POLICY PROVISIONING PERFORMANCE EVALUATION USING COPS-PR IN A POLICY BASED NETWORK

A. Corrente, M. De Bernardi, R. Rinaldi
Cefriel, Getronics

Abstract: This paper presents a study carried out to evaluate policy provisioning performance, by COPS-PR protocol, in a policy based environment. First of all the prototypes developed are presented. These simulate the component behavior of the two lower layers of a policy based management system. Then the various measurements, done on the prototypes, are shown. All measures are carried out for several scenarios, described through the use appropriate configuration policies, in an increasing complexity order. It is demonstrated that the architecture of COPS-PR is scalable, though the PDP can become a bottleneck.

Key words: *Configuration Management and Policy-Driven Management, Policy Provisioning, COPS-PR, PRC*

1. INTRODUCTION

Configuring networks is becoming an increasingly difficulty task. The problem exacerbates as networks grow in dimensions (number and type of devices. To face this growth, administrators discovered that they have new requirements for the configuration of their networks. One of these new requirements is "configuration provisioning" to a specific set of devices. An example of this type of configuration would be instructing all routers, along a path in the network, to provide a specific quality of service to a particular set of customers. Historically, network operators used mechanisms and protocols such as SNMP (Simple Network Management Protocol) [1] and CLI (Command Line Interface) to provision configurations to network devices. Today these protocols have some difficulties due to their low level granularity and their device specific nature. Today it's clear that network administrators and the existing software of configuration management can't face the

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35674-7_66 complexity growth of networks. Some IETF working groups (Policy Framework WG) suggest solving these problems by moving configuration management from device to network layer. This high level management can be obtained with the specification of management policies and their mapping into proper configurations, which have to be distributed to the "right" network devices to be enforced. This abstraction mechanism allows to have simplified management data and to exploit commonality across devices. These two features, simplified management data and re-use of configurations, can lead to the automation of some configuration management tasks allowing the network to operate with minimum human intervention. This work presents a study carried out to evaluate the performance of configuration provisioning by COPS-PR protocol [3], in a policy based network. As case study we have considered the provisioning of DiffServ (Differentiated Services) configurations to a DiffServ enabled device. Section two presents the architecture of a policy based network and the interaction between its components for the configuration provisioning. Section three presents the developed prototypes. Section four presents the performance measurements and, finally, section five the results. The presented work is innovative since it investigates policy provisioning and, evaluates the efficiency of policy based configuration management.

2. BACKGROUND

Policies are set of rules that allow defining the desired behavior for network resources or for distributed systems. As defined in [4], policies are a mean to translate high level objectives into proper configuration of network devices. To be implemented in an automated environment as a network, policies must be transformed from abstract rules to functional ones. At this level policies are ECA (event-condition-action) rules [4]. IETF Policy Framework WG works especially on the "condition action" part to define a UML information model for the representation of policy information [5] [6]. These models can be extended to represent policy for a specific management area (e.g. QoS policy information model). To control a network via policies, an administrator needs a set of software tools that allow the definition and deployment of policies to network devices. With "policy based network" we mean the set of tools and systems that are at administrator's disposal and are policy compliant. The major components of this architecture are PDP, PEP and COPS-PR protocol. In this work we have studied the distribution of DiffServ configuration [9].

2.1 PDP/PEP "configuration provisioning" interaction

This subsection describes the interaction between a PDP and a PEP in the "configuration provisioning" scenario [3].

When a device boots, it opens a COPS-PR persistent connection to its primary (default) PDP. After the connection is established the PEP sends, information about itself to the PDP, as a configuration request. This information includes the controlled device capabilities and PEP specific information used by the PDP to determine relevant policies for that PEP. In response to this request, the PDP sends policies/configurations to the PEP for the enforcement. In the paper we will refer to this behavior as *"start up"* interaction. If the PDP detects any events that require a configuration change, then it sends the changes to the PEP. The events that cause the PDP to reconfigure policies could be: the deployment request of a new configuration made by the network administrator, the activation of an higher level policy, the configuration request to the PDP made by other sources besides the PEP. We have called this behavior *"external event"* interaction.

