POLICY-BASED NETWORK MANAGEMENT WITH SNMP

S. Boros[1]

[1]Computer Science Department, University of Twente,
P.O. Box 217, 7500 AE Enschede, The Netherlands,
e-mail: boros@cs.utwente.nl

ABSTRACT

This paper presents a way of managing configuration of network elements via a set of high-level rules or business policies rather than managing device by device. First, there is a need for abstraction of the capabilities of the individual devices, thus switching the control to network level. The benefit is that we can specify and issue unique management rules (or commands) that can be applied to a cross-section of network devices.
Second, a way to manipulate (create, install, monitor, modify, revoke) the policy rules across the network is needed. The IETF SNMPConf (Configuration Management with SNMP [3]) working group explores how the Internet Standard Management Framework SNMP can be used for policy-based configuration management. The material presented in this paper is heavily based on the work of this group.

1 INTRODUCTION

In the current Internet world configuration management systems are developed based on the Internet Standard Management Framework SNMP.
The SNMP architecture [6] involves nodes acting as management stations (managers) and nodes acting as managed entities (agents). Agents have access to management instrumentation, run a command responder application and a notification originator. A manager contains a command generator and a notification receiver. Management entities control and monitor managed elements. Examples of managed entities are routers or hosts. To convey management information between the managed and the management entities the Simple Network Management Protocol (SNMP) is used. In order to achieve robustness of management, the SNMP protocol uses the connectionless transport protocol UDP.
Monitoring and controlling functions are achieved by polling the agents and by notification of manager by the agent (by means of traps and inform messages).
Management information is stored in a virtual information store known as Management Information Base (MIB). Objects in MIBs are organized in a way that is described by Structure of Management Information SMI. The language used to describe MIB objects is a reduced set of ASN.1 constructions. This only allows existence of scalars and two dimensional arrays in the MIBs. Extension of management information is also possible by creating new MIBs or augmenting existing ones.

While traditional SNMP-based configuration management enables a device-by-device configuration of network elements, increased size and complexity of them turned the configuration into a more difficult task. The increase in size means more devices to configure, the increase in complexity means that devices are of different types and from different vendors and perform far more operations.
In order to cope with these difficulties, policy-based configuration management started to be developed. This type of configuration management consists of a set of configuration operations performed on (potentially many) different network elements in order to produce a coherent network behaviour over an autonomous domain.
Managing large networks as a whole has the following benefits:

- consistent behaviour of the network
- higher reliability
- higher efficiency

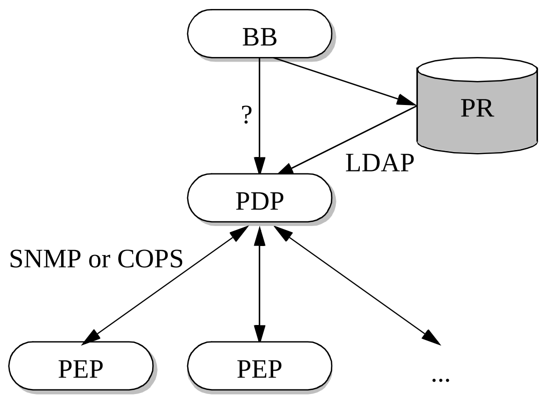
but it also carries several problems and difficulties:

- a single erroneous high-level policy will negatively affect the overall behaviour of a network

- security leaks are possible that can affect the whole network
- because of multiple devices, incompatibility issues can appear (this depends however on the level of abstraction, discussed later)

It is possible to think of policy-based configuration management in high-level terms: we can say that policy-based management is about implementation of a set of rules which command the allocation of network resources on a per user or application basis to best support established business objectives. However, from the network service users' point of view, policy-based configuration management means receiving appropriate treatment from the network for critical applications. From the network operator's point of view, policy-based management is about minimizing the complexity of end-to-end management and security.
The scope of policy-based configuration management primarily includes QoS (bandwidth, latency, priority) and security (authentication, authorization, auditing).

Figure 1 shows the policy-based configuration management architecture [2] that is currently being used within the IETF. Policies are created by Bandwidth Brokers and stored in policy repositories. Policy Decision Points retrieve the stored policies. They interpret and validate them and send them to Policy Enforcement Points (routers, bridges) to enforce them.



