IMPLEMENTATION ISSUES OF POLICY BASED NETWORK MANAGEMENT SYSTEMS

E.Grampín 1, J.Rubio 1, N.Vardalachos 2, A.Galis 2, J.Serrat 1

1 Universitat Politècnica Catalunya, Dept Teoria Senyal i Comunicacions, Jordi Girona 1-3, 08034 Barcelona, Spain, tel.: +34934016786, e-mail: {grampin, jrubio, serrat}@tsc.upc.es
2 University College of London, Dept. of Electrical Engineering, Torrington Place, London WC1E7JE, UK tel.: +442076793956, e-mail: {nvardala, a.galis}@ee.ucl.ac.uk

Abstract
Policy Based Network Management has been presented as a paradigm for efficient and customisable management systems. The IETF has provided a framework to describe the concept but the implementation of such idea is left out of recommendation. Undoubtedly, a physical architecture is needed to drive the implementation of such systems and get rid of the full advantages of the concept. This paper presents an implementation architecture and existing tools to be used to put it into practice.

Keywords: Network Management, Policy Based Management

1. Introduction

The behaviour of management systems is based on sets of rules. Sometimes, this rules are rather simple as usually happens in configuration management, or they can be very complex like the rules driving an alarm correlation algorithm usually encountered in fault management systems [1].

Regardless of their complexity, what is a factor characterising the “conventional systems” is that these rules are hard coded. In other words they haven’t been conceived to be changed during the system lifetime. They are defined at the design phase and coded in the software of the operation systems. Although management systems conceived with this strategy have been proven useful [2], service providers and network operators and even the final users are demanding today easy and flexible deployment of new services that usually entail a high level of customisation.

A system based on traditional operation systems is not easily customisable. Some times they require the complete change of software modules if not the whole application. Therefore, new software paradigms have been looked for to overcome the rigidity of conventional systems. These new paradigms should be based on the online programmability of the rules governing the management system. Rules defined at any time by an authorised actor. Rules in turn that could be modified or replaced if necessary. Rules that at the same time were able to hide the peculiarities of the physical devices implementing a network and therefore were able to be defined at high level, i.e. at user level. All these characteristics are the defining attributes of the Policy Based Management paradigm. Therefore, a policy based managed system allows the definition of its driving rules at any time making such system more flexible, scalable and customisable [3].

Considerable attention has been paid to develop such systems from different points of view: policy characterisation, functional architecture, etc. [4]. At the same time, specific solutions based on such concept have appeared in the market [5] to [10]. Nevertheless, there is still a long way to get a system having the full advantages that have been adhered to the concept of policy based management. With this objective in mind this work presents some results concerning implementation aspects of the generic framework of a PBNM system. In the next section we present the functional architecture that has been used as a reference. The following section describes the implementation properly said and in the final one, we look at the existing tools in the market that could be used to support part of the framework.

2. A Policy Based Network Management functional architecture

The concept of management policy is related to one or more rules which in turn are decomposed in conditions and actions. Whenever the condition(s) is fulfilled, the action(s) is executed. The following is an example of what is just said.

Assume we want to Provide a jitter free MPEG2 video service for authorised users between authorised points at agreed-upon times. We define a rule constituted by the following conditions:

Provide a jitter free MPEG2 video service for authorised users between authorised points at agreed-upon times.

This rule has been decomposed into conditions and actions. The conditions are:
1. Authorised users
2. Authorised points
3. Agreed-upon times

The actions are:
1. Provide a jitter free MPEG2 video service
IF( user IN ApprovedUsers) 
AND( service IN VideoServices) 
AND( source IN VideoSources) 
AND( destination IN VideoDestinations) 
AND( time IN ApprovedTimePeriods)

and an action that is stated as:

THEN provide JitterFreeMpeg2

The scope of the set of rules that will drive the management of a system or service can be defined in any application domain and at any functional level. Therefore we can talk about Security Policies, Access Policies, QoS Policies, Network-element Level Policies, Network Level Policies, etc. On the other hand, policies must be unambiguous and verifiable and appropriate mechanism must exist for defining, combining, storing and accessing policies.

The operation of the PBNM system can be described by means of the architecture proposed by the IETF that is depicted in figure 1 and whose functional components are described hereafter.

- **Policy console**

  Policies are considered to be managed by an administrator or any other actor adopting this role. The management of policies will be executed through this system component acting as a user interface to allow constructing policies, deploying policies, and monitoring the status of the policy-managed environment.

A PBNM system needs tools for policy specification. Various specification languages can be used in the context of policies, therefore various policy formats exist. The following are the most usual
1. Natural Language format: supported by user-friendly GUI for customer to input or review policy in easy way. This depends on the design of GUI. This format must be mapping to the markup language format.

2. Markup language format: this format can be processed and interpreted by a computer. But it has not mapped to the program code that can be executed directly. The most famous example of this kind of language is XML.

