An Integrated System for QoS Monitoring of Policy-Based Networks

Marcelo Borges Ribeiro, Lisandro Zambenedetti Granville
Maria Janilce Bosquiroli Almeida, Liane Margarida Rockenbach Tarouco

Federal University of Rio Grande do Sul - UFRGS
Institute of Informatics
Av. Bento Gonçalves, 9500 - Bloco IV
Porto Alegre, RS - Brazil

{mribeiro, granville, janilce, liane}@inf.ufrgs.br

Abstract. *Policy-based management and QoS monitoring are both tasks related to the management of modern QoS-enabled network. Although related to each other, these tasks are currently executed in a non integrated fashion. This paper presents an architecture that integrates policy-based management and QoS monitoring through the extension of the original IETF policy-based management architecture. The main advantage of using our proposed approach is that network administrators are freed to execute other tasks, while the QoS-enabled network is still monitored. Another advantage is that we are monitoring policies and verifying if they are respected in the network even after its deployment, which is a new feature absent in the IETF solution.*

1. Introduction

The provisioning of QoS facilities in computer networks is intended to introduce levels of guaranties that are absent, for example, in the IP best-effort approach. Different QoS architectures have different accuracy for the provided services, which leads to a scenario where the expected QoS may not be always properly provided by the underlying architecture. In this context, QoS monitoring [Eder and Nag, 2001] is required to verify the observed QoS and compare it with the expected/contracted QoS.

In a policy-based network, the expected QoS is defined by policies that orchestrate the network behavior. The Internet Engineering Task Force (IETF) has standardized several elements of a Policy-Based Network Management (PBNM) system [Moore et al., 2001], in which policies are defined using high-level languages, stored in policy repositories, deployed with the help of Policy Decision Points (PDPs) and enforced by Policy Enforcement Points (PEPs) [Westerinen et al., 2001]. In the IETF approach, once a policy is successfully deployed, the PBNM system no longer checks the policy behavior, and the QoS defined in the policy is assumed to be provided by the network.

In fact, that is not always true. The underlying QoS provisioning architecture can be unstable or face problems, and the expected QoS may not be achieved. In order to verify QoS degradation in real management environments, today, the network administrator monitors the network and determines the provided QoS. Such QoS is then

manually compared with the QoS defined by the network policies. Finally, if differences are found, the administrator proceeds with actions to lead the underlying architecture to a consistent state.

To date, policy definition and deployment, and QoS monitoring are tasks executed separately by the available solutions [Granville et al., 2001]. Also, as shown before, the verification of the provided QoS against the QoS defined by the network policies is left to the network administrator. In this context, automation of QoS-policy monitoring and integration among the cited tasks are required from the network management point-of-view.

In this paper we propose an approach (and a system) for QoS monitoring, where policy definitions are input data used to check the observed QoS. When a QoS degradation is detected, our monitoring system notifies an SNMP-enabled manager using InformRequest messages. The system's architecture is internally divided in QoS monitors and QoS monitoring controllers. Each QoS monitoring controller controls several QoS monitors that collect data from the network. Such data is compared with the policies translated by the controller, and if degradations are detected the monitoring controller notifies the SNMP-enabled manager. The communication between controllers and monitors is also SNMP-based. The main goal here is to provide a solution that integrates QoS monitoring and PBNM in a unique management environment.

The remaining of this paper is organized as follows. Section 2 reviews QoS monitoring and policy-based management, while Section 3 introduces the proposed QoS monitoring system. The solution's internal architecture and communication are described in Section 4. Finally, Section 5 concludes de paper and outlines future work.

2. Related Work

Several monitoring systems have been proposed and are widely available. Some systems, like NetSaint [Ballard, 2001] and MRTG [Oetiker, 1998], verify services availability using SNMP-based polling, but such systems introduce management messages that consume network resources. Other systems, like ntop [Deri et al., 2001] and NeTraMet [Brownlee, 1997], operate as network sniffers watching the network traffic without introducing management messages. Even in this case, some management messages will exist to provide communication between the monitoring system and the management station. RMON and RMON2 [Waldbusser, 1997] are also examples of such systems.

Although the amount of available monitoring systems is expressive, specific QoS monitoring systems are not. IETF has been working in QoS monitoring-related issues, and its rmonmib working group has proposed the Application Performance Measurement (APM) MIB [Waldbusser, 2001] and the Application Response Time (ART) MIB [Warth and McQuaid, 1999]. The general approach is to extend the RMON MIB to provide information about end-user perceived application performance (APM MIB) and network provided services performance (ART MIB). An SNMP-enabled manager retrieves performance information and checks if the observed performance is consistent with the contracted QoS. Although such MIBs are still IETF drafts (ART MIB is currently an expired draft), some implementations can be already found [Systems, 2001]. Tham, Jiang and Ko [Tham et al., 2000] have developed important work defining QoS

monitoring schemes that are able to retrieve not only flow end-to-end QoS, but also the per hop QoS experienced by a flow within a network. This is possible because several QoS monitors are deployed in diverse network segments. The monitors are coordinated by a higher level entity that collects monitored flows information and calculates the provided QoS in each network segment [Jiang et al., 2000]. From a policy-based perspective, one important drawback about these QoS monitoring systems is the fact that no integration with policy-based systems is provided.