BB - Bandwidth Broker
PDP - Policy Decision Point
PEP - Policy Enforcement Point
PR - Policy Repository

Figure 1. Policy-based management architecture within IETF

Functions of the PDP include: retrieving policies, interpreting policies, detecting policy conflicts, receiving role descriptions, receiving policy decision requests from PEPs and returning policy decisions to them. PDPs also send asynchronous policy decisions based on updates or external requests [9].
Policy enforcement involves the PEP applying actions according to the PDP's decision and based on current network conditions. These conditions can be static (source/destination IP address) or dynamic (current bandwidth availability, time of the day) [9].
The work within the IETF concentrates on questions like: which protocols to be used between different components. The Policy and RAP [4] working groups concentrate on LDAP and COPS. The SNMPConf working group is laying the fundamentals of SNMP based configuration management.

1.1 COPS approach

The COPS [5] approach translates policies into configuration instructions, which are downloaded to network devices via a protocol: Common Open Policy Services (COPS) Protocol. Configuration information is stored, along with user, device and application information, in a Lightweight Directory Access Protocol (LDAP) compliant directory system so multiple network applications can share and make decisions based on the information. Information in the directory is updated dynamically. The network will dynamically learn of changes from the directory and will reconfigure itself to ensure that QoS and security (and possibly other) policies are appropriately applied.
This approach holds several inconveniences:
- it requires development of a new protocol: COPS
- it uses PIBs for storing policies (instrumentation already existent for MIBs)
- each PEP is connected only to a single PDP

but also several advantages:
- it uses reliable communication between PDP and PEP (uses TCP)
- it has the possibility of expressing higher level policies (compared to SNMP)

1.2 SNMP approach

The SNMPConf working group carries out the work to map the Internet Standard Management Framework (SNMP) to the IETF policy-based management architecture.
Early outputs of this working group show that the SNMP framework can provide all the necessary capabilities required for configuration and monitoring, so this type of policy-based configuration management can provide an integrated approach to management.
Using policy-based configuration management with SNMP doesn't hinder from using traditional, instance specific configuration. The main advantage that we gain here is that the traditional configuration methods

can be used in combination with more powerful policy-based configuration operations.
Using objects from the same name space for configuration with policies as for monitoring, will help the error detection and recovery process.

This paper concentrates on the second approach, the policy-based configuration management with SNMP, presenting the work that has been done in the SNMP-Conf working group.
Chapters of this paper are organized as follows:
Chapter 2 gives definition of terms and the overall architecture of SNMP policy-based configuration management is described. Chapter 3 describes the objects of the policy management MIB module. Chapter 4 is dedicated to the DiffServ Policy MIB module, description of objects and tables, as well as relationship between the tables is given. Chapter 5 describes the necessary steps to perform a simple configuration. Relation of objects in Policy Management MIB, DiffServ Policy MIB and DiffServ MIB modules is outlined. Chapter 6 contains conclusions and further tasks.

2 SNMP POLICY-BASED MANAGEMENT

Before we can proceed with the presentation of the policy-based configuration management architecture it is necessary to have a clear understanding of the terms we are going to use. Many of these terms are defined by multiple working groups (Policy WG, RAP WG and SNMPConf WG) in their papers (unfortunately with slightly different meanings). There is a need to harmonize these definitions.

2.1 Definition of terms

Lots of discussions went on in the above mentioned working groups trying to define the term “policy”. Difficulties come from the diversity of definitions that already existed: from high-level administratively defined, technology independent policy (like: all departmental managers will have premium service access to web-browsing) to more specific policies (all workstations will have the latest version of operating system).
What makes the most confusion here is the level of abstraction that one intends. In an effort to clear this situation, the SNMPConf working group [3] defined the following notions:

- **technical domain** (or just domain) is a general area of technology such as QoS. DiffServ can be seen as materialization of the quality of service area.
- **mechanism** is an implementation of a particular domain. For example one can use a class based or priority queuing mechanism to support differentiated services expedited forwarding.
- **device** is a physical entity that has a mechanism implemented on it. We will hardly find a device independent mechanism, as the vendors tend to implement the mechanisms in a way that benefits from the particularities of a device.
- **instance**: all individual hardware or software elements involved in the fulfilment of policy rules. The main goal of policy-based management is to eliminate the necessity of configuring individual instances, improving in this way the efficiency of configuration.