3. THE DEVELOPED PROTOTYPES

During the work have been developed two pairs of PDP/PEP, called "start up" and "decision", as well as two PIB for the representation of policy information. All the prototypes have been developed in C++ language on Linux OS (RedHat distribution, version 6.2). For the development of PDP and PEP we have used a free set of COPS-PR API (Application Program Interface) developed by the Vovida Organization.

3.1 The two PIBs

A PIB is a logical database used for policy information which allows the PDP and PEP to share information in a standard way, using a common information syntax and semantic. Policy information is represented by PRC (Provisioning Classes) and related instances PRI (Provisioning Instance). We have developed part of two different PIBs: the framework [7] and the DiffServ [8] PIB. The implementation of the first one allows the organization of network device capabilities into an "interface set", as well as, the representation of the support information, needed by the PDP, to determine the relevant configurations for a PEP (e.g. role combination). The second permit to represent policy for the configuration of a DiffServ enabled device interfaces. The DiffServ PIB knows the concept of "data path" configuration [9]. This is the sequence of processing elements that must elaborate an IP packet so that it can obtain the correct DiffServ treatment. In our DiffServ PIB implementation the IP traffic can be classified (by a filter or by an access list); the classified traffic can be marked with a DSCP (Differentiated Service Code Point) code (by a packet marker); the marked traffic can be queued as input to a scheduler. Each element (the filter, the packet marker, the queue and the scheduler), according to standard DiffServ PIB, is modeled by one or more PRCs and is configured in compliance with the QoS policy we want to express. We have also implemented some PRCs of the DiffServ PIB that allow representing the DiffServ capabilities of an interface (e.g. classification capabilities, queuing and scheduling).

3.2 PDP and PEP

The first pair of PDP/PEP is called "start up". These two prototypes simulate the interaction between a PDP and a PEP that request its initial configuration ("start up" interaction, see 2.1). The second pair of PDP/PEP is called "decision". These prototypes simulate the interaction between a PDP and N PEP, when the PDP detects an "external event" that involves a (re)configuration of PEPs ("external event" interaction, 2.1).

Start up PDP and PEP

The start up PDP is a multi threaded server that can manage N start up PEP. Figure 1 shows the interaction between PDP and PEP.

The PEP connects to the PDP and sends, as COPS-PR request message, the DiffServ capability, of the interface that it controls, and the information used by the PDP to determine the relevant configuration for it. The PDP installs the capability in its own PIBs and, then sends the DiffServ configuration to the PEP in a COPS-PR decision message. The DiffServ configurations are contained into the DS PIB, previously loaded in memory by the PDP. The implemented PDP is state-full. To maintain the correct association between a single PEP and its capability, the PDP inserts the PEP identifier (PEPid object) and the capability received into appropriate associative maps. To build the decision message with the DiffServ configuration, the PDP exploits the DiffServ "data path" concept. The sequence of the constituent elements of the data path is fixed, while their number and their interconnection pattern is variable. The PDP obtains the configuration by browsing along the data path and exploring it completely, using the interconnection information included in every PRC. The "correct" data path is determined by the PDP using the "role combination" information received from the PEP. When receiving the decision message the PEP parses and installs it, copying, the configuration in its own DiffServ PIB.

Decision PDP and PEP

Our implemented "decision PDP" is a multi threaded server that takes an operative event and, as response to this, configures N connected "decision PEPs". The decision messages with the configuration are built as previous. The PEP parses the received message and installs the configuration in its DiffServ PIB. The operative event we selected is the Nth COPS OPEN registration message received by server. This event is simple to implement, to monitor and, from measurement perspective, it is logically equivalent to an external request or a higher-level policy activation. Figure 2 shows the interaction between a decision PDP and its client.