3. Rule-based format: it interprets the policy as a sequence of rules, in which each rule is in the form of a simple condition-action pair. Policies specified in this fashion are easier to analyse than policies specified by a markup language or Java classes. IETF has chosen a rule-based policy representation in its specification.

- Policy decision point (PDP)
  It is the entity that decides if the conditions of a policy are fulfilled and as a consequence triggers the actions involved in that policy. Besides this function that is considered the main one, we can also attribute to this component the detection of policy conflicts, the retrieval of the relevant policy when required by an external trigger event and the interaction with the PEP component described hereafter.

- Policy enforcement point (PEP)
  It is the entity that ensures that the actions ordered by the PDP are executed. It has also the role of metering and monitoring for auditing of policy compliance.

- Policy repository
  A directory and/or other storage service (e.g. relational database) where policies and related information are stored.

- Policy communication protocols
  Appropriate communication protocols are also needed for data exchange between the above mentioned entities. The Common Object Policy Service protocol (COPS) is often mentioned to support the communication between the PDP and the PEP. Nevertheless, other mechanisms like SOAP (Simple Object Access Protocol) and XML-RPC (XML Remote Procedure Call) have also been proposed. In respect to the access to the repository, LDAP or SQL are often used depending that the repository is implemented as a directory or as a database respectively.

3. Implementation model

Figure 2 shows an implementation architecture of the above described model, whose building blocks are described hereafter.

![Figure 2. A Policy Based Network Management System Implementation Model](image-url)
• **Policy Editor**
The Policy Editor acts as the interface between the policy management system and the system administrator. It compiles the policies creating a policy class which in turn will be a Policy Object at runtime. On the other hand this module sends the policies edited by the system administrator to the Policy Repository.

• **Policy Repository**
It is intended to manage a distributed hierarchy of policy objects grouped in domains. The Repository may be implemented using an LDAP server that generates events for changes to the membership of a directory and allows object to be members of more than one domain.

• **Policy Objects (PO)**
Policy Objects are the entities modelling the policies defined within the system. The key for distributing policies is to determine which enforcement agent is appropriated for resolving a given policy. In this model the policy object will evaluate the target set of managed objects to which it is associated and will determine the access controller for each target object in the target set. This is represented in figure 2 by means of the arrows associating the PO bloc, the AC bloc and the Policy Repository.

• **Enforcement Agents and Access Controller (AC)**
The Enforcement Agent is the subsystem that implements the functionality of the PDP and the PEP of the model of figure 1. The Access Controller is a constituting part of the Enforcement Agent to differentiate the different sets of managed objects. In this implementation model each managed object has a single access controller (AC), which enforces all the policies for the target object. The ACs send actions to the managed objects and receive their notifications.

4. **Policy Based Network Management tools**

Many PBNM tools are emerging, both in the academic and the commercial world. The following is devoted to reference these tools and how they fit into the implementation architecture described above. Hereafter we present some of them grouped in three categories, namely Open Source tools, commercial tools and policy definition language tools.

4.1. **Open Source tools**

Intel [10] and Iphighway [11], had opened their client COPS implementation in order to promote protocol adoption by router vendors. Developers can use these SDKs to develop COPS clients (the PEP side) tailored to be used in conjunction with a Policy Server, not provided by these companies. The Iphighway SDK implements also the Proxy Client variant, useful to interface with non-COPS devices.

Vovida [12] is a communications community site dedicated to provide a forum for open source software used in data communications and telecom environments. These community has a full COPS implementation which matches with their mainstream VOCAL application, a VoIP development environment.


In order to be used in a PBNM project, these tools have a supporting role, since they don’t implement the PDP and/or PEP functionality. They can be used to developed policy-enabled Element Managers, since COPS is focused in communication with networking devices.

4.2. **Commercial tools**

IPHighway’s PerformancePro [5] provisioning solution allows enterprises and service providers to add Quality of Service (QoS) capabilities to their wide-area networks. The PerformancePro solution runs on distributed policy servers and configures network devices to deliver predictable network performance.
PerformancePro permits to use simple and powerful business vocabulary to provision bandwidth and prioritise application traffic by means of a policy definition GUI. For example, business-critical applications can be given precedence over casual web browsing and time sensitive e-business transactions can be guaranteed lower latency over delay-tolerant applications like file transfers. This tool has a public API for XML access to the Policy Administrator.

**Nortel Policy Services** [6] provides an integrated IP service management solution that delivers directory-based functionality to manage IP addressing and QoS. It also manages RADIUS user profiles and IP VPN service provisioning, with a central data repository.

Nortel Policy Services is a modular solution, composed by Dynamic Host Configuration Protocol (DHCP), Domain Name Service (DNS), Quality of Service (QoS), RADIUS, and IP VPN service provisioning, with each product within the solution providing best-in-class standalone capabilities for IP address management, QoS, RADIUS and IP VPN Service Provisioning. This tool provides some extensibility by means of a LDAP API for access to the Directory and a Java API to interface with the system’s GUI.

