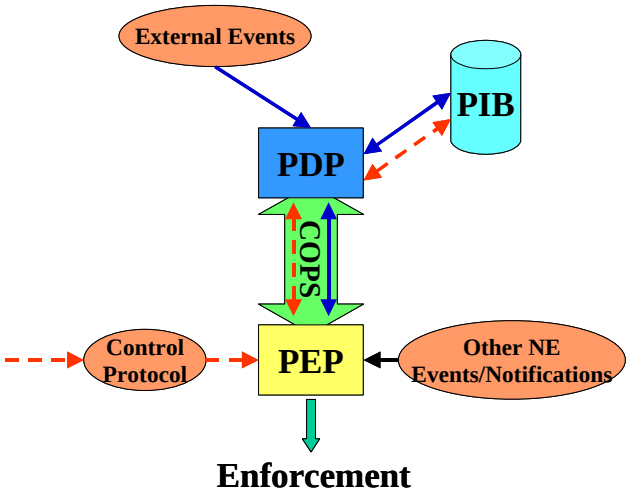

Figure 1. The IETF Policy-based Networking

Traditional network management techniques focused on the procedures that are required to carry out FCAPs functions. They have built the communication protocols and information models that are pervasive across a network so that the network can be managed as smoothly as possible. However, what they have not addressed was a framework for a high-level automated network management based on well-defined rules, which capture the semantics of the procedures that the network should adhere to. Policy-based network management filled this gap.

Policies capture the semantics of a second level and more advanced management logic, which can be associated with different uses of the network resources at the same time. Furthermore, policy-based network management is also instrumental in meeting most of the aforementioned requirements as it provides a common "language" [13] to communicate these requirements to the network infrastructure, which is then configured accordingly.

Figure 1 abstracts the main entities that participate in policy-based networking. Central to this figure are the Policy Enforcement Point (PEP) that resides in the NE and the Policy Decision Point (PDP) that resides in a policy server [26]. As in the case of MIBs in traditional management there exists a database, called Policy Information Database (PIB) that stores the policies (policy rules) that the NEs must adhere to.

The PDPs and PEPs collaborate by exchanging communication messages by means of a standard protocol in order to support policy control. Two common models are used for policy control: Outsourcing and Configuration [27].

Outsourcing was the first model that it was proposed in the context of policy-based networking [26] and it is used for dynamic checks. Consider for instance a control protocol e.g. RSVP, that has been extended to carry policy related information. The PEP formulates a request for policy control and passes that request to the PDP together with the policy related information that is also necessary for the PDP to reach a decision. Then the PDP checks the incoming request by comparing the request against the policy rules stored in the PIB. Finally the PDP returns a decision to the PEP, which either accepts or rejects the original request that came from the control protocol. In Figure 1 the flows that use the outsourcing model are depicted as dotted lines.

In contrast, during the configuration of the network, the request comes from an external source, which could be the result of a Service Level Agreement (SLA) between the network operator and a service provider or by a management application that responds to changes of the network state. This mainly results in updating and storing new PIB values and then the policies are forwarded to the PEPs in order the decisions to be enforced.

In either of the two models, a large number of management operations may be automated and simplified making network management simpler and more scalable. In addition, policy aware NEs manifest a vendor's independence making integration and interoperation feasible without prohibiting product differentiation.

Finally, treating PEPs and PDPs as execution environments that can be extensible, like the elastic server in the MbD paradigm, functionality can be added dynamically thereby adapting the network to new demands in the form of new policies. This was one of the motivations to design our management based on the policy framework.

3 The FAIN Project and its Context

The main objective of the FAIN project is to develop an open, flexible, programmable and dependable (reliable, secure, and manageable) network architecture based on novel active node concepts via the definition of active node and management architectures.

FAIN proposes an active network architecture that defines a set of active nodes, which provide full flexibility to the user to manage and provide active services. The defining characteristic of an active node in this context is the ability for users to load and execute software components dynamically. An active node provides a number of different execution environments, which in turn provide the capability of running user-provided code.

In order to describe best the management architecture, it is required to understand the context within which the management is carried out since it is the synergy of all FAIN architecture elements that satisfies the aforementioned requirements.

More specifically, these are the underlying Business Model [28] adopted by the FAIN consortium and the FAIN Active Node architecture [17]. Therefore, we provide a brief description of them in the following two subsections.

3.1 FAIN business model

In order to make our system able to cope with a large and diverse number of use cases that involve different roles and actors, we have identified the main actors that take part in all the operational phases of the network. These are described in the FAIN Business Model as specified in [28]. The FAIN business model attempts to address some of the requirements of previous section. As a detailed description of this business model is outside the scope of this paper, we only refer to the most important actors and their relationships.