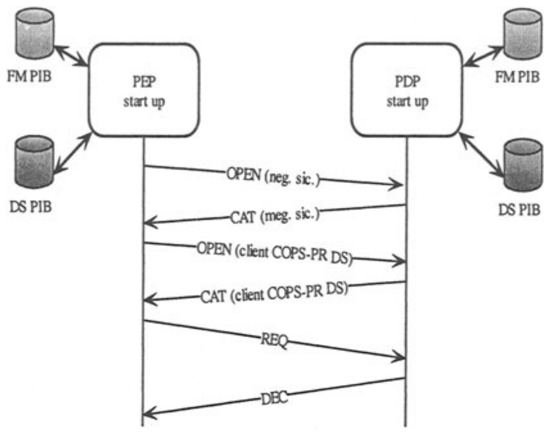


Figure 1. **PDP/PEP start up interaction.**

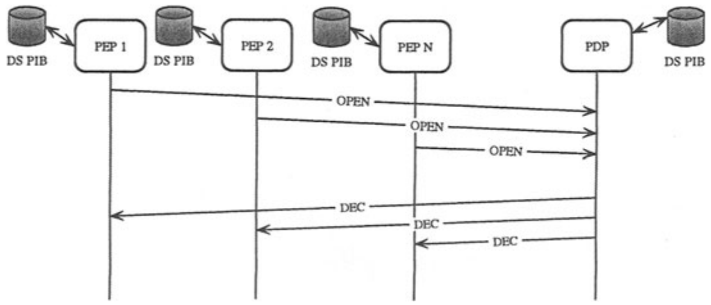


Figure 2. **Decision PDP/PEP interaction**

4. EXPERIMENTAL ENVIRONMENT

Based on the implemented PDP, PEP, and PIB prototypes a set performance measurement has been carried out. The executed measures have been designed to cover, in an exhaustive way, the two lower levels of the architecture of a policy based network (PDP and PEP). All measures have been also repeated with four different DiffServ configuration in order to evaluate the impact of the policy complexity on the recorded times. We have used two different test bed. In the first test bed (called "start up" test bed) a computer is dedicated to the start up PDP process and a computer is dedicated to the corresponding PEP. In the second test bed (called "decision" test bed) a computer is dedicated to the PDP process and two separated machines host up to N PEP running in parallel, with a maximum of 500 processes. In both test beds the computers are connected by a 100 Mbit switched LAN.

4.1 Measurements

Start up interaction time (T*start-up*).

With this measurement we want to estimate the time for a start up interaction on the server side. It is done (in the "start up" test bed) with one "start up PDP" and one "start up PEP", since is improbable that more than one PEP starts up simultaneously, or that a PEP starts before that the PDP ends to serve the previous one. We measure the time elapsed between the arrival at the PDP of the first "OPEN" message and the return of the "send" function of the decision message (Figure 1). T*start-up* will depend on: the number of the messages exchanged during the interaction, their dimensions and the processing related to every message (e.g. the necessary time for the construction of the DEC message). The following table 1 shows the exchanged messages and their dimensions. One can see how the dimensions of the DEC message changes according to the contained policy.

Table 1. Message dimension

	PEP → PDP			PEP → PDP	
# MSG	MSG	Dim. (Byte)	# MSG	MSG	Dim. (Byte)
1	OPEN(sic)	40			
			2	CAT(sic)	
3	OPEN (Client)	40			
			4	CAT(Client)	
5	REQ	904			
			6	DEC	Pol. A= 528 Pol. B= 936 Pol. C=1256 Pol. D=1616

Decision time (Tdecision).

With this measurement we want to estimate the time, needed by a "decision PDP", to provision N "decision PEPs" with a DiffServ configuration. The measure is done with the "decision" test bed. We measure the time elapsed between the recording of the "operative event" and the return of the "send" function of the last decision message to the last served PEP (Figure 2). In this interaction the PDP process N decision messages, each sent to a PEP. Tdecision will depend on: the number of the PEPs that must be configured by the PDP and the policy complexity. The dimensions of the DEC message (see previous table) and the time necessary to it construction will reflect on the last contributing factor.

Provisioning time (Tprovisioning).

With this measurement we want to estimate the entire provisioning time, including the installation time of configurations by the last "decision PEP" (client side). We measure (in the "decision" test bed) the time elapsed between the sending of the last PEP registration message (COPS-PR OPEN message) and the installation of the DiffServ configuration made by the last PEP served (Figure 2). Also in this measure, as in the previous one, N decision messages are exchanged and processed.

Policy information base update time, (TPIB).

