Journal of Network and Systems Management, Vol. 10, No. 1, March 2002 (© 2002)

Extending COPS-PR with Meta-Policies for Scalable Management of IP Networks

Raouf Boutaba[1,3] and Andreas Polyrakis[2]

During the past years, IP networks have grown considerably in size and complexity: the number and the variety of the connected devices have increased, new applications have emerged and Quality-of-Service is increasingly demanded. In such networks, traditional management techniques seem to suffer from significant scalability and efficiency limitations. Policy-Based Networking (PBN) has emerged as a promising paradigm for IP networks operation and management. In PBN, policy servers enforce network policies by sending the appropriate configuration data to the managed devices. IETF is currently developing COPS (Common Open Policy Service) and its extension for policy provisioning, COPS-PR, as the protocols to implement PBN. COPS-PR, although initially biased towards DiffServ, has received significant attention and seems efficient for several other management areas, such as accounting and IP filtering. However, in COPS-PR, the rigidity of the policy-enforcing mechanisms at the managed devices restricts the intelligence that can be pushed into them. This work attempts to raise these limitations by extending the COPS-PR protocol with meta-policies. Meta-policies are rules that can be stored and processed by the devices, independent of their semantics. They allow intelligence to be pushed towards the managed device, making in this way the scheme more efficient, scalable, distributed and robust.

KEY WORDS: Policy-based networking; policy interpretation; self-managed devices.

1. INTRODUCTION

Traditional Management techniques are based on low-level configuration of each network device individually. Management tools that attempt to automate this process do exist; however the diversity of these tools and the lack of interoperability among them reduce their efficiency significantly. Moreover, during the past years, computer networks have grown significantly in terms of size, complexity

[1] University of Waterloo, Department of Computer Science, 200 University Avenue West, Waterloo, Ontario N2L 3G1, Canada. E-mail: *rboutaba@bbcr.uwaterloo.ca*

[2] University of Toronto, Department of Computer Science. E-mail: *apolyr@cs.toronto.edu*

[3] To whom correspondence should be addressed.

and heterogeneity, new applications have emerged, and Quality-of-Service (QoS) is increasingly demanded. These facts stress the need to further raise the level of abstraction and hide the network details from the system administrators.

A promising solution to this problem is Policy-Based Networking (PBN) [1, 2]. PBN is based on control/management policies, i.e., rules that determine the network behavior. The administrator edits the policies in a management tool that performs syntax, semantics and basic conflict checking. These policies are then distributed, either directly or through the use of a directory service, to special policy servers called Policy Decision Points (PDPs) [3]. The PDPs process these policies, along with other data such as network state information, and make policy decisions regarding the policies that should be enforced. These policies are sent as configuration data to the appropriate Policy Enforcement Points (PEPs) [3], which (typically) reside on the managed devices and are responsible for installing and enforcing these data into them. PBN is illustrated in Fig. 1.

It is important to highlight that PDPs do not simply distribute policies to the PEPs. The role of a PDP is (i) to combine the high-level policies with the network

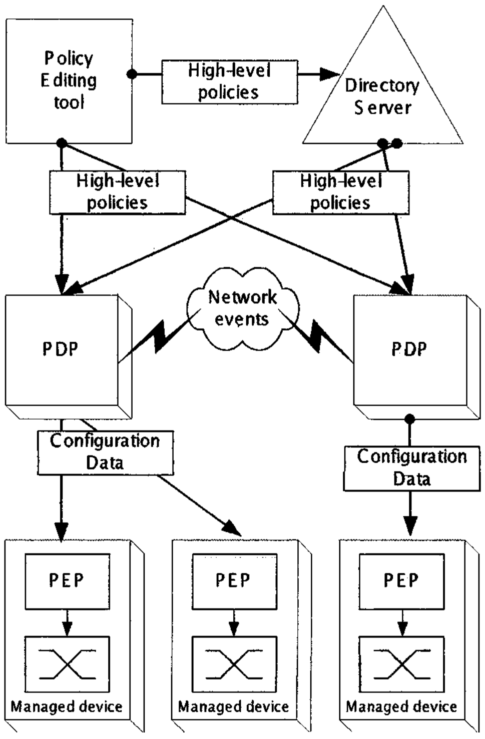


Fig. 1. Policy-based networking.

state in order to determine the desired behavior of every device at that specific moment; and (ii) to generate the appropriate low-level configuration data for each device (in a supported format and according to its capabilities/limitations) that enforces this behavior. This implies that if the network state or policies change, the PDP may need to readjust the behavior of the devices by sending updated configuration data.