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 secure and isolated access to such facilities, as well as part of its network resources to potential customers like service providers or large corporate customers. Such role may be assumed by network operators.

The Service Provider (SP) buys network resources from the ANSP and creates services comprised of active components delivered by a Service Component Provider. It then deploys these components in the network, and offers the resulting service to Consumers. A Service Provider may federate with other Service Providers in order to build more complex services. Descriptions of the offered services are advertised via a broker service.

The Consumer (C) is the end user of the active services offered by an SP. A Consumer 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 event another SP. In order to find the required service among those offered by the Service Provider, the Consumer may contact a Broker to use its service discovery features.

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

3.2 FAIN AN Node Architecture

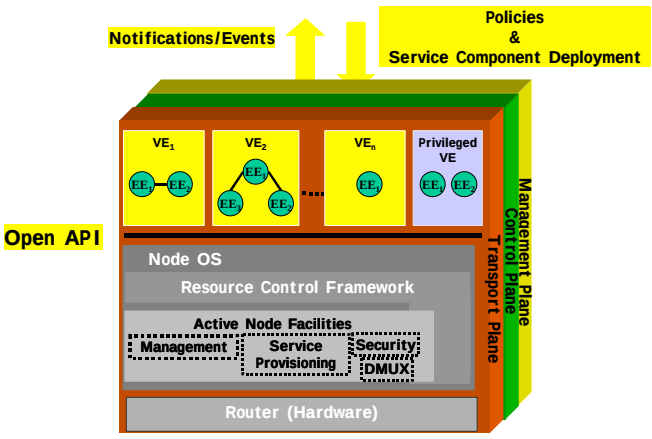

Figure 2. The FAIN Active Node Architecture

Figure 2 illustrates the FAIN active node architecture with its main functional components. As discussed in section 2, the Execution Environment (EE) is one of the main concepts in Active Networks. In FAIN we have distinguished EEs between *types* and *instances* on the analogy of classes and objects in an object oriented framework. EE types define the programming methodology and environment whereas EE instances represent technologies (*e.g.*, Java Virtual Machines). EE instances host the components that make up of a service while services may use more than one types of EEs to deploy their components in the corresponding instances thereof. Finally, EEs may reside in and communicate across all operational planes, namely, transport, control and management. The fact that services are deployed in different EEs constitutes a considerable difference from the original Active Network (AN) architecture model of the DARPA program, according to which each service was only deployed in the same type of EE.

EEs, and consequently services are encapsulated by Virtual Environments (VEs). VEs, that are connected together, provide a proper virtual private network (VPN) on top of the network infrastructure. To this end, the VE is an abstraction that is used only for the purpose of partitioning the resources of the AN Node. VEs enclose AN node resources and access rights for AN users. VEs are built on top of the node operating system (NodeOS). The latter incorporates services of a number of extensions in the form of active node facilities that are required to support the instantiation and operation of different VEs. These facilities are:

Security: protects node and principal resources against unauthorized access. It provides integrity service for network communication, authentication service for active packets and sessions and authorization service for authorizing principal access to NodeOS API. Access is governed by security policies set by an authorized principal.

Resource Access Control: receives requests for allocating computational and communication resources to different users.

Demultiplexing (Dmux): is in charge of forwarding active packets to the corresponding EEs within a VE, based on the packet header information (*e.g.*, ANEP header [29]) and a forwarding table used by this component.

Active Service Provisioning (ASP): is in charge of downloading active services into the active nodes and deploying their service components to the proper EEs or management stations when necessary.

Virtual Environment Manager (VEMgr): includes activities that assist the policy-based management system to enforce its policies, *e.g.*, monitoring of resources, event notification, VE instantiation, *etc*.

Further details on the FAIN AN Node architecture may be found in [17].

4 The FAIN Management Architecture

The FAIN management architecture is a hierarchically distributed architecture, consisting mainly of two levels (two-tiered architecture). The network management level contains the Network Management System (NMS) while the element management level that contains the Element Management System (EMS). These two systems receive and process policies, resulting in decisions and enforcement thereof. The policies received have been grouped according to categories that define the semantics of specific operations, which may range from QoS operations to service specific operations. Accordingly, policies that belong to a specific category are processed by dedicated to this category PDPs and PEPs (Figure 3)

