Chapter 3

III. STATE OF THE ART IN MANAGEMENT OF ACTIVE NETWORKS

Section III.1 – Introduction

Up to this point we have presented the problem space where the thesis is located as well as the requirements set to the problem solution. Before proceeding with a solution proposal, we will first look at different approaches suggested to the particular problem targeted by the thesis or similar ones.

More specifically, this chapter reviews different research projects developed in the area of management of active and programmable networks that try to encompass the whole range of solutions proposed. We have divided the analysis in non policy-based management proposals first and policy-based management ones afterwards. Nonetheless, we focus in the policy-based management ones, as they are closer to our proposed solution. At the end of the chapter, we elaborate around the direction that current research and standardisation activities seem to take for the short future.

The goal is to have a clear overview of the different alternatives proposed to manage programmable networks and at the same time focus on those that have chosen a similar approach to ours, that is, policy-based management.

Section III.2 - Non policy-based management of active and programmable networks

There are a number of research projects that cover the field of active and programmable networks management using non policy-based approaches. Most of these projects are indeed using active or programmable network techniques to achieve a more efficient management. Hereafter, we are going to briefly review some of these research projects. The projects presented have been chosen to exemplify the wide range of solutions proposed.

1st ABONE Management

The ABone [ABone] is a DARPA funded virtual testbed for the active networks research program. It is composed of a set of computer systems configured into virtual active networks.

The ABone nodes are administered locally, but can be used by remote users to start up EEs and launch AAs. Each core ABone node is configured with seven Unix specific accounts. Each account runs an instance of Anetd - active network management daemon [Berson00]. These daemons allow remote EE and AA developers to install, configure and control EE instances in these nodes. The "Anetd Client - sc", can be used to communicate with Anetd on these machines and perform the required functions. Anetd performs two major functions: deployment, configuration and control of network software, in particular EE prototypes; and demultiplexing of active network packets encapsulated using the Active Network Encapsulation Protocol (ANEP) [ANEP] to multiple EEs located on the same network node.

The ABone has reached a lot of popularity for being the first, and only, large testbed for active networks. However, from the management point of view its functionality is very limited. It considers just the management of network elements individually, and furthermore, it manages just ABone nodes. That is, linux or unix machines where the anetd is running. This is explained by the fact that it was conceived just as a maintenance utility for the ABone testbed.

2nd ABLE

The Active Bell Labs Engine (ABLE) [ABLE], [Kornblum00] proposes a novel active network architecture that primarily addresses the management challenges of modern complex networks. Its main component is an active engine that is attached to any IP router to form an active node.

The active engine is designed and implemented to execute programs that arrive from the network. Both engine facilities and executed programs are oriented to the monitoring and control of the attached router. The active code is implemented in JAVA and active packets are encapsulated in a standard ANEP header over UDP.

The authors claim that ABLE offers an efficient access to the local state of the router, a secure system to modify the router behaviour as well as easy to use programming abstractions and interfaces.

ABLE is not a management system itself but a facility for a management system. ABLE “activates” passive routers. That is, it allows a management system to manage a passive router with active packets. These packets are captured by the ABLE engine, which executes them causing the corresponding management actions on the passive router. As ABLE is attached to a passive router it is a network element management facility. The goal of the thesis (management of heterogeneous active, programmable and passive networks) is different, and wider, from the one in ABLE.

3rd AVNMP

The Active Virtual Network Management Protocol (AVNMP) [Bush01] [Galtier01] prediction algorithm is a proactive management mechanism; in other words, it provides the ability to solve a potential problem before it impacts the system by modelling network devices within the network itself and running that model ahead of real time. Predictions range from network performance to possible network or node faults.

Such a proactive management approach is particularly useful in many applications as handovers in a mobile environment, if the handover is prepared in advanced the service quality degradation is minimised, or QoS-sensible applications, particularly those that are affected by an excessive or variable delay since the management system can avoid congestion before it actually happens.

The system is composed by different types of active nodes with different targets. Some active nodes realise predictions based on the information they have got and publish them on the network. These predictions can be either about the network or about an offered service. Then a second type of active nodes, capture these predictions and introduce them on the management algorithms implemented. The algorithms mainly compare the actual state of the network with predictions received in the past. If the prediction was incorrect, the configuration actions caused by this prediction are removed from the network. This correction is done through a special kind of messages called: anti-messages. A schematised explanation of this algorithm is shown in the figure below.

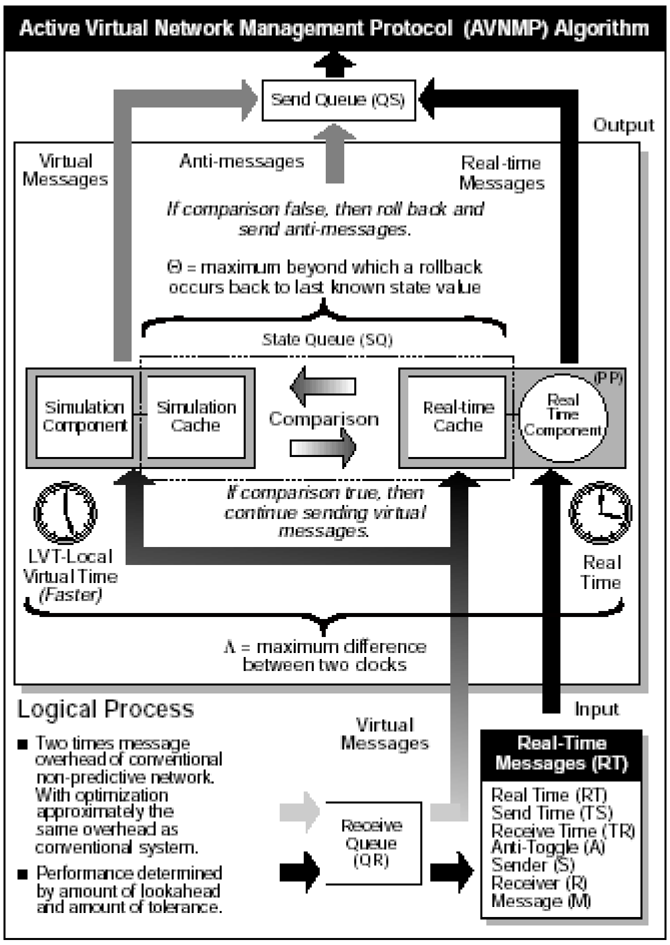
Active Virtual Network Management Protocol (AVNMP) Algorithm
Send Queue (QS)
Output
Virtual Messages
Anti-messages
Real-time Messages
If comparison false, then roll back and send anti-messages.
Θ = maximum beyond which a rollback occurs back to last known state value
State Queue (SQ)
Simulation Component
Simulation Cache
Comparison
Real-time Cache
(PP)
Real Time Component
If comparison true, then continue sending virtual messages.
LVT-Local Virtual Time (Faster)
Real Time
Λ = maximum difference between two clocks
Logical Process
Two times message overhead of conventional non-predictive network. With optimization approximately the same overhead as conventional system.
Performance determined by amount of lookahead and amount of tolerance.
Virtual Messages
Input
Receive Queue (QR)
Real-Time Messages (RT)
Real Time (RT)
Send Time (TS)
Receive Time (TR)
Anti-Toggle (A)
Sender (S)
Receiver (R)
Message (M)