Thus the level of abstraction of the policies leads to the following classification:

1. administratively defined policies (business level) - these are domain, mechanism, device and instance independent expressions of business requirements. They contain no specification about how the policy would be realized and no systems or network elements are mentioned to support the policy.
   An example might be: ‘*User Joe gets Premium Service for WEB browsing*’
   Service descriptions and Service Level Agreements often contain policies expressed at such a high level. Expression of these policies may span over several domains thus mapping to a domain dependent representation may be a difficult operation.
   We will assume domain dependency for now on.
2. mechanism, device and instance independent - is the expression of a policy translated into domain specific format. To express the previous example in a domain (e.g. DiffServ) specific fashion, one would say: ‘*If DestIPAddress == 123.45.67.89 then mark WEB traffic as Expedited Forwarding*’.
   The policy has not been yet assigned to a specific device or network element, nor has it been described how the Expedited Forwarding mechanism should be implemented.
3. device and instance independent, mechanism dependent - parameters to realize a specific mechanism are detailed here. The considered example expressed at this level would specify parameters to implement Class Based Queuing, as CBQ is a way to realize Expedited Forwarding. One should specify how packets arriving from the IPAddress 123.45.67.89 would be directed to the highest priority queue while traversing a traffic control bloc.
4. mechanism and device dependent, instance independent - what this level has in addition to the previous one is the specification of device specific parameters involved in realization of a specified mechanism. Since different vendors implement different features in their devices (and extend the standard MIB modules accordingly) there may exist several parameters that should be defined in a device specific fashion.

Note however that there is often an overlap between level 3 and 4 of abstraction as vendors try to implement mechanisms on their devices in their particular way.

5. mechanism, device and instance specific - this is the most specific expression of a policy. All parameters are expanded to all network elements that are involved in enforcement process of this policy.
   “*Ingress interfaces on router A will reject all connections from friday 8pm to monday 6am*“

For the scope of this paper we will take a policy as an administratively defined rule that states if certain object has certain characteristics then operations are to be applied to that object. These rules can be expressed as a pair of condition and an action [1]:

if (policyFilter) then (policyAction)

The following definitions are based on the draft-ietf-snmpconf-pm-01 document edited by the SNMPConf working group [3]:
policyFilter is a boolean expression used to determine if an element is member of the set of elements that the policy is going to be applied on;
policyAction: is a set of operations to be performed on the elements on which the policyFilter evaluates to true.
Policy rules (conditions and actions) have to be unambiguous and verifiable. There should be only one rule to be applied if a condition exists. For this purpose policy verifications are required.
Elements are objects that policies act on. Elements can be interfaces, processes, CPUs, queues etc. As these elements are MIB objects, they can be easily identified by their OID.
Roles are administratively defined strings that are associated with elements. Roles can have political, geographical, legal or architectural meaning and typically designate informations, characteristics which are not internally stored in the elements. Examples of roles can be: executives department or edge interface. Types of roles can be political, financial-legal, geographical, architectural etc.
Capabilities In order to identify elements that are appropriate for executing specific (type of) policy actions elements (devices) are "endowed" with capabilities. Thus certain policies can be applied only to elements that have the proper capabilities.
Time An important aspect of policies is the period of time they are valid. To achieve this functionality every policy is assigned a time variable. This time periods can be "regular" or can be defined "ad-hoc".

2.2 Architecture

RFC2571 [6] defines that a SNMP management system contains several agents (nodes which have access to management instrumentation, run a command responder and a notification originator), at least one manager (SNMP entity that contains a command generator and a notification receiver) and a management protocol to convey management information between the SNMP entities. Adding policy-enabled capabilities to this existing architecture seems straightforward.
A couple of MIB modules need to be added to the agent. These are the Policy Management MIB module and a domain specific MIB module (in this paper we will consider the DiffServ Policy MIB module as an example, as this is the only one domain specific module that is currently under development. It is expected that development of other domain dependent modules -e.g. IPSec - will start soon).

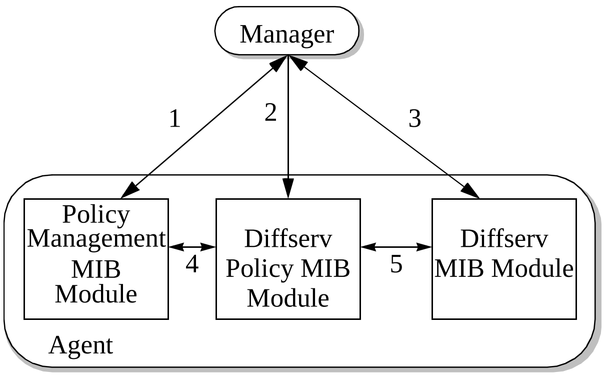


Figure 2. Policy enabled SNMP architecture

Relations 1 and 2 are new for the existing SNMP architecture, these represent the new policy-based management interactions.
Relation 3 is the traditional device level configuration management executed with SNMP.
Relations 4 and 5 are going to be explained in chapter 5 along with a configuration example.
We are going to focus on the first two interactions in this section.
From the policy-based configuration point of view the management station performs the following tasks:

- to distribute policies to all managed devices (agents)
- to monitor usage and status of execution of policies in the devices across the network

Policies are executed on the managed devices, thus characteristics of objects can easily be inspected and operations described by policies can be performed on them.

