PCIMe for Self-Aware Networks

Famory Touré, and Armen Aghasaryan
Alcatel R&I, Route de Nozay, 91461 Marcoussis, France
{famory.toure, armen.aghanaryan}@alcatel.fr

Abstract. Policy-based management integrates two very different information models, an IP packet based model, and a system/device/network resource based model. The Internet standards or drafts incorporate the two models, however, the IP packet based model is much more used in today’s policy-based management systems. We propose a more extensive usage of the second model coupled with a rule engine in order to build PCIMe based Self-Aware nodes.

1. Introduction

The trend of self-managed networks implies more and more intensive automation of management functions. At the same time, it is accompanied with the distribution of management processes leading to the transition from a centralized mode of management to a distributed one which in its turn reinforces the control plane and the intelligence embedded in network nodes. Then, self-awareness of nodes can be defined as a capacity of a node to instantiate its model and to use it in the local realization of globally coherent management functions. Policy-based management is a widely accepted solution that promotes the automation of the network configuration. It suggests declarative approach of policies (sets of rules) to define the global behavior of the network, to manage and to control its resources. These rules are then translated into commands that are enforced on the equipments through various protocols such as CLI, SNMP or COPS.

Implicit and explicit variables in PCIME

The PCIMe [1][2] model proposed by the IETF is now widely accepted in the IP world for the representation of policies; it is based on CIM (Common Information Model) defined by DMTF [3] to cover larger scope of system/device/network resources. Rules are usually expressed using ‘implicit variable’ which means that the objects they refer to are not explicitly represented in the CIM model. Typically, these rules are expressed on IP packets, for instance, the rule <if source_address matches 188.9.12.0/24 then set DSCP to EF> cannot be evaluated using the values of model objects; its evaluation will be done implicitly at the correspondingly configured network nodes. However, another type of variable known as ‘explicit variable’ has been defined in order to requence a property of an object defined in the system/device/network resource model. Example: <if alert_indication.event is "congestion" then <if source_address matches 188.9.12.0/24 then set DSCP to EF>>. This rule will be evaluated using the ‘alert_indication’ object at the reception of a congestion event. Once all the conditions with explicit variables are satisfied, the rule is installed. Similarly, whenever such conditions are invalidated, the rule is uninstalled. Our research focuses on how these variables can be used, coupled with a rule engine to install or uninstall policies according to external events received on self-aware nodes or on Policy Decision Points.

Meta-policies

A similar approach based on meta-policies has been proposed in [4], and allows the installation/un-installation of policies in a PEP. These rules are then evaluated in the PEP when necessary, giving thus a self-awareness feature to the PEP. An XML description of the rule has been proposed for that purpose, as well as a PIB format to carry the rules from the PDP to the PEP using COPS-PR. The rules evaluation is done by an ad-hoc rule evaluator. The model used for the representation of the rules does not use the PCIMe standard.
2 The proposed solution

The approach we develop consists in using the PCIMe model “as is” and to use a rule engine for the evaluation of the explicit conditions (conditions that contain explicit variables) either in the PDP or in the PEP. So one needs to translate the rules expressed in PCIMe into the format used by the rule engine. Note that only two kinds of predefined actions, install and uninstall, will be present in the resulting rules.

Coupling with a rule engine

Rule engines are more and more used in various areas. Among them, one finds commercial tools such as Jrules from Ilog, QuickRules from Yasutech, and open-source source ones such as Jess, Mandarax and Drools [5]. Several rule languages have been already defined for interoperability between rule engines. A consensus seems to be reached on RuleML which is XML based and a corresponding Java API (JSR 94) has been proposed. However, the model of RuleML is generalist and very far from PCIMe.

We define an XML representation of the PCIMe classes in order to map the explicit conditions into the rule engine conditions. This allows to work with a well-known PCIMe model and to re-use the provisioning tools already available for transformation of the rules into a Diffserv PIB or CLI commands.

Closed-loop mechanisms for self-management

In our approach, the objects of the system/device/network resource model are represented in the rule engine in XML or in Java, and on reception of external events they are asserted in the working memory. The facility of reaction on external events by installing/uninstalling the rules allows the construction of closed-loop mechanisms that automate the management functions (self-management): for example, fault management with self-healing, dynamic traffic policing and access control adaptation, and dynamic traffic engineering (pipe resizing and rerouting). The fact that the rule engine uses optimized algorithms such as Rete or backward chaining ensures a better efficiency in the performance of such closed-loops.

The loop can be closed either at the management level, through a PDP which will integrate the PCIMe/CIM model and the rule engine, or at the control level. In the first case, this allows to rely on legacy devices that will not easily support self-aware functions. In the second case, a local PDP can be embedded in the network equipment or in a proxy PEP, thus re-using the same functional blocs and rule engine integration as a centralized PDP. Such a model-based functionality embedded in the network equipment realizes the concept of self-awareness.

3 Conclusion

We promote the concept of self-awareness through the usage of PCIMe model at management or node level. This demonstrates that the current models and architecture proposed by the IETF can be re-used ‘as is’ for installing/uninstalling policy rules to have a dynamic and adaptable network behavior. Different realization scenarios are possible: centralized PDP, local PDP embedded in a PEP proxy, and finally the all-embedded solution. As a consequence of this dynamic and decentralized approach the management processes become more robust. At the same time, the issues like conflict detection/resolution in rules become more challenging.

References