The Common Open Policy Service (COPS) [4] protocol is an attempt of the IETF to standardize the communication between PDPs and PEPs. COPS is being developed by the Resource Allocation Protocol (RAP) [5] working group. Although RAP purpose is to “establish a scalable policy control model for RSVP” [5], COPS soon received significant attention from other research groups, within or outside IETF. RAP has also developed COPS for Policy Provisioning (COPS-PR) [6] as an extension of COPS. COPS-PR was initially biased towards DiffServ policy provisioning [7]. However, it appears to be suitable for several other management areas (accounting [8], IP filtering [6, 9], security [10], etc. [11]).

However, in COPS-PR, all the intelligence is concentrated at the PDP level. The decisions that a PEP can make are very limited, due to the rigid structure used to store and process policies. This poses some shortcomings in the efficiency, scalability, distribution and robustness of the protocol. We attempt to raise these limitations by using meta-policies, rules generated by the PDP that allow the PEPs to make policing decisions with less or no guidance from the PDP.

The structure of this discussion is as follows: Section 2 presents COPS-PR, demonstrates how it works, with a simple example, and discusses its shortcomings. Section 3 introduces the concept of meta-policies and demonstrates how these are used and how they increase the efficiency, scalability and robustness of the protocol. Section 4 presents the necessary extensions in the COPS-PR protocol to carry meta-policies, and discusses how PDPs and PEPs are affected by this extension. In Section 5 some interesting thoughts and issues are presented, and Section 6 concludes the paper.

2. COPS-PR

In COPS-PR, the PEPs reside on the managed devices and the PDPs on special policy servers. Each PEP may contain one or more clients, responsible for different, nonoverlapping policy areas (security, QoS, admission control, accounting, etc). The clients connect to the appropriate PDP (different PDPs may control different clients in a single PEP), report their capabilities and limitations, and request the initial policies to be downloaded into them. The PDP processes the request of each client and, according to the global policies and network state, generates and downloads the appropriate configuration data. If the network state or policies change afterwards, the PDP may decide to update these configuration data, in order to keep the behavior of the managed device consistent.

Although a PEP may contain multiple clients, for simplicity reasons in the rest of this discussion we examine PEPs with a single client. Hence, when we refer to the PEP, we may actually mean its client. However, this will be obvious from the context and should not confuse the reader.

2.1. The Policy Information Base

Each client stores the received configuration data into a special database, called Policy Information Base (PIB) [6, 12]. The PIB is a structure similar to a MIB; it can be described as a conceptual tree namespace, where the branches represent structures of data, or Provisioning Classes (PRCs), and the leaves represent the instances of these classes and they are called Provisioning Instances (PRIs). PIBs are defined by COPS-PR only as abstract structures; the details of each PIB (PRCs and their semantics) are specified in separate standard documents (such as internet-drafts or vendor private documents). Different PIBs are defined in order to cover the various management areas (DiffServ, accounting, security etc). PIBs are defined in a high abstraction level; in this way they hide the details of the underlying hardware and provide to the PDP a unified way to control the behavior of the devices, with regard to a specific management area, across the entire network.

PRIs are identified within the PIB through a PRI identifier (PRID) (see Fig. 2). The PDP can install or update PRIs by sending an install decision specifying the appropriate PRIDs and their values, or remove PRIs with a remove decision containing the PRIDs of the PRIs to be removed. Policies are formed as a set of PRIs in the PIB; by adding or removing PRIs, the PDP can implement the desired policies, which will be enforced on the device.

It is important to highlight that the policies that the PIBs can implement are predefined (in the documents that define those PIBs). In order to control a device, the PDP has to map the high-level network policies and the network state into policies that can be implemented in its PIB.

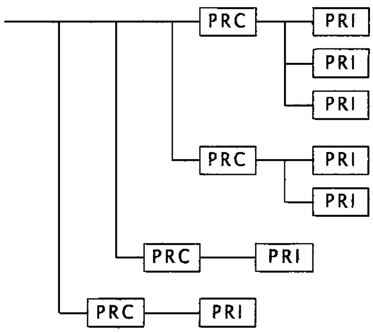


Fig. 2. PIB structure.