Figure 3 - 1. AVNMP Prediction algorithm

The AVNMP project suggests a prediction algorithm for enhancing system's management by avoiding potential future problems or pre-configuring the network for future events. The algorithm needs the presence of several active nodes realising precise roles inside the managed network. The goal of this project is different from the thesis goal. Furthermore, the need of having concrete active nodes over the network implementing the algorithm functionality restricts the network heterogeneity characteristic. Nevertheless, as one of the MANBoP requirements is the capacity of managing heterogeneous devices and services, the MANBoP system can be also prepared to manage the active nodes where AVNMP algorithm is implemented, the AVNMP algorithm itself and even handle its results to decide on the enforcement of related policies.

4th Smart Packets

Smart Packets [Schwartz99] focus on applying active networks technology to network management and monitoring without placing undue burden on the nodes in the network. The management applications developed are oriented to diagnostic reporting and fault detection.

The framework is based on active packets carrying programs that are executed at nodes on the path towards one or more target hosts. Smart Packets programs are written in a tightly-encoded safe language (spanner) specifically designed to support network management and avoid dangerous constructs and accesses. The spanner code is obtained after compiling the program written in a high-level programming language specifically created for the project: sprocket.

Smart packets are generated by management or monitoring applications and are encapsulated in ANEP (Active Network Encapsulation Protocol) [ANEP]. The ANEP daemon is responsible of receiving and forwarding smart packets correctly. The figure below shows the Smart Packet encapsulation.

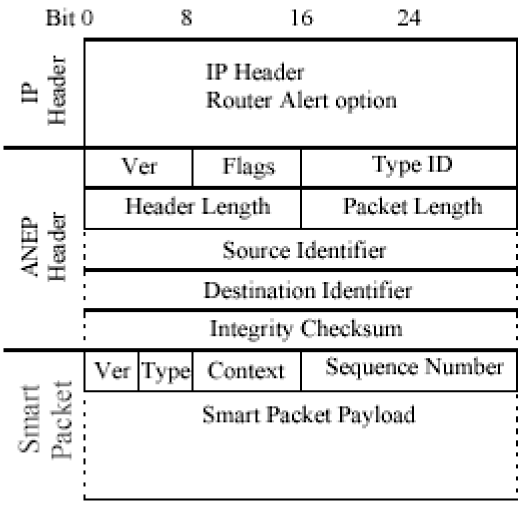

Figure 3 - 2. Smart packet encapsulation on IP and ANEP

Security is achieved through the limitations imposed to the tightly-encoded safe language and through a prudent execution of smart packets code: if the virtual machine does not know how to proceed with the code it stops the execution. Additionally, further security checks are realised such as user authentication and data integrity checks.

Smart packets need the anetd and a particular EE running within the active node to work. Therefore, the smart packets solution is not suitable for heterogeneous networks. Furthermore, the Smart Packets project is focused on diagnostic reporting and fault detection on an active network, it does not cover any of the other management functional areas (FCAPS) [ITU00]. Summarising, the smart packets project gives a solution to only a small part of the problem covered by this thesis. However, as for the AVNMP algorithm, the MANBoP framework will capable be of, if needed, managing active nodes where smart packets are supported, controlling the smart packets mechanism itself and using its outputs to take appropriate decisions.

5th SENCOMM

The main objective of the Smart Environment for Network Control, Monitoring and Management (SENCOMM) [SENCOMM] [Jackson00] framework is to implement a network control, management and monitoring environment using active networks. SENCOMM is somehow a continuation of Smart Packets since it reuses much of the Smart Packets system.

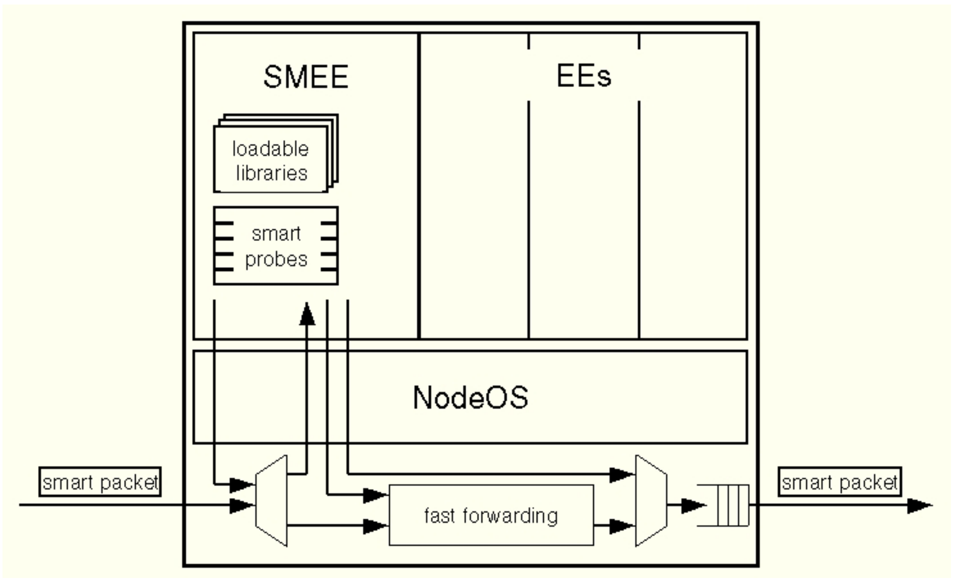

Figure 3 - 3. SENCOMM architecture

User-written network management and monitoring programs generate smart probes, which are encapsulated in ANEP frames. The probes are demultiplexed to the local SENCOMM Management EE, which injects the smart probes into the network. A probe can be sent to be executed only at the destination or at every active node running the SENCOMM Management EE (measurements and control operations might be taken in a single packet's traversal of the network). The probe contains directives to access loadable libraries of functions on the node, registers to receive incoming packets that meet a filter specification, and optionally inject the packet back into the network. Probe packets can be sent either to unicast or multicast addresses. The information content returned by probes to the management center can be tailored in real-time to the current interest of the center.