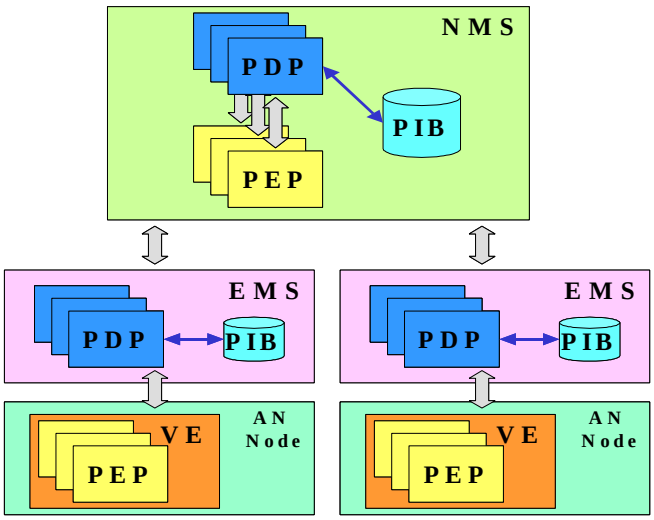

Figure 3. The hierarchical FAIN Management Architecture

The NMS is the entry point of the management architecture. It is the recipient of policies, which may have been the result of the network operator management decision due to a change in the network state or a service level agreement (SLA) between ANSP & SP, or SP & C. This SLA requires configuration of the network, which is automated by means of policies sent to the NMS.

The network configuration policies are checked by the NMS PDPs if they can be enforced and then are delivered to the NMS PEPs that perform a mapping between network level and element level policies, which are then sent to the EMS PDPs. The EMS PDPs perform EMS related checks and finally are given to the AN node PEPs for enforcement (Figure 3).

This is what we have previously referred to as the policy control configuration model and its use in hierarchically distributed management architecture combines the benefits of both approaches, namely, automation with reduction of management traffic and distribution of tasks.

The FAIN management architecture is based on the FAIN business model. In particular, the relationship among the three main actors, namely, ANSP, SP, and C, is projected directly into the FAIN management architecture. Accordingly, each one of these actors may request and get his own (virtual) management architecture through which he is enabled to manage the (virtual) resources of the virtual network.

In this way, each actor is free to select and deploy his own model, namely, his own management architecture, of managing the resources, which can be centralized, hierarchical, policy or non-policy based. The type of services and the complexity of the virtual network dictate the particular choice of the management architecture. To this end, we create an environment that is capable of accommodating opposing requirements that a traditional approach of one-size-fits-all does not scale to accomplish.

Our model extends the Tempest approach [20] in the management plane, which was the first to advocate the simultaneous support of control architectures for ATM networks.

It also extends the meaning of the term MbD as it allows to delegate the entire responsibility of network management to a third party, e.g. an SP, which can be deployed and hosted in a separate physical location than the NMS of the owner of the network, e.g. the ANSP.

Figure 4 illustrates the above claims. Starting with the management architecture of the network operator, namely, the ANSP, it instantiates and registers a new management instance (MI), which is delegated to one of his customers, i.e. the SP. This management instance will host the SP's management architecture. The SP has the choice to buy from the ANSP an instance of the ANSP's architecture, in our case a policy-based. To this end, the network management architecture developed by the ANSP not only is used for managing the NEs but it becomes a commodity, which may become another source of income for to the ANSP. In contrast, the SP does not need to build his management architecture from scratch but it can use an existing one as a basis to customise it according to the services that will run. In FAIN we have experimented with this instantiation of management architectures using the PBNM system of the ANSP to instantiate another one for the SP, which was used by the latter to deploy and deliver a service e.g WebTV, to his customers. Note also that this instantiation relationship can be recursive in the sense that the SP may further delegate his own instances to a Consumer, for example to differentiate and manage different groups of

customers.

One important assumption underlying the above 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 as a strong condition but there is convincing evidence that there exists a strong push towards ubiquitous open interfaces. Initiatives like the IEEE P1520 and lately the ForCES IETF working group serve as a proof for such claims. Furthermore, the programmable and active networks paradigm also relies on similar assumptions [22].

5 Management Component Description

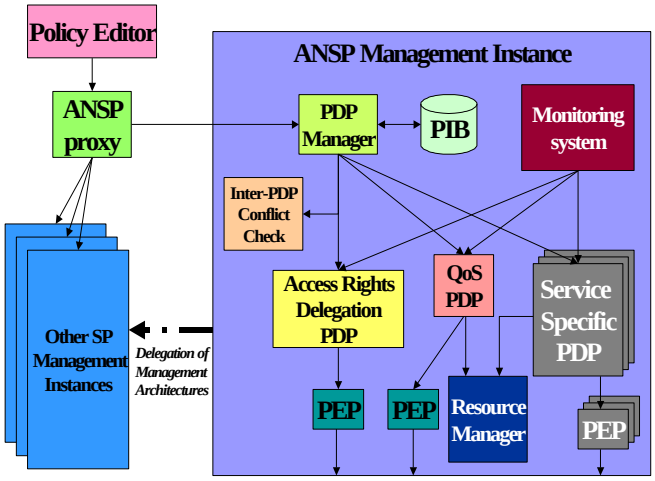

Figure 4. FAIN Management Instances and their components

We proceed to present the details the FAIN policy-based management architecture. Following the FAIN business model, the first instance of a management architecture is that of the ANSP's during the bootstrap of the whole network. The NMS and EMS of the ANSP instance have similar functionality and components therefore we focus on the NMS and wherever applicable we will note the differences between the two.