Policies translate to elements on a managed device as follows:

*foreach **element** for which (**policyFilter** is true)*
*execute **policyAction** on that element*

Policy filters and actions are defined with the policy expression language which is suitable for describing parenthesized logical and arithmetic expressions and several function calls for accessing (and setting) information on the local system (get and set values of objects in MIBs, functions on roles and capabilities). This language is a subset of ANSI C (refer to draft-ietf-snmpconf-pm-01 [3] for the complete description of the EBNF grammar of this expression language).
Definition of new functions, local variables, assignments, arrays, structures, pointers (except constant string pointers) and pre-processing have been excluded.

In order to apply a policy, the information from the policy module and from the domain specific module is combined to create a mechanism, device and instance specific information that can be used by a specific element to enforce it.

3 POLICY MANAGEMENT MIB MODULE

The Policy Management MIB module contains:

- information on the filters to apply for the selection of elements to apply the policy to;
- information on the existing roles and their association with specific instances;
- scheduling information of the policy (when to apply, for how long);
- information on how to apply mechanism specific parameters of the policy on the local device. These parameters are to be found in the mechanism specific MIB modules.

This MIB module contains objects that are SMIv2 compliant. Information contained in this module is organized into tables as follows:
The **pmPolicyTable** describes a pairing of policyFilter and policyAction. The filter and action are specified in the policy expression language. Calendar is a pointer to an entry in the schedTable of the Scheduling MIB. The policy is active only when specified by this entry. The Description column contains a human readable explanation of this policy. pmPolicyMatches is the current number of elements that policyFilter matches.
The **pmRoleESTable** and **pmRoleSETable** allow easy mapping of elements to roles and vice versa.
The **pmCapabilitiesTable** contains a description of the inherent capabilities of the system.

4 DIFFSERV POLICY MIB MODULE

The SNMP Configuration Working Group is chartered to elaborate an example MIB module that converts from the mechanism and device independent methods to mechanism, device and instance specific levels. The domain that has been chosen is Differentiated Services.
Differentiated Services provide the ability to handle different classes of traffic separately. Classes (or behaviour aggregates) are created by setting the DSCP value of an IP packet. At a differentiated services capable router different classes of traffic can receive different treatment known as per-hop-behaviour (PHB). In RFC2475 [7] per-hop-behaviour is defined as “*a description of the externally observable forwarding behaviour of a DS (Differentiated Services) node applied to a particular DS behaviour aggregate.*“
Currently two PHBs are being developed by the DiffServ working group: Expedited Forwarding and Assured Forwarding. The first specifies that an aggregate class will receive bandwidth that equals or exceeds a configured rate, the latter one specifies that each class is supposed to have a minimum allocated bandwidth.
PHBs however do not specify the mechanism to achieve specified performance. Different queuing mechanisms, classification, metering and marking procedures can be used to implement the desired per-hop-behaviour. To model these structures the DiffServ MIB module contains classifier, marker, meter, queue set and queue tables as well as action (drop action, mark action and count action) tables.
The DiffServ Policy MIB module acts as a template for the domain specific MIB module, objects of this module describe a possible configuration of DiffServ subsystem at a conceptually higher layer than the DiffServ MIB module, providing default values that fulfill different per-hop-behaviours.
The DiffServ Policy MIB module is designed to interoperate with the Policy Management MIB and the DiffServ MIB modules for an integrated architecture of both network-wide (policy-based) and device-specific network management.
This module is intended to be used on top of DiffServ MIB module (which operates at a device level) to create an interface towards the PolicyMIB module (which operates on a network-wide scale).
The DiffServ Policy MIB module acts like an interface between high-level "network wide" policy definitions (that affect configuration of the DiffServ subsystem) and instance specific information described in the DiffServ MIB module.