The SENCOMM project extends the Smart Packets mechanism by including it within a broader management system. Such a system is structured around a management center that takes all decisions. Hence, the management infrastructure in SENCOMM is fixed and cannot be altered. Unlike the Smart

Packets project, SENCOMM is capable of covering all FCAPS functionality although it is still limited to the management of active nodes where the SENCOMM Management EE (SMEE) is running; hence, it does not support heterogeneous networks. Moreover, SENCOMM does not support either another of the MANBoP requirements, which is the capacity of delegating management functionality.

6th VAN

The Virtual Active Network (VAN) management framework [Brunner01] allows customers, on the one hand, to access and manage a service in a provider's domain, and, on the other hand, to outsource a service and its management to a service provider.

VAN supports generic, i.e. service-independent, interfaces for service provisioning and management, and customised service abstractions and control functions, according to customer's requirements.

Only two types of EE exist in the management architecture: the management EE that works on the management plane, and the service provider EE that works on the data transfer as well as on the control plane.

The tasks of the management EE are limited to node configuration and the management of virtual active networks in the active network provider's domain. Note that in this context VAN management means the creation, modification, monitoring, and termination of virtual active networks. The management EE is not concerned with the management of active services running in the virtual active networks.

In the VAN architecture, a service and the corresponding service management run in the same instantiation of a service provider EE. The figure below shows the VAN Management architecture.

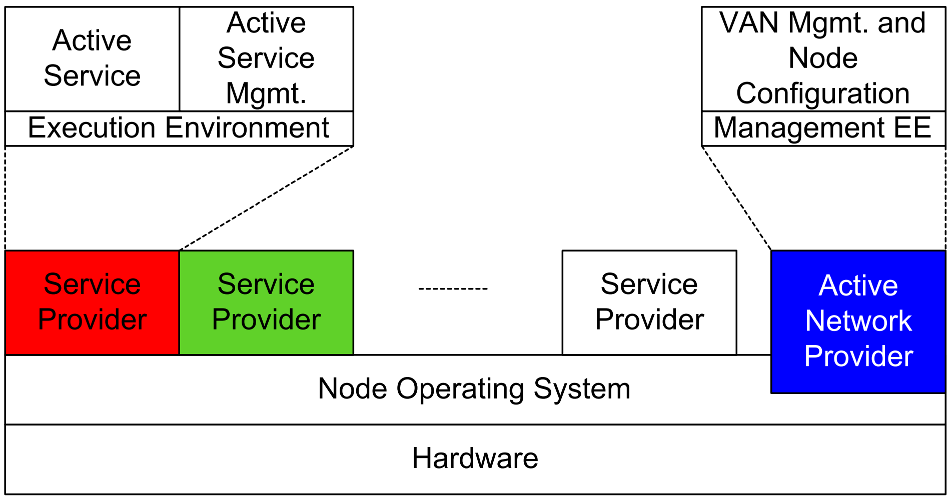

Figure 3 - 4. VAN Management Architecture

The VAN Management is an interesting solution focused on the management of Virtual Active Networks on the active nodes. Thereby, it mainly considers element-level management functionality. Furthermore, it is oriented towards the management of active nodes capable of isolating EEs from different service providers, and hence, it would not be suitable for the management of heterogeneous networks.

Section III.3 - Policy-based management of active and programmable networks

Policy-based management is an emerging technology for the management of networks that can be adapted to deal with active networks. In relation with the MANBoP requirements, policy-based network management technology eases the handling of active networks specificities. For example, policies are particularly suited for delegating management responsibility, essential to enable the customisability of network resources. Also, the policy's device-independence property is optimum for the management of heterogeneous network technologies. Finally, policies permit a more automated and distributed approach to management, taking decisions based on locally available information according to a set of rules.

Many research projects have covered the field of policy-based management. Among these, the most relevant ones for MANBoP are the Ponder and Jasmin projects. The Ponder project has been one of the first technology- and manufacturer-independent policy-based management frameworks. Aside, it defines a policy specification language from where many concepts have been re-used in MANBoP. The Jasmin project, although not being an active networks-related project, explores the automation and distribution of policies and policy-decisions. These are properties of the highest relevance also in MANBoP.

When focusing on policy-based management of active networks we realise that up to now there are not many efforts that analyse the synergies that can be obtained from joining active and policy-based network management technologies. Some of the more widely known and accepted works are Seraphim, ANDROID, PxP, A-PBM, Policy networking using active networks, Polynet, Policy specification for programmable networks and FAIN.

Hereafter, we describe all these works and comment those characteristics that are relevant to our work.

1st Ponder

The Ponder project [Damianou01] has had a good acceptance within the research community and its results have been used in many research projects needing policy-based management. Ponder defines a language and framework for specifying security policies that map onto various access control implementation mechanisms for firewalls, operating systems, databases and Java. It supports obligation policies that are event-triggered, condition-action rules for policy based management of networks and distributed systems. Ponder can also be used for security management activities such as registration of users or logging and auditing events for dealing with access to critical resources or security violations. Key concepts of the language include roles to group policies relating to a position in an organisation, relationships to define interactions between roles and management structures to define a configuration of roles and relationships pertaining to an organisational unit such as a department.