**Orchestream** offers a product named Service Activator 3.0 [7]. This package implements the MPLS/BGP VPN model [8]. It also features an Integration Module, a set of APIs that allow external applications to have access to internal information about the services created on the network. For instance, a billing application can direct Orchestream to add a new VPN site, and bill for that new service once activated. On the other side, for example, Orchestream can send fault traps to a fault management package. They can be reported as SNMP traps or accessed via a CORBA interface.

**Cisco** has a set of management applications, with some frameworks for OSS integration, namely Cisco Element Management Framework, Cisco Assure Policy and Cisco Networking Services.

**Cisco Networking Services** (CNS) is a set of Directory Enabled Networking (DEN) software development tools, including an implementation of DEN and DEN-derived Cisco Lightweight Directory Access Protocol (LDAP) schema, LDAP client software, and scalable event services.

Cisco has different policy-based tools grouped under the name **Cisco Assure Policy Networking**. These tools are the QoS Policy Manager (QPM) [8], Cisco Secure Policy Manager and Cisco Network Registrar.

Cisco’s QoS Policy Manager, which provides centralised policy control and automated policy deployment for enterprise networks, use CLI commands for configuration management and can use COPS for carrying policy information between the PDP and the PEP when an RSVP request is received by the device (outsourced model). Nevertheless, these tools have a poor support for integration with external applications.

**HP** has also a set of management applications [9]. **The PolicyXpert** [14] covers the Performance and Service management areas. It basic function is to intuitively manage bandwidth allocation and quality of service (QoS). It provides capabilities to control IP QoS and offer differentiated services and also RSVP signalled QoS management.

PolicyXpert implements a three-tier distributed architecture: console, server, and agents. These may be run on one system or distributed, where multiple consoles and agents can connect to the policy server. COPS is used to communicate between the PolicyXpert components. Proxy agents, developed by device’s manufacturers, are used as translators between the policy server and each network device. A backup server can also be configured for automatic switch-over.

### 4.3. Policy Definition Languages/Tools

**The Routing Policy Specification Language** (RPSL) [15] allows a network operator to specify routing policies at various levels in the Internet hierarchy; for example at the Autonomous System (AS) level. At the same time, policies can be specified with sufficient detail in RPSL so that low level router configurations can be generated from them.

**RAToolSet** [16] is a set of tools from the Routing Arbiter which actually implements RPSL for the definition of routing policies.

**Ilog Rules** [17] is a general-purpose expert-system generator. Rule-based applications can be build based on this tool, by combination of rule-based techniques and object-oriented programming. This tool provides ways to add decision support and data flow control functions to applications. It is tailored for developing intelligent agents and business rule processors. This tool integrates with a given application applying rule-based decisions using the user-defined objects.
This tool has its own Policy Language. The definition of policies can be done through the Policy Console, and rules can be compiled to speed up application. Rules can also be imported from an XML file.

Ponder [18] is a declarative, object-oriented language for specifying security and management policy for distributed object systems. The language includes constructs for specifying the following basic policy types: authorisation policies that define permitted actions; event-triggered obligation policies that define actions to be performed by manager agents; refrain policies that define actions that subjects must refrain from performing; and delegation policies that define what authorisations can be delegated and to whom.

Ponder implements the Policy Console and the Compiler. The enforcement agents, that performs both the PDP and PEP functions, need to be developed by the user in order to realise a Policy Server.

5. Conclusions

A physical architecture for a policy based management system is proposed with the goal of getting full advantage of the policy based management concept. This architecture identifies four main building blocks whose functionality is specified and matched to the generic IETF framework entities.

Many policy tools have been reviewed with the intention of selecting some of them in the implementation of the proposed architecture. The commercial tools have manufacturer-specific features, have been conceived to work as standalone systems and therefore are difficult to integrate into a PBNM development. Policy specification languages can be used at least as Policy Editor for policy definition as well as for policy compiler that generates the Policy Objects. Some of them, like IlogRules, provide part of the functionality of the Enforcement Agent. These tools are rather easy to integrate with user developed software. At the Element Manager level, COPS libraries are available among the analysed Open Source tools that and can be used in order to communicate with network devices.

Existing market and freeware tools can be used to partially implement the proposed architecture. Nevertheless it is necessary a careful evaluation of their benefits in front of the difficulties of a proprietary development versus the overheads that usually impose a generic tool. Work currently in progress is addressing this subject and it is expected to have the preliminary results on a system prototype in the next coming months.

Acknowledgements

This paper describes work undertaken and in progress in the context of the WINMAN, a two and a half year IST project during 2000-2002. The IST programme is partially funded by the Commission of the European Union.

References