With this measurement we want to estimate the updating time of the DiffServ PIB on PEP side. We measure the time elapsed between the return of the "receive" function of the decision message and the end of the PIB update function. We have used the "time stamp counter" of the Pentium processor for time measurements. The counter is incremented by one every clock cycle and can be used to record the execution time of a set of instructions. This time is the difference between the value of the counter just after and just before the instruction block, divided for the processor frequency. The value of the counter can be read using the assembler (Intel) instruction RDTSC (read time stamp counter). This method has been chosen for its high precision (theoretically only one clock cycle) and for its low overhead. This method is applicable to the proposed measurements with a few other considerations due to the multi-threaded execution of the PDP.

5. THE PROVISIONED POLICIES, THE RESULTS

This section presents the results. Initially we discuss the policy and the data path, while the last section deals with the results done on these policies.

5.1 Policy A, basic complexity

This case study considers the scenario of a financial department that is the source of FTP (File Transfer Protocol) traffic. This traffic must be aggregated and marked with a DSCP code equal to AF11 (Assured Forwarding 11) for its treatment by nodes of the DiffServ domain. The aggregate flow must be granted 10%- 15% of the total available bandwidth. Figure 3 shows the data path representing the policy and, thus, the configurations sent from PDP to PEP for the policy enforcement. Each rectangle is an entry of a specific PRC of the DiffServ PIB of the PDP. Note how the condition part of the policy ("IF" part) is translated into an appropriate access list of the PIB (classifier element and IP filter), while the action part is rendered through specific PRCs (DSCPMarkAction, Queue, Scheduler) with appropriate parameters.

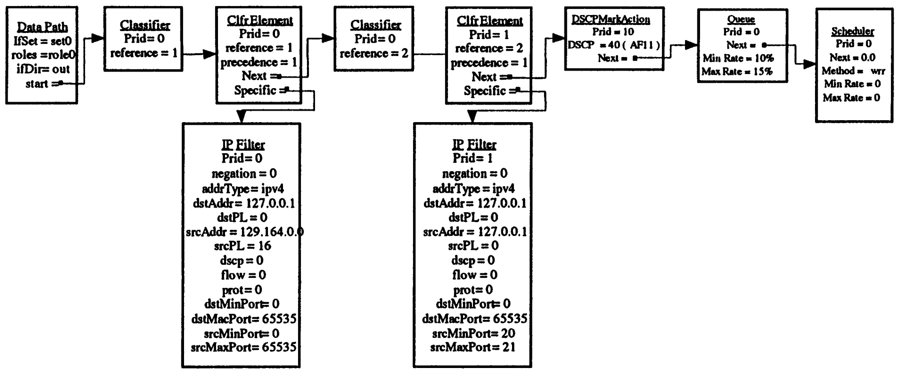


Figure 3. Policy A data path

5.2 Policy B, medium complexity

This case study considers the scenario of a design department with a remote projects database. Thus the main traffic is a FTP flow to the DB to which a bandwidth between 15% and 50% (of the total) must be granted, to reflect its strategic relevance. The FTP flow must be marked as AF11 traffic. The department generates both Telnet and SMTP (Simple Mail Transfer Protocol) traffic in addition to FTP. Telnet Traffic is granted 10 - 30% of the bandwidth and is marked as AF12, while to the SMTP is given 5 - 10% and is marked as AF13.