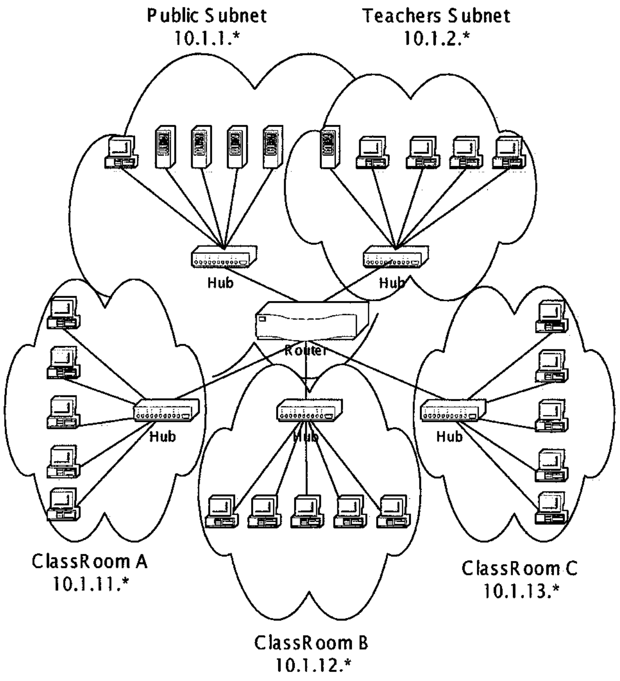


Fig. 3. The topology of the school example network.

2.2. Example

We shall use a small filtering PIB in order to demonstrate how COPS-PR works. The network of our example is the network of a modern school (Fig. 3), with the following topology:

- A public subnetwork (10.1.1.*) with public servers (file servers, ftp, etc.) and some administrative servers (authorization server, PDP, etc)
- A teachers subnetwork (10.1.2.*) which is accessed only by the teachers.
- 3 classroom subnetworks (10.1.11.*-10.1.13.*)
- A router that connects these subnetworks.

Suppose that the following high-abstraction access rules have been set:

- Classrooms workstations have access to the public subnet during class time: Working days, 9 am to 5 pm, excluding breaks (last 10 min. of each hour) and lunch time (1 pm to 2 pm).
- A teacher can override the following rule and give access to a classroom subnet beyond the class time.

- A teacher may log to any workstation in a classroom and have access to the teaching subnet from there.
- The access of students is restricted within their classroom subnet and the public subnet.
- Traffic from the multimedia server (10.1.1.5) is denied to classrooms during congestion.
- Teacher subnet can always reach the public domain.

Assume that at a specific moment the state of the network is as follows:

- Time: 11:55
- A teacher is logged on a workstation with IP 10.1.11.5
- Access to/from classroom A has been extended during the break period
- The router is congested.

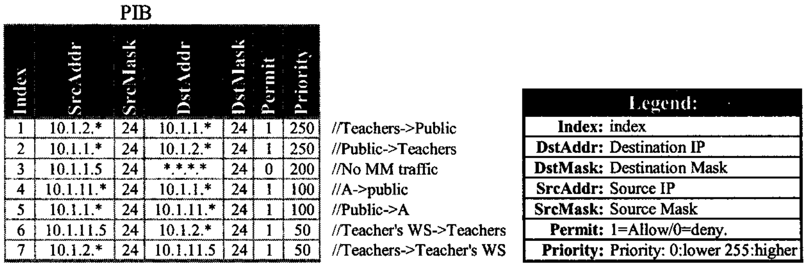
Suppose that the (PEP of the) router of this network supports a PIB with a single PRC. PRIs of this PIB describe source/destination criteria that allow/deny access to IP traffic within the network (Fig. 4). Note that a single PRI in this PIB is a stand-alone policy of the form:

If ((Source matches Srcaddr/Srcmask) and (Destination matches Destaddr/Destmask))
then allow/deny
(with policy priority: P)

Traffic that does not match any of these criteria is, by default, denied.

In order to achieve the desired behavior, the PDP has to modify the PIB to become as in Fig. 4. The priorities of the PRIs are chosen by the PDP, so that the desired behavior is achieved.