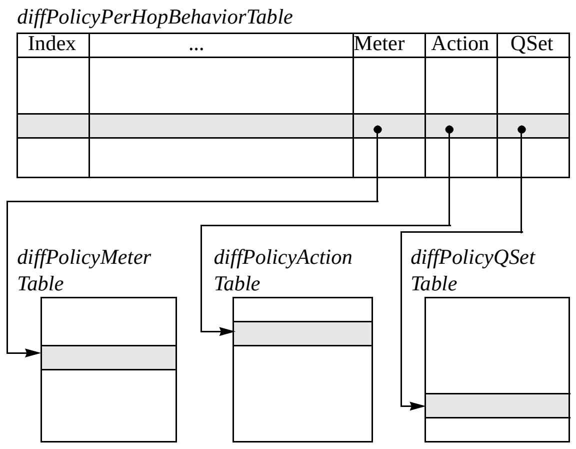


Figure 3. Relation of objects of diffPolicy MIB Module

Figure 3 shows the relations between the tables of the module. The *diffPolicyPerHopBehaviorTable* which enumerates the various per-hop-behaviours of which the system has default templates, has pointers to the other tables that contain default values for meters, actions and queue-sets that help implementing the desired per-hop-behaviour.
The meter, action and qset tables have similar structure as the tables in the DiffServ MIB module [8].

5 A SIMPLE CONFIGURATION EXAMPLE

Let's consider the following policy-rule: "User Joe gets premium service for web browsing as long as his traffic doesn't exceed a specified limit. The exceeding traffic gets best-effort service". There are several steps that are to be made in order to configure devices with this policy (or with policies in general [10]).

1. in the manager: users (humans or software) define the policy rules (filters and actions) and other essential information that is needed for proper application of policies on devices, such as roles or schedule information. Users also define device and mechanism specific information that is to be sent to the mechanism specific policy module.
   All this information is later sent to Policy Management MIB and to a domain specific MIB module.
   For the policy-rule example considered before policyFilter is going to be: DestIPAddress is 130.89.77.123 and sourceL4Port is 80 and rate is below 100kB/sec. PolicyAction is: assure premium service. Additional information to be provided is that expedited forwarding is used to achieve premium service.
2. The DiffServ Policy module which is capable of performing mechanism dependent configuration, registers with the Policy Management MIB module's pmCapabilitiesTable. It will indicate its capabilities of performing expedited forwarding.
3. The agent informs the manager about its capabilities. For this purpose the SNMP INFORM message should be used as it is an acknowledged one, thus the agent knows that a manager is aware of its capabilities. This is especially useful in case of multiple managers.
4. The manager then populates pmRoleESTable of the Policy Management MIB module and associates the roles with elements in the device. General examples of roles include: process owner is staff member, interface connects executive, user Joe has IP address 193.45.76.89. In our example a role association could be that premium service is going to be implemented by means of DiffServ EF with specific parameters, or that user Joe connects to the network through interface A.
5. The manager send policies to the managed devices. In simplest cases this implies setting of values in the pmPolicyTable of the Policy Management MIB module. Domain/mechanism independent expressions are loaded into the Policy Management MIB module and specific information is loaded into the domain specific MIB module.
6. The devices evaluate the policies (filters and actions) in order to determine which elements to apply the policies to (a look up in Roles table can be required here) and when to apply policies (pmPolicyCalendar field of pmPolicyTable).
7. The DiffServ policy module set the appropriate values either directly or via DiffServ MIB module. In the example, the DiffServ Policy module will populate the DiffServ module tables to implement EF for user Joe on the ingress interfaces specified by the roles in the pmRoleESTable.
8. Finally there is a need to monitor policy usage and status, and to verify policy results in order to apply refinements if necessary.

Figure 4. shows these configuration steps on the agent.