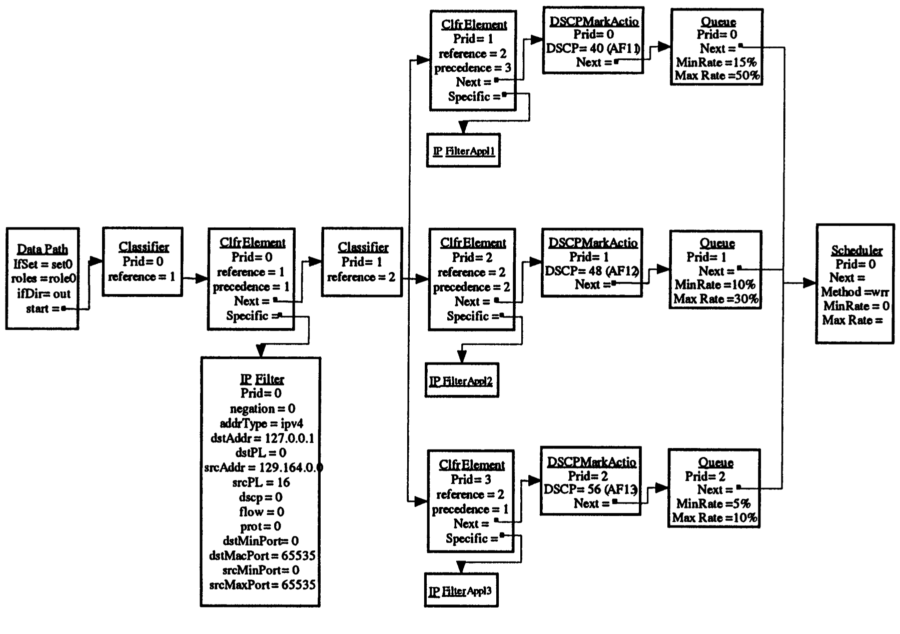


Figure 4. Policy B data path

5.3 Policy C, high complexity

This case study considers the scenario of a sub-net source of various traffic flows, from five different applications. These, however, can be divided into three classes of traffic: valuable, medium interest and low interest traffic. A single application (e.g. VoIP) generates valuable traffic which must be marked as EF (expedited forwarding) and granted 10 - 15% of the total bandwidth. Two different applications, (e.g. web application and FTP) generate medium traffic which must be granted 5-10% of the bandwidth with packets marked as AF11 and AF12 respectively. Another two applications belong to the low interest traffic class (e.g. telnet and SMTP) so are allocated 25 - 5% of the bandwidth with packets marked as AF21 and AF22 respectively.

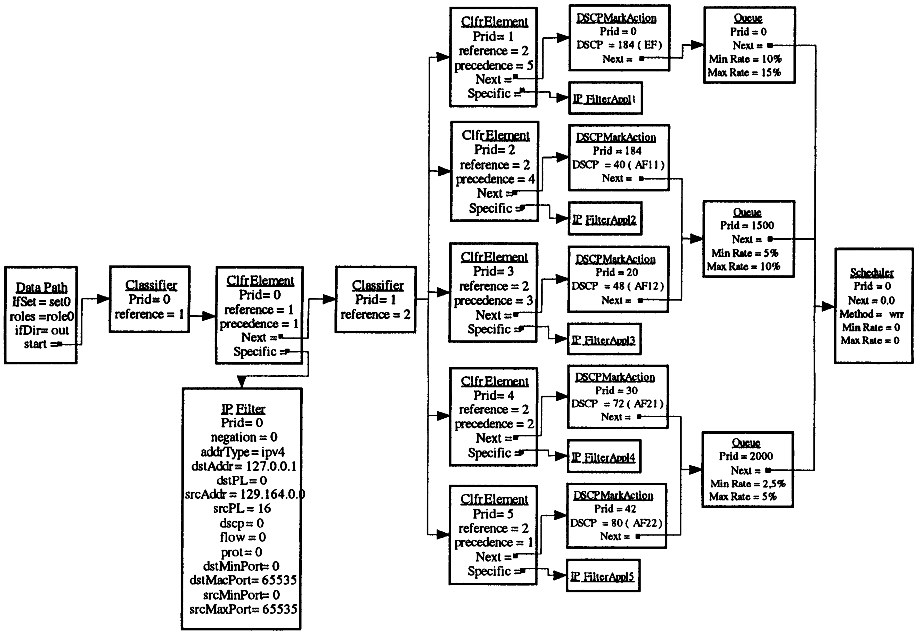
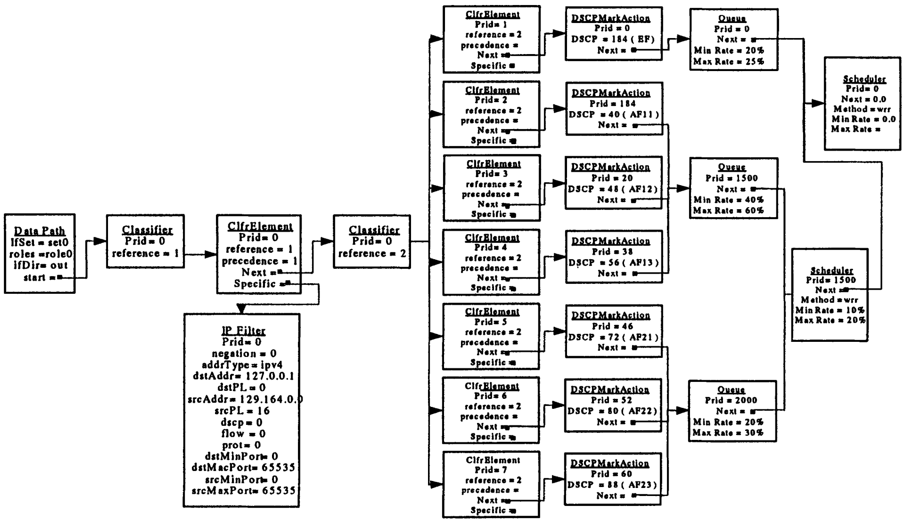