The Ponder project is focused on the specification of the policy language and the framework for its processing. Nonetheless, the actual management functionality is not considered in the project. Additionally, Ponder is oriented towards the management of passive networks. Thereby, it cannot be applied to active networks since it lacks mechanisms to handle some of the requirements that they impose. An example of these lacks is the incapability of dynamically extending the management functionality to cope with new services or resources added to the active network.

2nd Jasmin

The Jasmin project [Jasmin] aims to evaluate, enhance and implement the distribution and invocation of network management scripts with distributed network management applications. The implementation supports multiple languages and run-time systems. As part of the Jasmin project a set of classes have been added to support policy-based configuration management of Linux Diffserv nodes. In particular, general policy management language extensions, domain specific policy management language extensions and drivers mapping between domain specific policies and the underlying device-level mechanisms, have been realised.

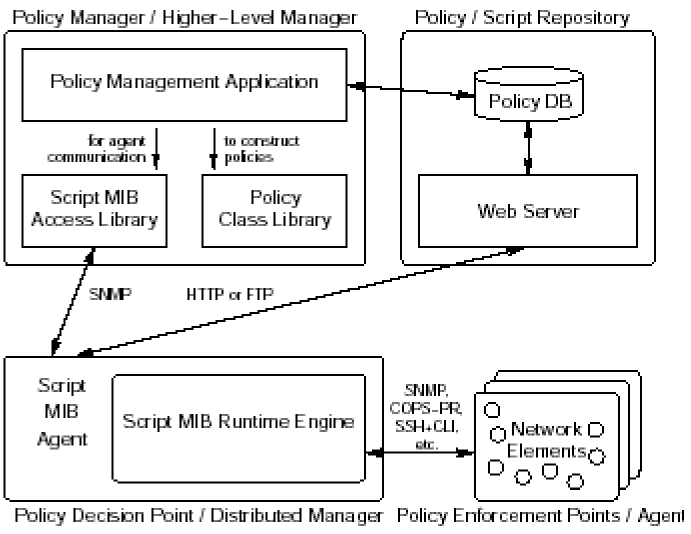
Policy Manager / Higher-Level Manager
Policy / Script Repository
Policy Management Application
Policy DB
for agent communication
to construct policies
Script MIB Access Library
Policy Class Library
Web Server
SNMP
HTTP or FTP
Script MIB Agent
Script MIB Runtime Engine
SNMP, COPS-PR, SSH+CLI, etc.
Network Elements
Policy Decision Point / Distributed Manager
Policy Enforcement Points / Agent

Figure 3 - 5. Jasmin Script MIB based management architecture

As Ponder, Jasmin is focused in the management of passive networks via scripts. It explores the distribution of policy condition monitoring and policy action enforcement although in many cases the decision is still made at the policy manager station. Aside from the capacity of managing active and programmable networks, Jasmin does not support either other MANBoP requirements. The capacity of creating different management infrastructures based on the operator needs, the delegation of management functionality and the capability of dynamically extending the management functionality are some of these requirements. However, the concepts used in Jasmin for the distribution of policy tasks and automation of policy decisions are also considered in MANBoP.

3rd Seraphim

One of the first projects to work with policies in active networks was the Seraphim project [Seraphim]. It enables the extension of the node security mechanisms by allowing the active code to dynamically install its own application-specific security functions. These code fragments, which are encapsulated inside active packets, have been named *active capabilities* (AC). An AC is able to carry not only the active code, but also the security policies customised for a particular application and even, the code needed to make a policy decision. Hence, the user "can" (in some way) establish security policies in the active node.

The active node has a framework to store, get and evaluate policies. Besides, every node has an evaluation/enforcement engine responsible for the execution of the policies loaded into the database.

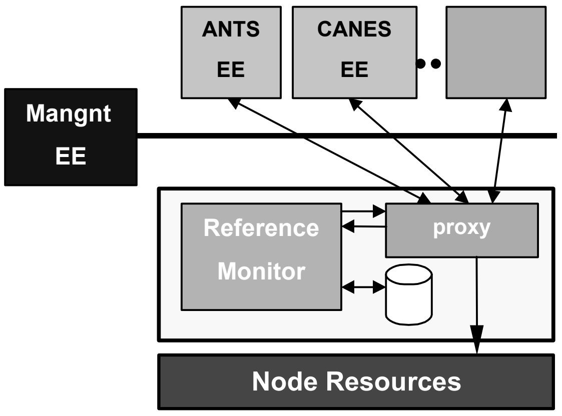

Figure 3 - 6. Seraphim security architecture

In order to assure that the AC carries well-formed expressions, the system includes an AC management infrastructure, with an administrator in charge of checking that the AC cannot compromise the system operation.

The interest of the Seraphim project resides on the fact of being the first project that explored the use of policies for the management of active networks. However, it is centered just on security management at the element-level. It does not consider management of heterogeneous networks and the management infrastructure is fix. In brief, the scope of the Seraphim project is different from the MANBoP scope.

4th ANDROID

The ANDROID (Active Network DistRibuted Open Infrastructure Development) project [ANDROID], proposes a policy- and event-driven architecture for the management of Application Layer Active Networking (ALAN) networks [ALAN], [Fry99]. The project is mainly focused to the management of active servers, where programmability up to the application level is allowed. Nonetheless, they also consider a reduced management of the active routers, i.e. configuration of users' routes towards their assigned active server.

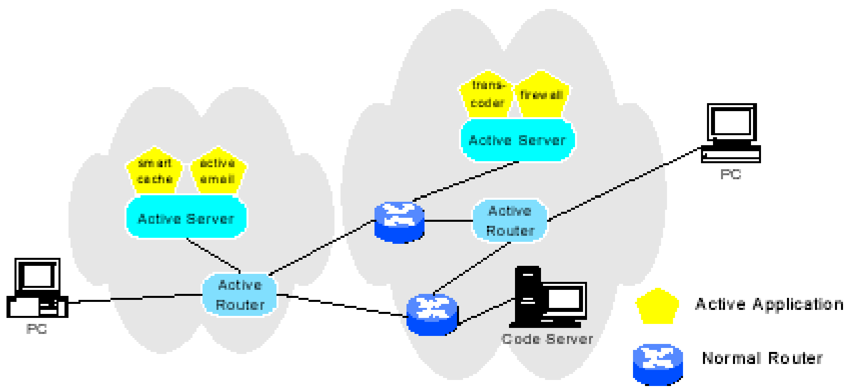

Figure 3 - 7. ANDROID Active Architecture

The ANDROID management framework is policy-based and event-driven. That is, policy actions are mainly triggered when particular events are received. Both events and policies are distributed with the Management Information Distribution (MID) system, also managed with policies.