The previous imply the existence of mechanisms that allow the PDP to be informed of all the related network events. For example, a script that runs when a teacher logs on or out can inform the PDP of this event. Similarly, the PDP can



PIB

Index	SrcAddr	SrcMask	DstAddr	DstMask	Permit	Priority	
1	10.1.2.*	24	10.1.1.*	24	1	250	//Teachers->Public
2	10.1.1.*	24	10.1.2.*	24	1	250	//Public->Teachers
3	10.1.1.5	24	*.*.*.*	24	0	200	//No MM traffic
4	10.1.11.*	24	10.1.1.*	24	1	100	//A->public
5	10.1.1.*	24	10.1.11.*	24	1	100	//Public->A
6	10.1.11.5	24	10.1.2.*	24	1	50	//Teacher's WS->Teachers
7	10.1.2.*	24	10.1.11.5	24	1	50	//Teachers->Teacher's WS

Legend:	
Index:	index
DstAddr:	Destination IP
DstMask:	Destination Mask
SrcAddr:	Source IP
SrcMask:	Source Mask
Permit:	1=Allow/0=deny.
Priority:	Priority: 0:lower 255:higher

Fig. 4. A snapshot of the PIB of the router.

use its clock, a clock network service, or some external signal to evaluate the event "class time."

2.3. COPS-PR Shortcomings

The previous example demonstrates how the PDP transforms the network policies and binds them with the network state, in order to generate the appropriate configuration data that control the devices. However, the intelligence of this scheme is concentrated at the PDP level. The PEP itself simply executes commands; it can take no decisions for any event (apart from these decisions implied by its PIB), even if this event has occurred before. For instance, during breaks, access from the classroom subnets to the public subnet must be denied. In order to achieve this, the whole set of PRIs that implement this policy must be sent to the PEP; just being notified that the break has started or ended is not sufficient. The same holds when a teacher logs off from a workstation and logs on to another: the PEP needs exact directions from the PDP to treat this event, although it was directed in the past to treat a similar event. Each time the state of the network changes, the PDP needs to install and uninstall PRIs in the devices that are affected by this event, even if the same or similar actions were taken in the past.

A second limitation lies on the fact that the PEP has no authority (or ability) to perform any monitoring by itself, even when this would be obviously more efficient. In our example, for instance, the PDP has to query the router in order to detect congestion; then, depending on the results, the PDP has to decide what actions are to be taken by the PEP. The protocol would have been more efficient, if the involvement of the PDP in the monitoring process had been avoided.

Finally, in COPS-PR, the PEPs rely on the existence of the PDP to operate properly. During a PDP absence, the PEP keeps caching older decisions (stored in its PIB); however this is not always desired: For instance, in the school example, a PDP failure during a break would prevent any classroom subnets from accessing the public subnet after the end of the break. Moreover, a PDP failure while a teacher was logged on at a workstation would result in leaving access privileges to this workstation towards the teachers' subnet (where confidential/critical files are kept), after the teacher had logged out.

We attempt to raise these limitations by extending the existing COPS-PR protocol with meta-policies.

3. THE CONCEPT OF META-POLICIES

3.1. Definition

We define a meta-policy as a rule of the form:

if (condition) then $\{$actions$\}$

where "*condition*" is a logical expression and "*actions*" is a set of pre-generated commands, similar to COPS-PR, that encode one or more policies. Since the actions encode a specific policy, this rule is a rule on how policies are enforced; this is why it is called a "meta-policy."

Meta-policies are generated by the PDP and sent to the PEP. The PEP evaluates the conditions of each meta-policy, and when it evaluate true, it enforces the actions. Both the condition and the actions may contain parameters (such as "network congested," "time" or "TeacherIP"); the values for these parameters are either sent by the PDP or evaluated by the PEP, according to directions provided by the PDP. The concept of meta-policies is demonstrated in the following example.

3.2. Example

Let us examine how the school example would have been affected by the use of meta-policies. For simplicity reasons, we shall disregard classrooms B and C.

First of all, the policy "*Teacher subnet can always reach the public domain*" must always be enforced. Hence, the PDP directly enforces this policy by installing the two PRIs (1 and 2) into the PIB (Fig. 5a).

Besides, the PDP downloads to the PEP the following meta-policies:

```
• if ((ClassTime) or (ClassATimeExtended)) then {
  if (source matches 10.1.11.*/24) and (destination matches 10.1.1.*/24) then
     allow
  if (source matches 10.1.1.*/24) and (destination matches 10.1.11.*/24) then
     allow
  }
• if (TeacherLogged) then {
  if (source matches TeachIP/24) and (destination matches 10.1.2.*/24) then
     allow
  if (source matches 10.1.2.*/24) and (destination matches TeachIP/24) then
     allow
  }
• if ((if1Util>80%) or (if2Util>80%) or (if3Util>80%) or (if4Util>80%)
  or if5Uril>80%) then {
  if (source matches 10.1.1.5/24 ) then deny
  }
```

The policies within the brackets (actions of the meta-policies) are presented in a high-level language here, but they are actually sent as install decisions (similar to COPS-PR) in the PIB of the router (Fig. 5c). The priorities of the meta-policies are chosen by the PDP. Note that the actions of these meta-policies are not conflicting (they can co-exist in the PIB), this is why they have the same priority.