Figure 5. Policy C data path

5.4 Policy D, very high complexity

This scenario is similar to the previous one but with traffic from seven different applications. Three classes of traffic are distinguished. The first one is constituted by a single EF application with 20 - 25% of the total bandwidth. The other two classes consist of three applications. Application packets of the first family are marked as AF1i (i = 1,2,3), while application packets of the second as AF2i (i = 1,2,3). To these two classes is granted a bandwidth between 10% and 20% of the total. The granted band is divided into the two families as follows: 40 - 60% (of the granted 10% or 20%) goes to family AF1, while 20 30% (of the granted 10% or 20%) goes to family AF2.



Data Path IfSet = set0 roles = role0 IfDir = out start =
Classifier Prid = 0 reference = 1
ClfrElement Prid = 0 reference = 1 precedence = 1 Next = Specific =
IP Filter Prid = 0 negation = 0 addrType = ipv4 dstAddr = 127.0.0.1 dstPL = 0 srcAddr = 129.164.0.0 srcPL = 16 dscp = 0 flow = 0 prot = 0 dstMinPort = 0 dstMacPort = 65535 srcMinPort = 0 srcMaxPort = 65535
Classifier Prid = 0 reference = 2
ClfrElement Prid = 1 reference = 2 precedence = Next = Specific =
ClfrElement Prid = 2 reference = 2 precedence = Next = Specific =
ClfrElement Prid = 3 reference = 2 precedence = Next = Specific =
ClfrElement Prid = 4 reference = 2 precedence = Next = Specific =
ClfrElement Prid = 5 reference = 2 precedence = Next = Specific =
ClfrElement Prid = 6 reference = 2 precedence = Next = Specific =
ClfrElement Prid = 7 reference = 2 precedence = Next = Specific =
DSCPMarkAction Prid = 0 DSCP = 184 (EF) Next =
DSCPMarkAction Prid = 184 DSCP = 40 (AF11) Next =
DSCPMarkAction Prid = 20 DSCP = 48 (AF12) Next =
DSCPMarkAction Prid = 38 DSCP = 56 (AF13) Next =
DSCPMarkAction Prid = 46 DSCP = 72 (AF21) Next =
DSCPMarkAction Prid = 52 DSCP = 80 (AF22) Next =
DSCPMarkAction Prid = 60 DSCP = 88 (AF23) Next =
Queue Prid = 0 Next = Min Rate = 20% Max Rate = 25%
Queue Prid = 1500 Next = Min Rate = 40% Max Rate = 60%
Queue Prid = 2000 Next = Min Rate = 20% Max Rate = 30%
Scheduler Prid = 0 Next = 0.0 Method = wrr MinRate = 0.0 MaxRate =
Scheduler Prid = 1500 Next = Method = wrr MinRate = 10% MaxRate = 20%

Figure 6. Policy D data path

5.5 Results

This subsection reports the measurement results obtained for the various policies described above. To compare policies we use, as ordering criteria, the number of the elements in the corresponding data path and their size of the policy (in bytes) when inserted into decision message sent by PDP (the fixed overhead of the COPS-PR protocol is not counted). The following table 2 shows the provisioned configurations, ordered according to the criteria just expressed.

Start up Time (Tstart up).

Table 3 shows the results for this measure. We have a mean time of 0.365 seconds, which is an acceptable time considered the poor performance of the computer and the length of the interaction. If we compare the mean value of the Tstart up we note a substantial independence of this time for the different types of provisioned policy. This is perhaps because the measured interaction between "start up PDP" and "start up PEP" is so long that it "masks" the dependency of the PDP time with the configuration.