Policy-Based Network Management (PBNM) has been seriously investigated. Universities, standardization bodies and network device providers are running research projects about several PBNM issues, from formal aspects (like definition of policy description languages [Lobo et al., 1999] and mechanisms to detect/solve policy conflicts [Lupu and Sloman, 1999]) to more concrete aspects (like prototype implementation and complete PBNM systems [Coelho et al., 2001]). Several works have been already published in important international management events like NOMS and IM. In 1999, the first whole policy-oriented workshop was conceived (POLICY 99), followed by POLICY 2001 and POLICY 2002.

PBNM systems have several functionalities and among them one can find: policy definition, policy translation and policy deployment verification. Once a policy is defined (ideally using a high level abstract language) a translation element (a Policy Decision Point - PDP) is supposed to translate such policy to device-specific actions, i.e. translations are required to deploy the policy into the network devices. Policy verification is needed because the policy deployment may fail due to lack of resources or device unavailability, among other reasons [Lupu and Sloman, 1999]. If the policy deployment fails, the PDP notifies the management station, but once the deployment succeeds no further policy verification is performed by the PBNM system.

A critical problem with PBNM and QoS monitoring is the lack of integration. To check if a policy fails after its deployment, the network administrator is forced to activate a separated QoS monitoring system to retrieve the real QoS provided by the underlying QoS architecture. Then, a comparing procedure is manually executed by the administrator to check QoS degradations, using as input data the policy definition and the collected QoS information. In this context, we believe that QoS monitoring and PBNM integration is important for the verification of QoS degradation in an automated fashion. The QoS monitoring system presented in this paper is designed to provide an automated QoS verification based on network policies.

3. Policy-Based QoS Monitoring System

According to the IETF work, a typical PBNM system is composed by 4 main elements [Moore et al., 2001]: management application, Policy Repository, Policy Decision Point (PDP) and Policy Enforcement Point (PEP). The management application provides the user interface that allows a network administrator to edit, store and deploy network policies. To deploy a policy, first the policy is edited and stored in the policy repository. The network administrator then determines in which devices (and PEPs) the stored policy is going to be deployed. The PBNM system, on its turn, selects the PDPs to be used in the policy deployment task. Each selected PDP is contacted by the management

application and the policy is transferred. Finally, the PDP translates the policy to device-specific actions, configuring the devices' internal PEPs previously selected by the network administrator. For further details on PBNM elements one can check the IETF work [Moore et al., 2001].

In this work, we extend this standard PBNM architecture to include elements able to provide QoS monitoring facilities in an integrated environment. Figure 1 presents the proposed extended architecture. On the left hand, one can see the standard PBNM elements (policy repository, PDP and PEP) and the common interactions through the dotted lines. On the right hand, the new introduced elements are presented, and the solid lines show the interactions among these elements. Taking figure 1, the next subsections explain the general operations performed by each architecture's elements from a bottom-up perspective.