Each ANDROID framework instance runs at least one MID server. Inside the MID, policies and events are introduced into a new XML document called Notification. The event destinations as well as the protocol that should be used to communicate with those destinations is specified in the MID by means of policies.

The XML policy defined in ANDROID carries at least six fields:

i) *Creator*: Specifies the source of the policy to establish its access rights.

ii) *Info*: Contains policy related information other than the policy itself, such as the expiration time, policies replaced by this one, etc.

iii) *Sender*: Lists the forwarding path followed by the policy

iv) *Subject*: Identifies the entities that pertain to a role that must process this policy.

v) *Trigger*: Enumerates the events that will activate the policy. When the trigger field of the policy is empty the systems assumes that the policy should be enforced immediately.

vi) *Actions*: This field can include, in addition to the policy actions, optional conditions that should be assessed before enforcing the actions.

Events are also defined in XML in ANDROID. Its structure is as follows:

i) *Event-id*: Unique identifier of the event

ii) *Time*: Specifies the time when the event was launched. It can be used by the receiving entity to reject the event if it is too old.

iii) *TimetoLive*: Establishes the time during which the information carried by the event will be relevant.

iv) *Source*: Identifies the entity that created the event.

v) *Sequence*: Integer which is incremented every time an event is sent from a particular source.

vi) *Information*: Explanatory text about the event information.

vii) *Data*: Structured information carried within the event.

When a user wants to install a new service inside an active server, it sends an event to the network operator. The operator initiates the resources and security checks, based on available policies, and then loads the active service. Active services are continuously monitored, so that if an unexpected behaviour is detected corrective policies can be enforced to correct this behaviour.

XML policies are used for managing user routes in active routers, resources and security in active servers and to manage the management information distribution systems.

Inside each active server there is at least policy infrastructure providing policy authentication, generic policy handling, policy storing.

Inside active servers the system manages, mainly, the initialisation and removal of active services, as well as the resources consumed by these services. These resources are CPU, memory and networking resources. To realise this task, resources and security management components run inside active servers. These components used the local information to carry out their task: enforcing policies and sending events.

ANDROID is focused in the management of active servers within an application-layer active network. In particular, it is focused in the management of security and resource access by services inside the ANDROID active server. Instead, MANBoP is targeted towards supporting the management of heterogeneous active, passive and programmable nodes, as well as multiple functionalities raging from security and resource sharing to traffic engineering and fault management.

5th Policy eXtension by Policy (PxP)

The Policy eXtension by Policy (PxP) project [Kanada02] suggests a mechanism for the dynamic extension of a policy-based management system. The mechanism uses policies within an active network environment to realise the extension. The Policy Extension by Policy (PxP) proposal is limited to the extension method, so it must be included within another management architecture.

The method defines two types of policies for realising this extension, i.e. Policy Definition (PD) policies and Policy Extension (PE) policies. On the one hand, PD policies allow a user to add a new type of policy into the Policy Server specifying the correct syntax and restrictions. Then, through PE policies users can specify the corresponding methods for translating the new policies types into commands on different types of network nodes. Both PD and PE policies are defined by either network operators or an application.

The architecture where this extension method has been conceived is the general policy-base management architecture containing a GUI, a policy manager (or policy server), a database and policy agents.

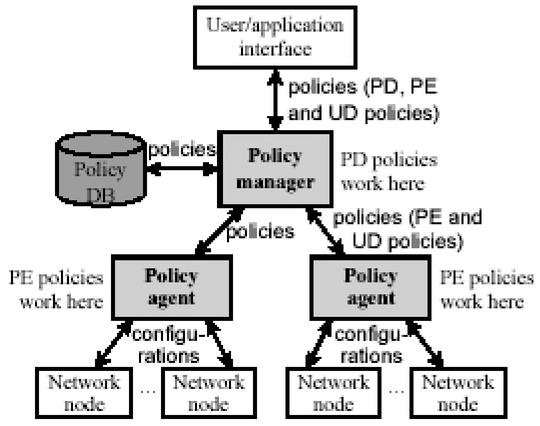

Figure 3 - 8. Policy Extension by Policy basic architecture

When a user introduces a new policy, the policy manager verifies the correctness of the policy (both syntactic and semantic) with the information contained in the corresponding PD policy.

The policy agent translates the new policy into managed device commands following the instructions contained in the corresponding PE policy. In consequence, the PE policy depends on the managed node where the policy should be enforced.

The way policies should be translated is described within PE policies by means of templates. These templates are completed with the policy information using "fillers". The "fillers" specify what information should be retrieved from the policy to complete the template. A program interpreter can be included inside each policy agent to evaluate "fillers". That is, to allow "fillers" specifying certain processing of the policy data before been included in the template.

Policy Extension by Policy (PxP) is a method for the extension of management functionality in a Policy-based management system.

Nevertheless, it only defines the extension mechanism (i.e. it does not cover the decision mechanism or the conflict checking mechanism), which should be included into a complete policy-based architecture like the one defined in this thesis, or others. Thereby, the research developed in PxP should be considered as complementary research in relation to MANBoP.

6th Active Policy-Based Management (A-PBM)

In [Fonseca01] a framework to allow the interoperability between different ISP management domains satisfying end-to-end requirements given by users is proposed. They have taken the policy-based management framework proposed by the IETF (PDP and PEP) and included capsules for the communication between the different components of this framework. Capsules represent user requirements. They are used for service negotiation and network elements configuration. The figure below shows the A-PBM architecture.