Table 2. Policy element number and dimension

	Policy A		*Policy B*		*Policy C*		*Policy D*	
	Num	Byte	Num	byte	Num	Byte	Num	byte
DATA PATH	1	48	1	48	1	48	1	48
CLASSIFIER	2	56	2	56	2	56	2	56
CLASSIFIER ELEMENT	2	96	4	192	6	288	8	384
IP FILTER	2	144	4	288	6	432	8	576
DSCP MARKER	1	40	3	120	5	200	7	280
QUEUE	1	44	3	132	3	132	3	132
SCHEDULER	1	40	1	40	1	40	2	80
TOTAL	10	468	18	876	24	1196	31	1556

Table 3. Mean and Standard deviation

Policy	Mean (sec)	Standard deviation
A	0.371377	0.000658
B	0.357130	0.028821
C	0.370961	0.002597
D	0.361734	0.053850

PDP Decision Time (Tdecision)

Figure 7 shows the tendency of the mean PDP decision time with respect to the function of the number of PEPs that must be provisioned. The trend of this function are clearly linear with the number of PDP clients (i.e. PEPs). This function indicates a good architecture scalability with respect to the classical exponential trend of response time, especially for a high number of PEPs (devices) to configure. The decision time is variable with policy complexity in a sensible way. Higher complexity policies grow more quickly, but always linearly.

Total provisioning time (Tprovisioning)

The total provisioning time also has a linear trend with the number of the connected PEPs (figure 8). The provisioning time is also depends on the configuration considered. If we calculate the time difference between the function of figure 8 and that of figure 9 we obtain the function shown in figure 11. From the trend of this figure we can argue that the total provisioning time (Tprovisioning) is the sum of the total PDP decision time (Tdecision) plus a constant. It's interesting to note that the value of this constant term is slightly than Tdecision. In other words, the PDP time (Tdecision) is the much heavier component of the total provisioning time of a configuration (Tprovisioning). Thus the PDP is the bottleneck of the architecture.

Policy Information base update time, TPIB

The PIB updating time obviously depends on the DiffServ configuration that we considered. This time, however, is so short that it's negligible with respect to the

PDP time and its high variability is not visible. Figure 10 shows the mean updating time on the X axis while the empiric frequencies of the observed time are shown on the Y axis.