Index	SrcAddr	SrcMask	DstAddr	DstMask	Permit	Priority
1	10.1.2.*	24	10.1.1.*	24	1	250
2	10.1.1.*	24	10.1.2.*	24	1	250

(a) PIB initial data (PRIs)

parameter: evaluation method
ClassTime: value sent by the PDP
Ext_Time_SubA: value sent by the PDP
TeacherLogged: value sent by the PDP
TeacherIP: value sent by the PDP
if1Util: MIB variable x.y.z.w.v1
if2Util: MIB variable x.y.z.w.v2
if3Util: MIB variable x.y.z.w.v3
if4Util: MIB variable x.y.z.w.v4
if5Util: MIB variable x.y.z.w.v5

x.y.z.w.v1-5: the MIB OID of the variable that shows the link utilization

(b) Parameters

id	Condition	Actions	Priority
1	(ClassTime) or (ClassATimeExtended)	install (4,10.1.11.*,24,10.1.1.*,24,1,100) install (5,10.1.1.*,24,10.1.11.*,24,1,100)	100
2	TeacherLogged	install (6,TeachIP,24,10.1.2.*,24,1,50) install (7,10.1.2.*,24,TeachIP,24,1,50)	100
3	(if1Util>80%) or (if2Util>80%) or (if3Util>80%) or (if4Util>80%) or (if5Util>80%)	install (3,10.1.1.5,24,*.*.*.*,24,0,200)	100

(c) Meta-policies

Fig. 5. The use of meta-policies.

Since these meta-policies contain parameters, the PDP has to inform the PEP of the evaluation method for these parameters. In our example, the PDP sends the values of the parameters ClassTime, ClassATimeExtended, TeacherLogged, TeachIP and directs the PEP that the parameters if1Util-if5Util get their values from the MIB variables of the router that denote the utilization of its interfaces. (Fig. 5b).

The PEP monitors the parameters, and when their value changes it re-evaluates the affected conditions. If a condition becomes true, the PEP executes the described actions (in this example, it installs some PRIs into the PIB). In this way, the PIB contains always the appropriate PRIs that implement the desired behavior. For the time moment examined in Section 2.2, the PDP would have sent the following values to the PEP: ClassTime=false, ClassATimeExtended=true, TeacherLogged=true, TeachIP=10.1.11.5. Besides, the PEP would have detected congestion (through the MIB of the router). With these values, the meta-policies would have modified the PIB to be exactly the same as in the previous (Fig. 4).

Due to the meta-policies, the PEP no longer needs to be informed of *how* to react when the network state changes; it only needs to be informed for this state (values for some parameters, sent by the PDP). Also, the PEP can now take some decisions (e.g., when a subnet is congested) without the involvement of the PDP. Besides, even for events that are not local to the PEP, the latter could be directed somehow (e.g., by downloading and executing some scripts, or through mobile agents) to contact a third entity (time service, authorization server, etc.) and directly be informed of the network events.

4. PROTOCOL EXTENSIONS

The previous section introduced the concept of meta-policies and demonstrated how these can be used in order to extend the functionality of the PEP. This section describes how the COPS-PR protocol can be enhanced in order to communicate meta-policies. The extended protocol is described in detail in [13]; we shall only present an overview of the extension here.

4.1. The Meta-Policy Object

In COPS-PR, the PDP and PEP communicate by exchanging objects that convey policy information. We extend the existing COPS-PR objects with a new one, called Meta-Policy Object (MPO), which is used to convey meta-policy related information [13]. This object has similar format with the existing COPS-PR objects and it encapsulates other (new) objects that encode meta-policies. The description of the objects is out of the scope of this work, but the reader may refer to [13] for details. The meta-policy related data that these objects convey are described in the next paragraph. We would like to highlight that the proposed extension fully conforms to COPS, and is a superset of COPS-PR. This feature is very important, since it allows interoperability with standard COPS-PR PEPs and PDPs.

4.2. Meta-Policy Related Data

Meta-policy related data are divided into the meta-policies themselves, and the parameters that are used by these meta-policies. All these data are encapsulated within Meta-Policy objects, whenever exchanged between PDPs and PEPs.

Meta-Policies: Each meta-policy consists of a condition and a set of actions. The condition represents a logical expression, such as “($C>80\%$) and ($D=$true).” The actions encode a specific policy in the form of configuration data, similar to COPS-PR (i.e., commands that install PRID-value pairs or remove PRIDs).