5.1.1 Policy editor

The policy editor exists only at the network level. It does the mapping between the SLA and the network level policies while it offers to the ANSP administrator a GUI and a toolset in the form of templates and wizards for the composition of policies.

5.1.2 ANSP Proxy

Policies coming from the policy editor are forwarded to the ANSP proxy. The ANSP proxy forwards the policies to the appropriate MI (either ANSP or SP MI). The ANSP proxy is introduced to provide an additional level of security to the ANSP and/or his customers, the SPs. It provides authentication and authorisation of the incoming requests (policies) and finds the right MIs to which the policies must be forwarded. The actual policy evaluations are processed by the proper instances and not by the ANSP-proxy. In other words the ANSP proxy makes the management architecture more robust from the security point of view.

5.1.3 Inter-PDP Conflict check

Conflicts may appear when a policy arriving to one PDP clashes with a contradicting policy processed by another PDP. Conflicts may be distinguished between [30], *syntactic* conflicts that can be resolved with policy syntax analysis and the use of policy priorities, and *semantic* conflicts, which 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 FAIN project thus is not included in this paper..

5.1.4 PDP manager

The PDP Manager receives policies and dispatches them to the appropriate PDPs. If the corresponding PDP is not installed, it requests its download and installation from the code downloading framework also developed within the FAIN project. This framework is called Active Service Provisioning (ASP) [31]. In this way, the management functionality of the system can be extended when needed.

The PDP Manager also acts as a point of control, as it has all the necessary information to understand the stage of the policies deployment procedure. For example, two different policies need to be deployed but the latter should only be enforced if the former is enforced successfully. In this case, the PDP Manager will keep the second into a halt state, until it receives the notification from a PDP of the correct enforcement of the first.

This coordination takes place when an SP requests the instantiation of a new virtual network and a new MI. The PDP manager will first forward the QoS policies for enforcement, which requires admission control, and if the enforcement is successful will release the access rights delegation policies. When this later enforcement is also in place, it will then instantiate the new MI and return it to the SP so that the SP can start using and managing his own resources according to the SP's policies.

5.1.5 Different types of PDPs

In general, there exist different types of PDPs, each one of them making decisions which apply to a specific context namely, QoS PDP, Delegation of Access Rights PDP as well as Service-specific PDPs.

They all perform conflict checks that are meaningful within (intra-PDP) their decision context. In order to reach a decision, they also interact with other components, which assist the PDPs in making a decision, e.g. a Resource Manager for admission control. The combination of inter-PDP and intra-PDP components enables us to build a hierarchical mechanism for processing complex policies. In what follows we look closer to the specific types of PDPs used in the FAINmanagement architecture.

5.1.5.1 QoS PDP

The QoS PDP is responsible for the analysis of the QoS policies. Specifically, it decides when a policy should be enforced or not; forwards decisions to PEP components in order to be enforced; accepts requests that come from PEPs; and controls the policy-validity period in order to uninstall expired policies.

In order to realise these functionalities the QoS PDP needs to receive information from the monitoring system and resource manager components and make proper decisions based on policy conditions and the states of the network and network nodes.

Policies processed by this PDP are oriented to the differentiation of certain flows, or groups of flows, with an enhanced quality of service.

5.1.5.2 Access Rights Delegation PDP

By Access Rights Delegation policies [13] we refer to those policies that specify to what extent a principal is allowed to access network resources by means of accessing the control interfaces thereof. To this end, controlling the access to resources, you also control the operations on them, eventually restricting the capabilities of services that are deployed by this principal.

In FAIN the Delegation PDP is used by ANSP to determine what operations an SP's services are allowed to carry out on the network resources assigned to him as part of his virtual network creation.

The delegation of access rights involves the configuration of the nodes security components that are part of the topology of his virtual network. Every request for the use of a particular interface is checked by the security component. Access is granted only to the authorised entities.

For example, the ANSP may want to release specific parts of the node resource control interface to the SP. Eventually, the element-level Delegation PDP will configure the node-level security components that exist in every active node that the SP wants to manage. The security components, based on the result of the authentication service and security policies will grant or deny access to the node resource control interface.