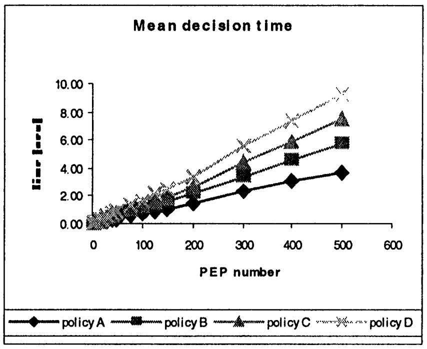


Figure 7. PDP decision time

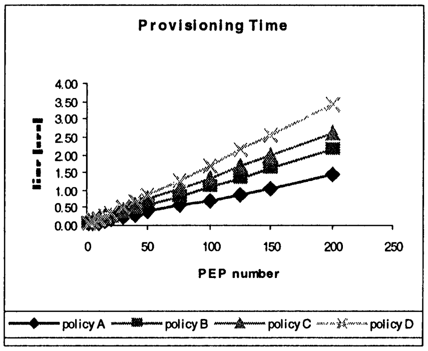


Figure 8. Total Provisioning time

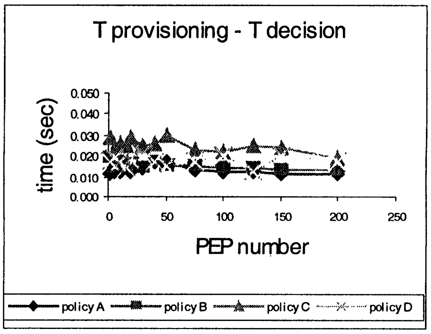


Figure 9. Provisining time minus decision time

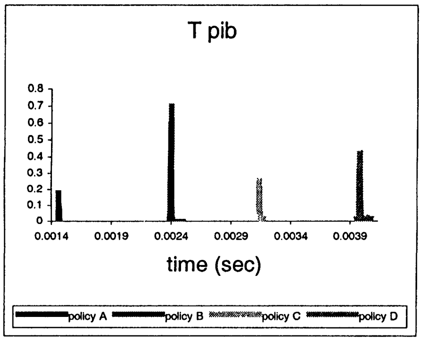


Figure 10. PIB installation time

6. CONCLUSIONS

In this paper we presented a range of COPS-PR measures, simulating the behavior of a policy based network, when acting in the configuration provisioning scenario. The prototypes are the "start up PDP" and the "decision PDP" and the client (PEP) developed as part of two different PIBs. We also proposed a simple way to translate a generic PRC into an object oriented language. The realized prototypes permit us to evaluate performance of COPS-PR protocol. The estimated times cover the two lower level of the architecture of a policy based network. In particular we have determined the linear trends of the PDP decision time for the provisioning of N PEP, underlining good architecture scalability and of our PDP prototypes, also for a big number of client (PEP). Further we have determined the bottleneck of the enquired architecture: the PDP. Finally, we have evaluated the impact of policy complexity on the overall processing time, pointing out that it is negligible in a start up scenario while it's influence is much greater in an "external event" scenario. It seems that the policy based approach can efficiently be used, in fact the policy interaction protocol has a linear growth with the number of PEPs, so the architecture and the protocols proposed by IETF are scalable. One more issue that can be further investigated is how simple to use are policies and how user friendly are policy based applications. In fact other specific technology as like as SNMP, CLI and SCRIPT, able to support "configuration management" were affected by usability problem.

7. REFERENCES

[1] J. Case, D. Harrington, R. Presuhna, B. Wijen, "Message Processing and Dispatching for the Simple Network Management Protocol", RFC 2572, April 1999.

[2] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry, "The COPS (Common Open Policy Service) Protocol", RFC 2748, January 2000.

[3] K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, A. Smith, "COPS Usage for Policy Provisioning (COPS-PR)", RFC 3084, March 2001

[4] R. Wies, "Policy in Network and Systems Management – Formal Definition and Architecture-", Journal of Network and System Management, volume 2, number 1, March 1994

[5] B. Moore, E. Ellesson, J. Strassner, A. Westerinen, "Policy Core Information Model", RFC 3060, February 2001

[6] B. Moore, L. Rafalow, Y. Ramberg, Y. Snir, A. Westerinen, R. Chadha, M. Brunner, R. Cohen, J. Strassner, "Policy Core Information Model Extensions", internet-draft, November 2001

[7] M. Fine, K. McCloghrie, J. Seligson, K. Chan, S. Hahn, R. Sahita, A, Smith, F. Reichmeyer, "Framework Policy Information Base", internet draft, November 2001

[8] M. Fine, K. McCloghrie, J. Seligson, K. Cha, S. Hahn, C. Bell, A. Smith, F. Reichmeyer, " Differentiated services Quality of Service Policy Information Base", internet draft, November 2001

[9] Y. Bernet, K. McCloghrie, J. Seligson, K. Chan, S. Hahn, R. Sahita, A. Smith, "An informal management model for Diffserv routers", internet draft, February 2001.

[10] Vovida organization, www.vovida.org

[11] L. Lewis, "Policy-Based Configuration Management: A Perspective from a Network Management Vendor", The Simple Times, volume 2, number 1, September 2000

[12] J. Roese, "Configuration Management Services for the Large Enterprise Network", The Simple Times, volume 2, number 1, September 2000

[13] L. Lewis, "Implementing Policy in Enterprise Networks", IEEE Communications Magazine, January 1996.

[14] R. Yavatkar, D. Pendarakis, R. Guerin, "A framework for policy-based admission control", RFC 2753, January 2000.

[15] Y. Nomura, A. Chugo, M. Adachi, M. Toriumi, "A policy based Networking Architecture for Enterprise Networks", IEEE Internation Conference on Communications, 1999.

[16] M. Damianou, N. Dulay, E. Lupu, M. Slaman, T.Tonouchi, "Tools for Domain-based Policy Management of Distributed Systems" NOMS2002, April 2002.