Meta-Policies are generated by the PDP and consumed by the PEP. The key idea in meta-policies is that the PEP can store and process these meta-policies without knowing their exact semantics. The condition is treated as a logical expression (in the previous example, the PEP does not need to know the semantics of C and D, their values are sufficient). Similarly, the actions are pre-generated by the

PDP and the PEP does not need to know what policies they implement, in order to enforce them. In this way, the PEP can process any meta-policy, independently of its complexity and its meaning.

Each meta-policy must be associated with a meta-policy identifier, so that the PDP will be able to update or remove this meta-policy later. Also, each meta-policy must be assigned with a priority. This priority is used by the PEP in order to resolve a conflict between two meta-policies that need to be activated at the same time, but their actions are conflicting.

Meta-policies may use parameters in their conditions or actions. The parameters that a meta-policy uses must be installed by the PDP prior to installing the meta-policy.

Parameters: The parameters are used in meta-policy conditions in order to determine when a meta-policy must be activated. Moreover, they are used by meta-policy actions in order to dynamically bind the network state within policies. For example, the meta-policy in Fig. 5c "*if (TeacherLogged) then {install (6, TeachIP, 24, 10.1.2.*, 24, 1, 50), install (7, 10.1.2.*, 24, TeachIP, 24, 1, 50)}*" contains two parameters: TeacherLogged and TeachIP.

Each parameter is assigned with an identifier (name), which is used by the meta-policies that use this parameter. Also, these identifiers are useful in removing or updating parameters.

When installing a parameter, the PDP must also specify an evaluation method for this parameter. For instance, the PEP can be directed to get a value for a parameter from the MIB of the device. Or, the PDP could provide the value for this parameter. However, other methods are also possible, depending on the capabilities of the device, such as to download and execute a script, use mobile agents, or get the desired information from some server or service (e.g., clock service).

4.3. Message Extensions

COPS-PR objects are exchanged in three types of messages: Request and Report messages, send by the PEP to the PDP, and Decision messages, send by the PDP to the PEP. We extend these messages to be able to carry meta-policy related information, through Meta-Policy Objects. The reader can also refer to [13, 14], for more details.

Requests: In COPS-PR, when a client of a PEP connects to the PDP, it issues a request message, asking for configuration data. Through this message, the capabilities and limitations of the client are reported. In the proposed extension, the clients must also report meta-policy related capabilities and limitations. These include information such as memory available for meta-policies, maximum number of meta-policies and parameters, supported parameter evaluation methods and types of logical expressions (conditions) that can be processed by the PEP. All this information is encapsulated within Meta-Policy Objects.

Decisions: COPS-PR decisions are used to install or remove PRIs. A single message may carry several install/remove decisions. By using Meta-Policy Objects, these decisions can be extended to install or remove meta-policies and parameters.

Reports: When a PEP receives a decision, it must issue a report, reporting the success or failure of the decision. In case of a failure, the report message specifies the source of the error (e.g., the PRID which caused the error). If an error is related to a meta-policy related decision, the Meta-Policy Object is used in order to point out the erroneous meta-policy or parameter.

4.4. PDP Extensions

In COPS-PR, the PDP generates decisions according to rules, which determine when a policy must be enforced and how conflicts are resolved at the PDP level. Similar rules can be used by the extended PDPs in order to generate the conditions and the priorities of the meta-policies.

The extended PDP must also decide which parameters will be evaluated by the PEP itself. The parameters that originate from the MIB of the device would better be evaluated locally. However, the solution to this problem is not always that obvious. Hence, the extended PDP needs to be augmented so that it can take such decisions efficiently. Rules, set by the administrator for this purpose, might be necessary.

4.5. PEP Extensions

First of all, the PEP must be able to store and process the meta-policy related information. All meta-policies, independently of their content, are composed of a condition and a set of actions, they are associated with a unique identifier and they are assigned with a priority. Consequently, all kinds of meta-policies can be stored in a unified way. The same applies to parameters, as well, since they are all composed of an identifier, an evaluation method, and their values.