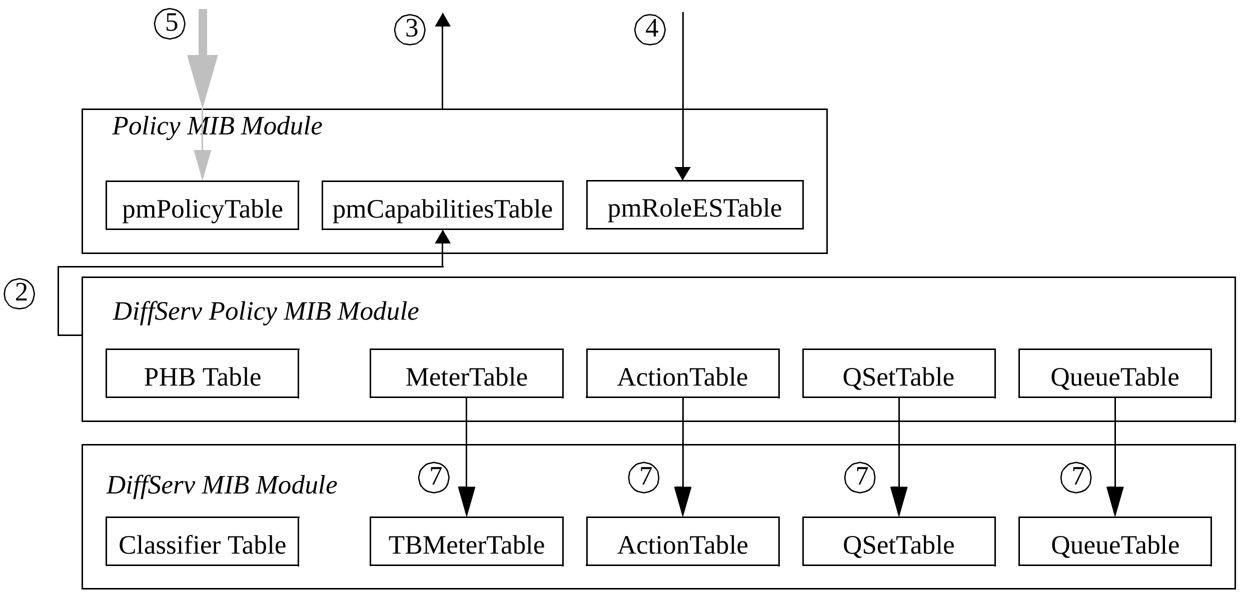


Figure 4. Relation between DiffServ Policy Module, Policy Management Module and DiffServ Module

6 CONCLUSIONS AND FURTHER WORK

Policy-based management means management of configuration of network elements based on a set of rules or business objectives.
The most simplistic way to define policy: “*a policy is a set of rules that describe the actions to be taken when specified conditions are met*” [9].
Configuration of networks based on policies implies several functions:

- definition of policies;
- storage of policies;
- enforcement of policies on network devices;
- verification of operations.

It was shown that SNMP provides an integrated framework both for policy-based configuration management and monitoring of networks.
The paper gave an overview of the state of the art in the policy-based configuration management with SNMP. The work is still in progress in the SNMPConf working group of the IETF so there are still quite a lot of things to be done.
If one parses the steps of the configuration example described in chapter 5 one can think of the following insufficiencies:
Registration in the capabilities table (described in step 2) needs to be worked on. This procedure tends to be the same as the sub-agent registration mechanism in an master and sub-agent technology.
There is still a need for an unequivocal naming framework for capabilities. This is being worked on in the IANA.

And finally there is a need for a implementation that practically proves the utility of the policy-based management with SNMP.

ACKNOWLEDGEMENTS

The author would like to thank to the members of the SNMPConfig working group for providing the material elaborated in this paper, and to the members of TSS Management Group for the useful comments and remarks to the early drafts.

REFERENCES

[1] Mark L. Stevens and Walter J. Weiss, “*Policy-based Management for IP Networks*”, Bell Labs Technical Journal, Network Management, Vol. 4 No. 4, October-December 1999.

[2] R. Rajan et al., “*A Policy Framework for Integrated and Differentiated Services in the Internet*”, IEEE Network, Vol. 13 No. 5, September/October 1999

[3] snmpconf WG homepage at www.ietf.org/html.charters/snmpconf-charter.html

[4] RAP WG homepage at www.ietf.org/html.charters/rap-charter.html

[5] D. Durham et al., “*The COPS (Common Open Policy Service) Protocol*”, Internet RFC2748, January 200

[6] D. Harrington, R. Presuhn, B. Wijnen, ”*An Architecture for Describing SNMP Management Frameworks*”, Internet RFC2571, May 1999

[7] S. Blake et al., “*An Architecture for Differentiated Services*”, Internet RFC2475, December 1998

[8] Fred Baker et al., “*Management Information Base for the Differentiated Services Architecture*”, Internet Draft, draft-ietf-diffserv-mib-02, March 2000

[9] “*Introduction to QoS Policies*” - White Paper at www.qosforum.com/white-papers/qospol_v11.pdf

[10] Jon Saperia, “*Policy-Based Configuration Management with SNMP*”, to be published in *The Simple Times* (www.simple-times.org)