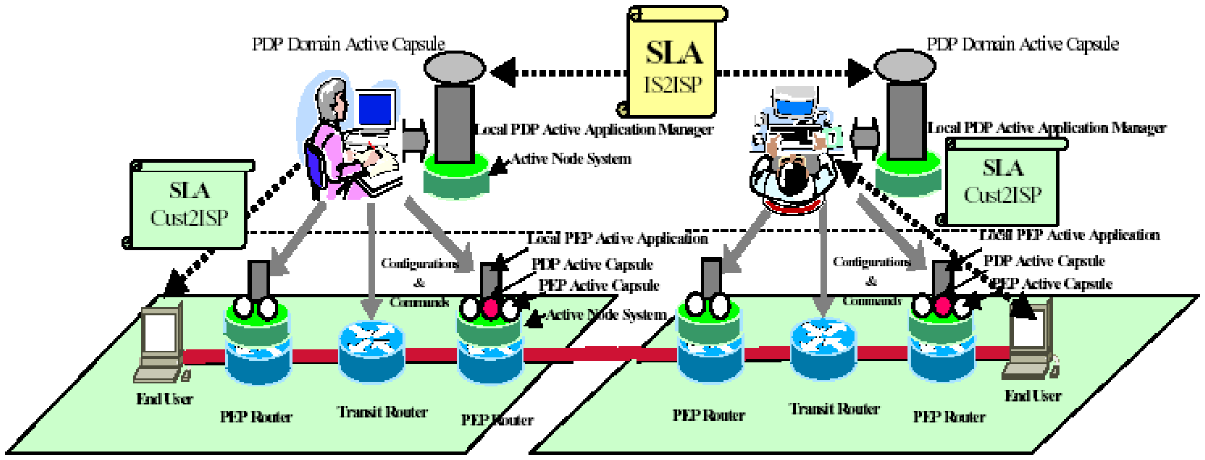
PDP Domain Active Capsule
SLA
IS2ISP
PDP Domain Active Capsule
Local PDP Active Application Manager
Active Node System
Local PDP Active Application Manager
SLA
Cust2ISP
SLA
Cust2ISP
Local PEP Active Application
PDP Active Capsule
PEP Active Capsule
Active Node System
Configurations & Commands
Local PEP Active Application
PDP Active Capsule
PEP Active Capsule
Configurations & Commands
End User
PEP Router
Transit Router
PEP Router
PEP Router
Transit Router
PEP Router
End User

Figure 3 - 9. – Architecture of interdomain A-PBM

They have defined three types of capsules: one to request decisions from the PEP to the PDP, one for notify decisions from the PDP to the PEP and a third one to negotiate between ISPs.

A-PBM focus is inter-ISP management of QoS requirements. To cope with this requirement they have created a capsule (i.e. the inter-PDP capsule) that carries out the QoS negotiation taking into account user necessities. The proposed architecture is deeply based on the IETF policy-based management architecture [Durham00a], although the COPS protocol messages have been replaced by capsules.

On the other hand, MANBoP is designed for the management of an administrative domain owned by a single network operator. It defines a more complex policy-based management architecture (than the one supported by the IETF) to cope with the introduction of new network services and to allow the operator the creation of a management infrastructure that best suits his needs. Summarising, the goals of the A-PBM research differs significantly from the goals in MANBoP, and hence also the proposed solution.

7th Policy networking using active networks

In [Kato00] a management framework designed for reducing management traffic by allowing network elements to take decisions is proposed. This is done by defining active packets, that might even contain a portion of a policy decision point, which are executed inside network elements; thus, allowing network elements to take autonomous, intelligent decisions.

In concrete, the policy server sends to network elements active packets that decide what policies should be applied at each particular time. In the figure below we can see APES (Active Program Execution System), which provides an environment to execute and control these active packets.

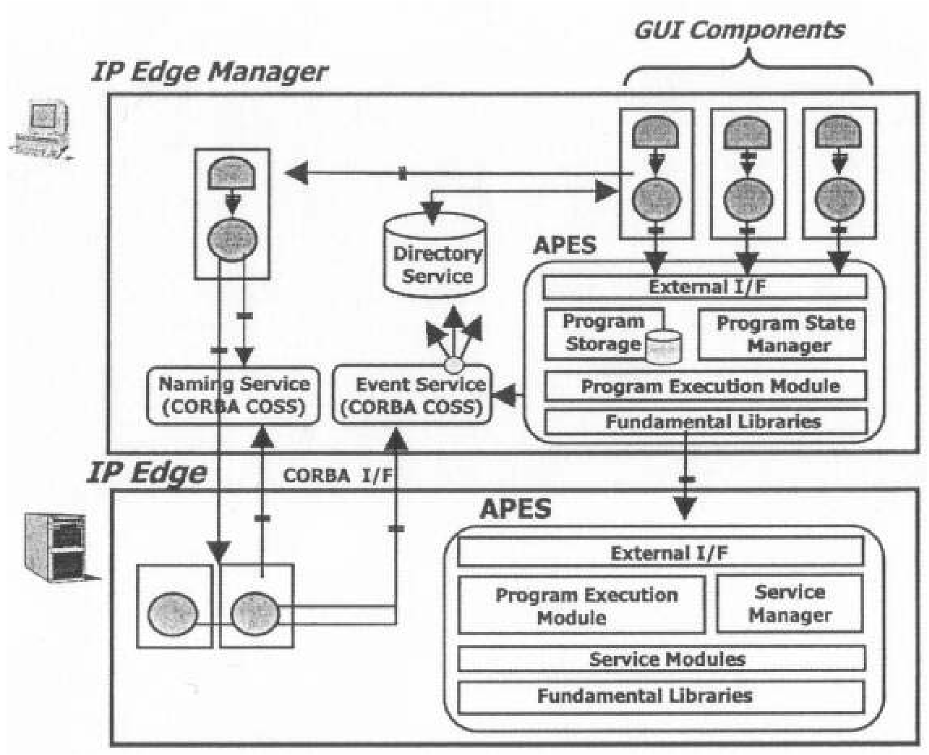

Figure 3 - 10. APES architecture

The policy is edited in the GUI and executed in APES that carries out, when appropriate, the actions specified in the policy, e.g. contacting another APES.

The proposal of replacing policies by active packets is an interesting idea, although it might not be considered (strictly speaking) a policy-based system.

However, it presents different challenges as the potential scalability problems of running a large number or policies (programs) inside an active node or how the policy conflict is managed inside the nodes. In addition, the work does not describe how service modules (which at the end represent the supported management functionality) can be extended inside APES. Furthermore, the work presented does not support the management of passive nodes, not even non APES-enabled active nodes.