5.1.5.3 Other Service Specific PDPs

The FAIN management architecture is designed to accommodate new PDPs that participate in decisions which are service-specific. This service-specific PDPs may be deployed on demand using the same deployment mechanisms built in FAIN.

5.1.6 Policy Enforcement Points (PEPs)

Each type of a PDP has its own PEP counterpart. Network level policies are translated by the network level PEP into element level ones, and then are sent to the right 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 as to where policies should be distributed.

Similarly, element level PEPs enforce the element level policies, sent by the EMS PDPs, by mapping them onto the proper 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.

5.1.7 Monitoring system

The monitoring system is logically distributed in the overall management framework. It covers the network level and the element level. The monitoring logic at the node level is basically offered by the FAIN active node through its interface. The EMS contains the monitoring logic at the element level, complementing the node level, while it offers information analysis, alarm correlation, filtering, setting of thresholds, event reports, etc.

This component is structured in layers. The lower layer is the data acquisition layer, which is in charge of gathering the value of the parameters under supervision and processing such values to obtain the information of interest. A distribution layer is placed upon it. By decoupling event producers ,who create events, and consumers, who receive and analyze events, it increases the system flexibility: any value is available to a wide number of consumers without the need of having defined them previously. Finally, the control layer aims to make decisions affecting the way that the monitoring operations are carried out.

In this architecture the interfaces for accessing the monitoring system are located at two different levels: on the first one there is the possibility of controlling how the monitoring is performed by distributing appropriate policies. On the second one access to the distribution layer is provided, allowing the retrieval of the data gathered by the acquisition layer.

The monitoring system at the network level and at the element level has the same structure with the difference of the type of information processing. The monitoring system at the network level is connected to the notification channel at the element level, as any other consumer. The definition of appropriate filters avoids event flooding processes and provide the necessary degree of scalability.

6 Management System Implementation

6.1 Overview of the implementation based on a scenario

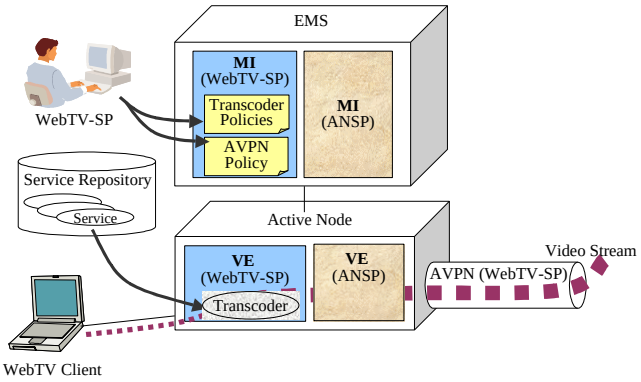

Figure 5. The WebTV Scenario

In the service scenario we have chosen to implement and experiment with an SP who wants to offer its customers (end-users) a WebTV service (Figure 5). We call this SP, WebTV-SP and broadcasts a video program in the Internet that end-users are able to watch it irrespective of their terminal capabilities. The WebTV-SP requests from the ANSP to set up an Active Virtual Private Network (AVPN) wherein he can deploy services that may be customised to meet customer requirements. Customers then subscribe to this WebTV service by directly contacting the WebTV-SP server.

In this context, one of his customers uses a terminal that is not capable of displaying correctly the video stream of the WebTV content. For instance, this particular customer may use a handheld device with low processing power and a low access bandwidth. In this case, the WebTV-SP can individually select and process the video stream destined to his customer by deploying an audio/video transcoder in the network so that the video stream received by the handheld device is of the same format.

We will now describe the scenario interactions in detail, considering the interactions among the different FAIN network systems, namely the policy-based management system and the Active Service Provisioning (ASP) system.

As a result of an SLA agreed between the ANSP and the SP, policies are sent to the ANSP MI. These policies are edited using the GUI of the Policy Editor as depicted in Figure 6.