As far as processing is concerned, the PEP has to evaluate the parameters and the conditions, and enforce the appropriate actions. The evaluation of the parameters is done according to the directions of the PDP (from MIB, value send by the PDP, other method that the PEP has declared that it supports in the request message). When a parameter value changes, the affected conditions (or actions) need to be recalculated, and the PEP must decide whether the meta-policy needs to be enforced. The conditions of the meta-policies are encoded using a grammar and operators that the PEP has also declared that it supports in the request message (we use XML for this purpose; however other solutions may exist). The actions of a meta-policy are pre-processed commands, similar to COPS-PR, that install or remove PRIs from the PIB, hence, the PEP can easily enforce them into the PIB. However, the PEP has to check if the actions of a meta-policy are conflicting with PRIs already installed by another meta-policy (the case that it conflicts with PRIs directly installed should be prevented by the PDP). In this case, priorities are used to resolve the conflict. Finally, the PEP must uninstall the actions of meta-policies when their conditions non longer evaluate true, or if the actions of a conflicting, higher-priority meta-policy need to be installed.

5. DISCUSSION OF RELATED ISSUES

5.1. Security Considerations

The described extension fully conforms to COPS. Hence, COPS safety and security mechanisms apply to it. Since the proposed protocol extends COPS-PR, it inherits all its security issues. However, the extension does not seem to raise any important new ones: the enforcement functionality on the device remains the same; besides, illegally installing, removing or updating a meta-policy has similar effect as doing as with a set of regular policies.

5.2. Conflict Detection and Resolution

In COPS-PR, the PDP has to ensure that the PEP never implements two conflicting policies. In case two such policies need to be applied, the PDP should detect the conflict and resolve it according to some rules. The exactly same rules are used by the PDP in our extension in order to detect possible conflicts between meta-policies and in order to provide priorities that will allow the PEP to resolve a possible conflict. The PEP, on the other hand, must be equipped with rules that describe unacceptable combinations of PRIs in the PIB. In this way, two meta-policies that attempt to install conflicting PRIs will trigger the conflict resolution mechanism (priorities), which will resolve the conflict.

An interesting remark relates to conflict detection at the PEP level. With COPS-PR, exactly one PDP is responsible for each client of a PEP, and makes sure that no conflicting policies will be installed at this client. However, a PEP may have more than one client, and these clients can be controlled by more than one PDP. In this case, nothing prevents the different clients from implementing conflicting policies. For instance, a filtering client may deny access to a flow with a specific IP address, while a QoS client may give the same flow high priority. In our work, the idea of increased functionality at the PEP level, by means of meta-policies, allows us to imagine PEPs that can understand the semantics of the policies that they implement, detect conflicts and request from the PDPs to take the appropriate actions.

5.3. Active Networks

Finally, we would like to comment on similarities of our work to Active Networks approaches. Several of the Active Networks Management techniques that exist today attempt to push some of the intelligence towards the network devices [15]. Our work was inspired from such approaches, although the protocol extension does not require the PEPs to run on active devices. However, to the best of our knowledge, there is no work conducted yet that attempts to combine Active Networks and Policy-Based Networking. Such a combination would provide a linkage between the low level intelligence and the high level operational policies and it would make the existence of "smarter" devices and the concept of "plug-and-play" networks seem more feasible and realistic.

6. CONCLUSIONS

This discussion introduced the COPS-PR protocol and identified some of its shortcomings. We introduced the concept of meta-policies, and demonstrated how they can be used to overcome the identified shortcomings. Finally, we presented the necessary extensions of the COPS-PR protocol in order to support meta-policies.

The use of meta-policies allows further distribution of the policies towards the PEPs, which is beneficial in several ways. First, the efficiency of the protocol is increased. The PDP initially downloads meta-policies to the PEP. For a downloaded meta-policy, the PDP only needs to inform the PEP of the related network events, instead of sending every time the actions that must be enforced. This reduces the size of the exchanged messages across the network. Also, the PEP can be directed to monitor some network parameters by itself; if any of these parameters are local to the PEP, monitoring traffic is also reduced. Second, the PDP is relieved from significant processing. Indeed, in COPS-PR, the PDP has to monitor the network and produce commands for all the affected clients in real-time. In the extension, the PDP generates the meta-policies for each device when this device connects. Afterwards, the PDP only monitors the network, and informs each client for the events that this client is interested in. Also, the PEPs can be programmed to perform some monitoring, relieving the PDP from it. Finally, the scheme becomes more fault-tolerant and robust. The exchanged messages are smaller, hence less probable to get lost in a congested network. PDP malfunctions, caused by overload, are also prevented. Also, the PEP has the ability to make some decisions independently of the PDP, which ensures its correct operation even when the PDP is unreachable.