MANBoP goals differ significantly and hence, the solution adopted. In MANBoP we have not considered the possibility of specifying policies as active packets. MANBoP policies are expressed in XML. The reason is that, in addition to the above mentioned problems for policies expressed as active packets, XML policies are better suited for the management of heterogeneous active, programmable and passive networks (because of its portability properties), as well as for coping with other MANBoP requirements such as delegation of management functionality.

8th Polynet

The Polynet project [Polynet] is a recent project enclosed within the Programmable Networks programme of the Engineering and Physical Sciences Research Council (EPSRC) [EPSRC].

The project will investigate the use of policies to support adaptability at three levels: (i) within network-aware applications, (ii) within application-aware networks, and (iii) at the hardware level to support adaptability in the packet forwarding "fastpath" of network elements. It will make use of the Ponder Policy Specification Language, previously reviewed, which supports both management and security policy specification, to investigate how to use policies to manage QoS, how to provide application-specific routing configurations such as multiway multicast, and to define who can program specific components and what programming operations they can access.

Because of its novelty, by the time of writing this thesis document, there are not known results of this research project yet. Therefore, it is not possible to analyse the proposed solution.

9th Policy specification for programmable networks

In [Sloman99] the research group responsible for the Ponder and Polynet projects analyse and suggest a notion and framework for specifying policies related to programmable networks. The proposal is mainly based on Ponder concepts (such as policy grouping based on their roles) that they have adapted to handle programmable network requirements. In this work, they pay special attention to authorisation policies, which determine what a user, or an active service on his behalf, will be allowed to do on the managed device.

The work presented keeps many of the limitations already described for the Ponder project, as the adaptation of the Ponder framework is very limited. For example, it is not clear how the presented work would handle the dynamic addition of new management functionality, one of the key requirements for the management of active and programmable networks. Moreover, the proposed solution just covers the policy language specification and not the actual management algorithms themselves. Nonetheless, many of the policy language specification concepts are also relevant within the scope of this thesis and might be part of the solution, such as the use of roles or domains.

10th FAIN

The main goal of the FAIN project [FAIN] 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.

The management architecture developed within the FAIN project mixes policy based network management and active network technologies in order to cope with the main requirements for the management of active networks. More specifically, the FAIN management architecture is a two-tier management architecture that consists of one network-level management station and several element-level management stations (potentially one per managed device). The figure below shows a simplified version of the FAIN management architecture.

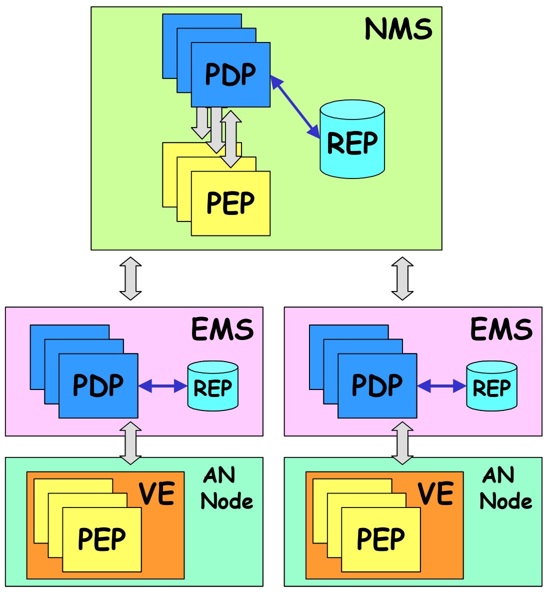

Figure 3 - 11. FAIN management architecture

The architecture is oriented towards the management of FAIN nodes and hence it is linked with other FAIN systems as the Active Service Provisioning system (ASP) [FAIN03c] or Virtual Environments on FAIN nodes

[FAIN03d]. Indeed some of the FAIN components at the element-level run inside a Priviliged Virtual Environment inside the node.

FAIN supports dynamic extension of management functionality by downloading couples of Policy Decision Point (PDP) and Policy Enforcement Point (PEP) components. These modules contain the logic for taking decisions and enforcing new policies respectively.

Finally, the delegation mechanism implemented in FAIN is mainly based on the creation of isolated environments assigned to users that obtain delegated management functionality. Within these environments, called Management Instances, the users are allowed to do anything with the management functionality that they have obtained through delegation or even introduce their own management code. The access rights authorisation is done inside the active node itself when requests coming from these management instances are received.

FAIN is the closest project to MANBoP in terms of both goals and scope. Indeed, MANBoP is heavily based on the work done in FAIN by the author of this thesis. However, in MANBoP we have widened the scope by including the management of heterogeneous active, programmable and passive networks, not supported in FAIN, and by making more flexible the management infrastructure, which is a fix two-tier architecture in FAIN. This higher flexibility in the management infrastructure allows the network operator to better adapt the management infrastructure to his particular needs in terms of costs, scalability or others. Another aspect not included within the scope of FAIN and included in MANBoP is the support for the dynamic addition or removal of new nodes within the managed network.

Furthermore, in MANBoP we have tried to enhance some particular aspects of the FAIN solution by completely modifying the extensibility and delegation mechanisms.

The widened project scope, the new requirements as well as the new extensibility and delegation mechanism adopted in MANBoP causes that although the goals, scope and concepts in both cases are very similar the solution adopted will differ significantly.

Section III.4 - Trends and expected evolution

In our context, network management [Haas01] means deploying and coordinating resources in order to administer and operate heterogeneous networks, with the objective of achieving the required quality of service (QoS), thus fulfilling the expectations of both the owners and the users of the network [Sloman94]. Methods to predict [Galtier01] or rapidly detect failures [Hood97] and alert the relevant personnel to take remedial action can substantially reduce user inconvenience.

There are two main areas of interest in the combined research area of network management and active networking. These are the requirements for a management system to manage an active network and the role of an active network to reduce the load on any management system. By doing so, the management process is improved in a specific manner compared to non-active approaches.

The general trend in network management is to achieve scalability in functionality. The research community constantly comes up with novel ideas for optimising efficiency and functionalities, only to fall short of having global end-to-end management capabilities. SNMP is still the de facto management protocol on the Internet.