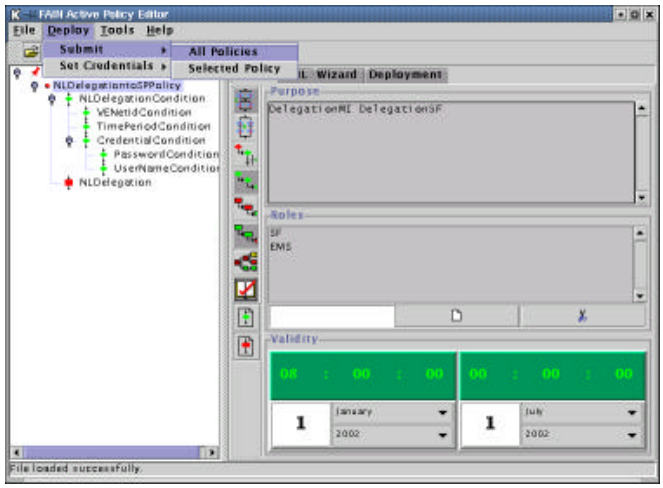

Figure 6. The FAIN GUI of the Policy Editor

Consequently, the ANSP PBNM receives a QoS policy and enforces it on both the NMS and the EMS. This results in invoking the active node management framework to create a new Virtual Environment (VE) for the WebTV-SP. If the VE creation is done successfully, then the ANSP PBNM enforces a delegation policy through the NMS and the EMS. This enforcement consequently requests the active node management system to activate the newly created VE. The ANSP then creates a Management Instance (MI) in all the appropriate EMS stations for this WebTV-SP and assigns the access rights to the active nodes interfaces.

The WebTV-SP is now ready to configure his AVPN by sending policies that are customer specific. The SP also installs service components into the active nodes (a transcoder component in our scenario). The service components are deployed by the ASP system based on the service descriptors.

In addition, the SP deploys service-specific policies in the QoS PDP of his MI after the deployment of the transcoder service component in the active node. In this way the SP can define its own service-specific policies that will be enforced in the active node.

Finally, the monitoring system is used for the reconfiguration of the transcoder at runtime, when for instance the access bandwidth changes dramatically and the end-user needs a different transcoding format

on the video stream.

6.2 Implementation in the FAIN test bed

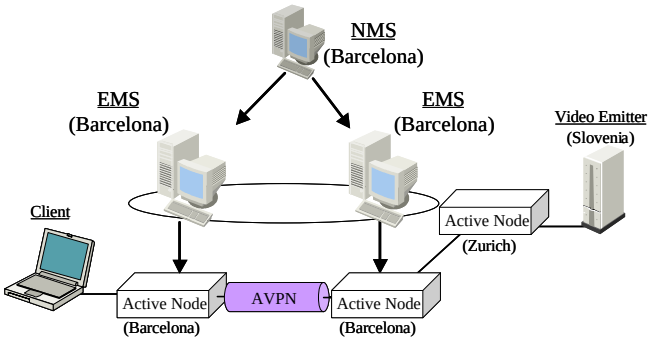

Figure 7. Deployment of the FAIN management system in the FAIN testbed

The prototype implementation was developed based on the FAIN testbed which used Linux PCs that reside across Europe, namely, Greece, UK, Spain, Germany, Switzerland, USA and Slovenia. The part of the FAIN testbed that was used to deploy the scenario is shown in Figure 7. Each site used a JVM (Java Virtual Machine) and was connected to the Internet. Initial trials have focused mainly on our management system functional evaluation. Performance evaluations will be carried out at the final stage of the project.

The PBNM software package includes all the classes, the interfaces and the policy data definitions. The interfaces are defined in IDL (Interface Definition Language) and use the CORBA technology for inter-component communications [32].

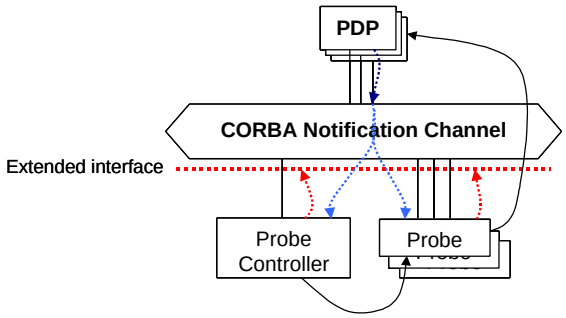

Figure 8. The FAIN Monitoring System

We have implemented those IDL interfaces in Java since Java possesses a number of very useful API's as well as because of its portability in various hardware platforms. The capability of dynamically instantiating new classes "on the fly" is also helpful in supporting extensibility.

We have used CORBA as our middleware technology and in particular the open source package OpenORB [33]. We extended the CORBA notification service to be used in our monitoring system. The PDP Manager, ANSP proxy, PDPs and PEPs are instantiated as CORBA objects, then registered in the CORBA naming service. With this approach we tried to show the seamless nature of performing management even if the EMSs and the NMS are physically located in different places.

The distribution layer of the monitoring system is supported on top of the CORBA notification channel offering the event subscription methods for PDPs. An additional façade simplifies the way that the PDPs attach and detach to/from the notification channel by registering events with the subscription filters to be monitored. Nevertheless, unlike the traditional notification channel, the distribution layer makes the transfer of additional information possible, allowing the automatic configuration of the acquisition layer. In particular the information contained in the subscription filters defined by the PDPs is used for this purpose.