ACKNOWLEDGMENTS

This research work was supported by research grants from Nortel Networks and the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

1. Susan J. Shepard, Policy-based networks: hype and hope, *IT Professional*, Vol. 2, No. 1, pp. 12–16, January-February 2000.
2. Introduction to Policy-based Networking and Quality-of-Service, *IPHighway*, White paper, January 2000.
3. A. Westerinen, J. Schnizlein, J. Strassner, Mark Scherling, Bob Quinn, Jay Perry, Shai Herzog, An-Ni Huynh, Mark Carlson, and Steve Waldbusser, Terminology, IETF, Internet-Draft, *draft-ietf-policy-terminology-02.txt*, November 2000. (*http://www.ietf.org/internet-drafts/draft-ietf-policy-terminology-0.2.txt*)
4. D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry, The COPS (Common Open Policy Service) Protocol, IETF, RFC 2748, January 2000. (*http://www.ietf.org/rfc/rfc2748.txt*)
5. Resource Allocation Protocol (rap), *http://www.ietf.org/html.charters/rap-charter.html*
6. K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, and A. Smith, COPS Usage for Policy Provisioning, IETF, RFC 3084, March 2001. (*http://www.ietf.org/rfc/rfc3084.txt*)
7. M. Fine, K. McCloghrie, J. Seligson, K. Chan, S. Hahn, A. Smith, and F. Reichmeyer, Differentiated services quality-of-service policy information base, IETF, Internet draft, *draft-ietf-diffserv-pib-03.txt*, March 2001. (*http://www.ietf.org/internet-drafts/draft-ietf-diffserv-pib-03.txt*)
8. D. Rawlins, A. Kulkarni, K. Ho Chan, and D. Dutt, Framework of COPS-PR policy information base for accounting usage, IETF, Internet draft, *draft-ietf-rap-acct-fr-pib-01.txt*, July 2000. (*http://www.ietf.org/internet-drafts/draft-ietf-rap-acct-fr-pib-01.txt*)
9. J. Ottensmeyer, M. Bokaemper, and K. Roeber, A Filtering Policy Information Base (PIB) for edge router filtering services and provisioning via COPS-PR, IETF, Internet draft, *draft-otty-cops-pr-filter-pib-00.txt*, November 2000. (*http://www.ietf.org/internet-drafts/draft-otty-cops-pr-filter-pib-00.txt*)
10. M. Li, D. Arneson, A. Doria, J. Jason, and C. Wang, IPSec policy information base, IETF, Internet draft, *draft-ietf-ipsp-ipsecpib-02.txt*, March 2001. (*http://www.ietf.org/internet-drafts/draft-ietf-ipsp-ipsecpib-02.txt*)
11. Harsha Hegde and Brad Stone, Load balancing policy information base, IETF, Internet draft, *draft-hegde-load-balancing-pib-00.txt*, February 2001. (*http://www.ietf.org/internet-drafts/draft-hegde-load-balancing-pib-00.txt*)
12. M. Fine, K. McCloghrie, J. Seligson, K. Chan, S. Hahn, R. Sahita, A. Smith, and F. Reichmeyer, Framework policy information base, IETF, Internet draft, *draft-ietf-rap-frameworkpib-04.txt*, November 2000. (*http://www.ietf.org/internet-drafts/draft-ietf-rap-frameworkpib-04.txt*)
13. R. Boutaba and Andreas Polyrakis, COPS-PR with meta-policy support, IETF, Internet draft, *draft-boutaba-copsprmp-00.txt*, April 2001. (*http://www.ietf.org/internet-drafts/draft-boutaba-copsprmp-00.txt*)
14. R. Boutaba and Andreas Polyrakis, Towards extensible policy enforcement points, *IEEE Workshop on Policies for Distributed Systems and Networks*, Bristol, United Kingdom, pp. 247–261, January 2001
15. R. Boutaba and A. Polyrakis, Projecting advanced enterprise network and service management to active networks, *IEEE Network Magazine*, Vol. 16, No. 1, 2002.

Raouf Boutaba has been a Professor in the Department of Computer Science of the University of Waterloo since 1999. He conducts research in integrated network and systems management, wired and wireless multimedia networks, and quality-of-service control in the Internet. He is the recipient of the Premier's Research Excellence Award in 2000.

Andreas Polyrakis finished his undergraduate studies at the National Technical University of Athens, Greece, and recently received his M.Sc. degree from the University of Toronto, Canada. His main area of research is Policy-Based Networks. He is also interested in other areas of Network Management such as Directory-Enabled Networking, Active Networks, and QoS provisioning in IP networks.