In recent years, new management paradigm proposals tried to overcome some of the key deficiencies of SNMP. The Management by Delegation (MbD) [Goldszmidt95] paradigm proposes a distributed hierarchy of managers that solves the problem of the management traffic generated because of the periodic polling of data between the manager and the agent. MbD was expected to be a scalable proposition when compared to the SNMP model because if data analysis is only conducted at the management station (as is the case for the latter), it will require data access and processing rates that do not scale up for large and complex networks (e.g., the Internet)

While the MbD approach is a trend away from centralised approach, i.e., pushing intelligence from management system to managed element (using mobile agents for code mobility), the policy-based approach is a trend towards simplification of configuration by means of high-level rules.

The introduction of automation of management tasks involves the most significant change with respect to current implementations of management tools with existing technologies (e.g., SNMP). The mobile agent [Sugauchi99] and active networking [Kawamura00] technologies have been extensively investigated over the last several years for this primary interest. The programmable networking paradigm offers the possibility of utilising dedicated plugins for per-flow monitoring.

Automation in the network environment [Greenwood99] has been proposed many times during the past 15 or so years, for example, in routing at switches, and it is arguable that a large-scale adoption and implementation is around the corner. However, operators have been concerned about adopting extensive automation.

To cope with interoperability and interworking, middleware technologies like CORBA and Java RMI are gaining relevance in the management area [Open].

Section III.5 – Conclusions

The review on the state of the art of programmable networks management given in this chapter has been divided in two main sections, the first one dealing with non policy-based proposals and the second one focusing on policy-based management of programmable networks.

There are a number of research projects that cover the field of programmable network management using non policy-based approaches. Most of these projects are indeed using programmable network techniques so as to achieve a more efficient management. Among them, the ones reviewed are: ABone, ABLE, AVNMP, Smart Packets, SENCOMM and VAN.

The ABone has very limited functionality from the management point of view. It is just focused in the management of ABone nodes.

ABLE is not a management system itself but a facility for a management system. It allows a management system to manage a passive router with active packets.

The AVNMP project suggests a prediction algorithm for enhancing system's management by avoiding potential future problems or pre-configuring the network for future events.

Smart Packets is based on active packets, written in a tightly-encoded safe language, carrying programs that are executed at nodes on the path to one or more target hosts.

The SENCOMM project extends the Smart Packets mechanism by including it within a broader management system. Such a system is structured around a management center that takes all decisions. The managed devices must contain a SENCOMM Management EE.

The VAN Management is an interesting solution focused on the management of Virtual Active Networks on the active nodes. The Virtual Active Network (VAN) management framework defines two types of EE: the management EE that works on the management plane, and the service provider EE that works on the data transfer as well as on the control plane.

Besides, we have also analysed a number of policy-based management of active networks approaches: Seraphim, ANDROID, Policy eXtension by Policy (PxP), Active Policy-based Management (A-PBM), Policy networking using active networks, Polynet, Policy specification for programmable networks and FAIN. Additionally, although they are not oriented towards the management of active networks, we have also reviewed the Ponder and Jasmin projects because of their relevance in the policy-based network management research field.

The Ponder project is focused on the specification of the policy language and the framework for its processing. Nonetheless, the actual management functionality is not considered in the project.

Jasmin explores the distribution of policy condition monitoring and policy action enforcement via scripts. However, in many cases the decision is still made at the policy manager station.

Seraphim was the first project that explored the use of policies for the management of active networks. It is centered just on security management at the element-level.

ANDROID is focused in the management of active servers within an application-layer active network. In particular, it is focused in the management of security and resource access by services inside the ANDROID active server.

Policy Extension by Policy (PxP) defines a method for the extension of management functionality in a Policy-based management system by means of policies. It does not cover any other aspect of a policy-based architecture.

A-PBM defines a system, based on the IETF policy-based management architecture, focused in inter-ISP management of QoS requirements. To cope with this requirement they have created a capsule (i.e. the inter-PDP capsule) that carries out the QoS negotiation taking into account user necessities.

In the policy networking using active networks approach the policy server creates an active packet that travels through the specified network elements deciding what policies should be applied at each particular time.

The Polynet project makes use of the Ponder policy specification language to investigate the use of policies to support adaptability at three levels: application, network and element level.

Policy specification for programmable networks is an study for analysing and suggesting a notion and framework for specifying policies related to programmable networks.

FAIN defines a two-tier, policy-based management architecture oriented to the management of FAIN nodes. The architecture supports dynamic extensibility of management functionality and delegation.

Except FAIN, any of the above projects has similar objectives to those in MANBoP. Some of them like ABone covered just a subpart of the MANBoP functionality and others like AVNMP or A-PBM had a different goal. Nevertheless, in some cases the mechanisms explored in some of these projects like the Smart Packets, the AVNMP or the PxP can be supported, if desired, by the MANBoP framework to manage the network elements.

Furthermore, some of the reviewed projects, like the Smart Packets and Seraphim, covered only a small part of the FCAPS management functional areas. Others, like the VAN and ANDROID projects, only supported element level management functionality.

The project that is closer to MANBoP in terms of goals and concepts developed is the FAIN project. The reason is that MANBoP can be seen as a continuation of the work developed by the author within the FAIN project. However, MANBoP both extends the FAIN scope and modifies some FAIN mechanisms. Thereby, we have seen that there is significant difference between the approaches of both projects.

Recapitulating, the three aspects covered in MANBoP and not addressed by any of the reviewed state of the art projects, not even FAIN are:

a. The support of heterogeneous active, programmable and passive networks. Those projects dealing with active networks did not support passive nodes and vice versa.

b. The capability of dealing with the dynamic addition or removal of nodes within the management infrastructure (this facet is just needed in those systems containing network-level management functionality).

c. The possibility of easily creating the management infrastructure that best suits the network operator needs.

These three properties represent the most innovative aspects of the MANBoP framework. The filling of these gaps within the current state of the art is the main justification for the research developed in this thesis. Nevertheless, there are other smaller innovative aspects also researched in MANBoP like the delegation approach taken.

In the following chapter we will describe in detail the design of the solution proposed for the MANBoP project. This solution must handle the requirements listed in the second chapter and shares several concepts and ideas with some of the projects reviewed in this chapter.