Components, called probes, are in charge for accessing the active nodes and for obtaining monitoring information from them based on received filters. In addition, probes are controlled by a probe controller that allows for their configuration (Figure 8). When a probe announces its capacity generating a certain type of event from active nodes, it is automatically subscribed to receive the filters that are initially requested by PDPs and defined for such an event type. Such information is subsequently used by probe controllers to define thresholds that meet requests of the PDPs.

The event model was defined based on CIM (Common Information Model) [34] and this helps to unify the information models used in the upper layers of the management architecture since PCIM (Policy Core Information Model), which was used in the management architecture [35] is also based on CIM. In our implementation, the QoS PDP has subscribed to events that were triggered by excess of resource usages in active nodes whereas the delegation PDP subscribed to events that were triggered by malicious accesses against active nodes.

```
<fainPolicyRule>
 <PolicyAction>
   <fainRCFAccessRight>readwrite</fainRCFAccessRight>
   <fainDemuxAccessRight>readonly</fainDemuxAccessRight>
   <fainSecurityAccessRight>disable</fainSecurityAccessRight>
   <ActionMode>0</ActionMode>
 </PolicyAction>
</fainPolicyRule>
```

Figure 9. Delegation Policy example

As said earlier, we have followed the PCIM of IETF to define our policy data and have also made some extensions to it. For instance, the *fainPolicyRule*, *fainSimplePolicyAction* and *fainPolicyCondition*

classes are inheriting *PolicyRule*, *SimplePolicyAction* and *PolicyCondition* classes of PCIM respectively. The *fainSimplePolicyAction* allows us to include new properties such as EnforcementStrategy and ActionMode. The fainPolicyRule added new properties that supported extended QoS functions and newly introduced delegation functions in the FAIN framework. We have selected to use XML for the representation of policies and schemas as it is a de facto standard and users are allowed to exchange text-based documents [36]. In addition, with XML we have a variety of choices of commonly available Internet protocols for transmitting policy data and as well as a plethora of XML parsers. For our prototype implementation, we have prepared two kinds of privileged policies, specifically a QoS policy and a Delegation policy that are in charge of controlling VEs. The example shown in Figure 9 contains a part of the delegation policy that is responsible for configuring the access rights in the active nodes.

7 Conclusion

In this paper, we have proposed and described as part of the FAIN project a hierarchically distributed policy-based network management architecture. We have applied policies to the management of active networks and active technologies and mechanisms to extend the management architecture by dynamically deploying additional PDPs and PEPs.

We have used different types of PDPs and PEPs as a means to differentiate among different groups of policies and facilitate 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 by allowing multiple management architectures to be instantiated and operate independently of each other. This was enabled by using the FAIN active node and its open interface.

Although PDPs and PEPs can be deployed on demand, they must be pre-defined and already part of the overall management logic. In other words cannot be extended beyond its original design. What we have managed to do in FAIN is to deploy parts of the architecture when it is needed. However, a more dynamic type of extensibility may be possible by introducing to the management architecture entirely new PDPs/PEPs. However, this requires extending our management framework to cope with this requirement and this is a future objective.

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

ACKNOWLEDGMENTS

This paper describes work undertaken and in progress in the context of the FAIN – IST 10561, a 3 years project during 2000-2003, 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.

8 References

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

[2] S. Blake et al., "An Architecture for Differentiated Services", IETF RCF 2475, 1998

[3] IP Highway, Inc "Policy standards and IETF terminology", http://www.stardust.com/policy/index.htm

[4] Distributed Management Task Force Inc., http://www.dmtf.org/

[5] IETF Policy Framework group, http://www.ietf.org/html.charters/policy-charter.html

[6] http://comet.ctr.columbia.edu/opensig/activities/activities.html

[7] "Architectural Framework for Active Networks", Draft version 1.0, K.L. Calvert, ed., July 27, 1999. http://protocols.netlab.uky.edu/~calvert/arch-latest.ps

[8] IETF ForCES, http://www.ietf.org/html.charters/forces-charter.html

[9] S. Aidarous and T. Pevyak (eds), "Telecommunications Network Management: Technologies and Implementations", IEEE Press, 1997.

[10] J.P. Martin-Flatin, S. Znaty. "A Simple Typology of Distributed Network Management Paradigms", Proc. Of 8th IFIP/IEEE International Workshop on Distributed Systems, Operations & Management (DSOM '97), Sydney, October 1997.

[11] M. Kahani, and H.W.P. Beadle, “Decentralised Approaches for Network Management”, Comp. Comm. Review, Vol. 27, No. 3, July 1997.

[12] G. Goldszmidt and Y. Yemini. “Distributed Management by Delegating Mobile Agents”. In The 15th International Conference on Distributed Computing Systems, Vancouver, British Columbia, June 1995 http://www.cs.columbia.edu/~german/papers/icdcs95.ps.Z

[13] N. Damianou, N. Dulay, E. Lupu, M Sloman, “The Ponder Specification Language” Workshop on Policies for Distributed Systems and Networks (Policy2001), HP Labs Bristol, 29-31 Jan 2001 - http://www.doc.ic.ac.uk/~mss/Papers/Ponder-Policy01V5.pdf

[14] G. Goldszmidt and Y. Yemini. “Delegated Agents for Network Management”, Special Issue on Management of Heterogeneous Networks, IEEE Communications Magazine, Vol. 36, No. 3, March 1998.

[15] J. M. Smith , K. Calvert, S. Murphy, H.K. Orman, and L.L. Peterson, “Activating Networks: A Progress Report”, IEEE Computer 32(4):32–41, April 1999. http://www.cs.princeton.edu/nsg/papers/an.ps

[16] A. T. Campbell, H. De Meer, M. E. Kounavis, K. Miki, J. Vicente, and D. Villela “A Survey of Programmable Networks”, ACM Computer Communications Review, April 1999, http://www.acm.org/sigcomm/ccr/archive/1999/apr99/ccr-9904-campbell.pdf

[17] “Revised Active Network and Active Node Architecture”, FAIN Project Deliverable 4, http://www.ist-fain.org/deliverables/del4/d4.pdf.

[18] S. Denazis, T. Suzuki, and S. Karnouskos, “Component-based Execution Environments of Network Elements and a Protocol for their Configuration”, Special Issue on “Technologies that promote computational intelligence, openness and programmability in networks and Internet services”, W. Pedryz et. al. (eds) IEEE Transactions on on Systems, Man and Cybernetics, 2002 (submitted for publication)

[19] Biswas, J., et al., “The IEEE P1520 Standards Initiative for Programmable Network Interfaces”, IEEE Communications, Special Issue on Programmable Networks, Vol. 36, No 10, October, 1998, http://www.ieee-pin.org/

[20] J.E. van der Merwe, S. Rooney, I.M. Leslie and S.A. Crosby, “The Tempest - A Practical Framework for Network Programmability”, IEEE Network, Vol 12, Number 3, May/June 1998, pp.20-28. http://www.research.att.com/~kobus/docs/tempest_small.ps

[21] M. E. Kounavis, A. T. Campbell, S. Chou, F. Modoux, J. Vicente, and H. Zhang, “The Genesis Kernel: A Programming System for Spawning Network Architectures”, IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Active and Programmable Networks, Vol. 19, No. 3, pp. 49-73, March, 2001. http://comet.ctr.columbia.edu/genesis/papers/jsac2001.pdf

[22] “Node OS Interface Specification”, AN Node OS Working Group, Larry Peterson, ed., November 30, 2001. http://www.cs.princeton.edu/nsg/papers/nodeos-02.ps

[23] IETF ForCES, http://www.ietf.org/html.charters/forces-charter.html

[24] Multiservice Switching Forum (MSF), http://www.msforum.org/

[25] D. Drogseth, “The New Management Landscape”, PACKET, CISCO System Users Maqazine, Third Quarter, 2002

[26] R.Yavatkar, D.Pendarakis, R.Guerin “A framework for Policy-based Admission Control” RFC 2753, January 2000.

[27] K. Chan, et. al., “COPS Usage of Policy Provisioning”, IETF RCF 3084, March 2001.

[28] “FAIN Business Model”, FAIN Project Deliverable 1, May 2001, http://www.ist-fain.org/deliverables/del1/d1.pdf

[29] The Active Network Encapsulation Protocol (ANEP). http://www.cis.upenn.edu/~switchware/ANEP/

[30] M. Sloman, E. Lupu “Policy Specification for Programmable Networks” in Proceedings of IWAN99, 1999

[31] “Specification of Revised Case Study Systems”, FAIN Project Deliverable 5, http://www.ist-fain.org/deliverables/del5/d5.pdf.

[32] Object Management Group Inc., editor. “The Common Object Request Broker: Architecture and Specification (Version 2.5)”, September 2001, http://www.omg.org,/

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

[34] Distributed Management Task Force, Inc., "DMTF Technologies: CIM Schema V. 2.6", February 2002

[35] B. Moore, , E. Ellesson, J. Strassner, and A. Westerinen, “Policy Core Information Model -- Version 1 Specification”, IETF Policy Working Group, RCF3060, February 2001.

[36] W3C working group, “Extensible markup language (XML) 1.0”, W3C Recommendation 6 October 2000.