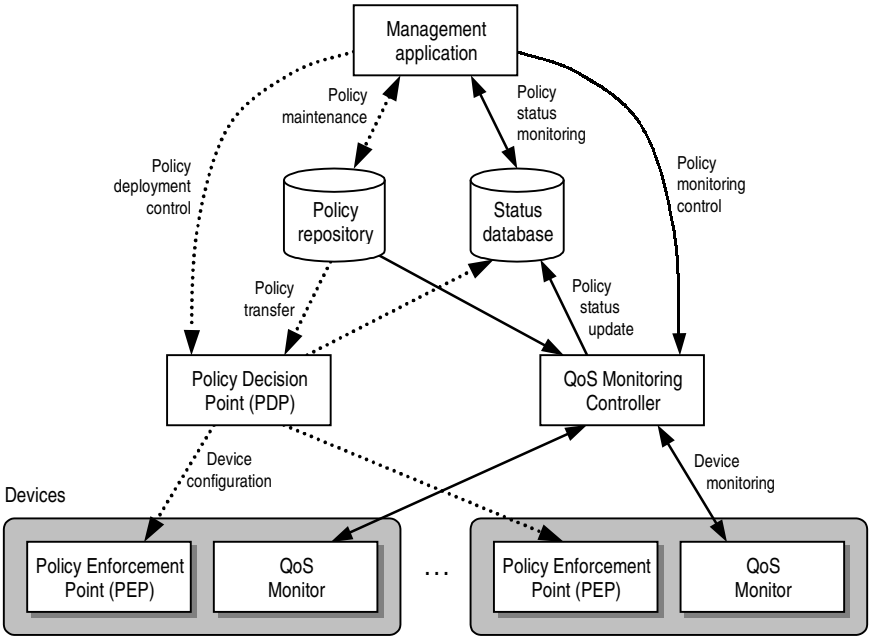

Figure 1: Policy-based QoS monitoring system

3.1. Devices, PEPs and QoS Monitors

In a policy-based network, each device holds PEPs that are internal elements that enforce the policies defined by the network administrator. Examples of PEPs are the queuing disciplines of the interfaces of a router, internal devices' prioritization processes and the marking element in a DiffServ-enabled router. A PEP is configured accessing the device that holds the PEP using some communication protocol. IETF suggests COPS/COPS-PR [Chan et al., 2001] protocol, but SNMP, Telnet, HTTP, among others, can be used as well.

Once a PEP is configured, the PBNM system assumes that the deployed policy is going to be correctly respected, and no further policy check is performed. To allow these further verifications, we use a QoS monitor that performs throughput and jitter checking for defined network flows inside a device. The concept of a QoS monitor was previously introduced by the Jiang, Tham and Ko [Tham et al., 2000] [Jiang et al., 2000]. Here, we slightly change the monitor internal operations (further verified), but the general concept

is the same. The monitor is configured through the definition of flows to be monitored, which are described by: source and destination network addresses, source and destination transport ports, transport protocol and DSCP/ToS [Nichols et al., 1998], if desired. Each flow is monitored in a specific device's interface and observed in an interface's direction: in, out or both. The monitor keeps track of the jitter and throughput associated to each defined flow and notifies external elements if the observed jitter or throughput exceeds some limits defined by the network administrator. The QoS monitor is also responsible for providing flow information to external elements, which is a requirement for flow loss and delay checking.

It is important to notice that neither the PEP nor the QoS monitor are aware of the network policies. PEP and QoS monitor are just configured by external elements that, on their turn, understand the network policies.

3.2. PDPs and QoS Monitoring Controllers

From a device's perspective, the PDPs and QoS monitoring controllers are the external elements that are aware about the network policies. PDPs receive policies from the policy repository and try to deploy them. Pull or push transfer approaches can be used to send a policy to a PDP. In the pull approach the PDP is notified about a new policy stored in the policy repository. The PDP then downloads such policy using some protocol (for example, HTTP, FTP or LDAP). In the push approach, however, the policy is sent by the management application that retrieves the policy in the policy repository and uploads it in the PDP. Although IETF suggests LDAP [Wahl et al., 1997] as the service to store and transfer policies, there is no consensus in a single solution. For example, even the IETF, in the context of the snmpconf working group, is working in another policy transfer mechanism based on SNMP [Waldbusser et al., 2002], while Martinez et al. [Martinez et al., 2002] propose the use of Script MIB to proceed with this transfer.

After receiving a policy the PDP proceeds with verification of possible policy conflicts [Lupu and Sloman, 1999]. If no conflict is detected, the PDP contacts PEPs to configure them in a way to support the policy being deployed. If a problem occurs in the deployment process, the PDP updates the policy status in the status database indicating the deployment problem and that the policy previously transferred can't be applied. In a successful process, however, one can say that PDPs proceed with policy translation, from policy definition to PEP-specific configuration.

The QoS controllers can be seen as the PDPs of the QoS monitoring process. Exactly as PDPs, QoS controllers access policies from the policy repository (through the pull or push approach using some transfer protocol), they check policy problems through device monitoring and notify external elements when policy degradations are observed. One important difference between PDPs and QoS monitoring controllers is that QoS monitoring controllers verify policies after the policy deployment, while PDPs verify policies during their deployment. The policy verification performed by the QoS monitoring controller requires some QoS monitoring controller/monitor interaction, as follows:

1. The transferred policy is analyzed and its flow definition and QoS parameters are determined. In our research we are dealing with the four main QoS parameters: throughput, delay, loss and jitter;

2. The controller contacts a set of QoS monitors and configure them informing: (a) the flow to be monitored, (b) the expected throughput and jitter associated to the flow and (c) the monitoring thresholds for the expected throughput and jitter.

Once throughput or jitter problems are detected in the monitor, a notification is sent to the monitoring controller. This notification triggers a status database update performed by the controller. Optionally, the notification can be forwarded to the management application, that can handle the detected problem. In order to verify delay and loss problems, the network controller analyzes collected data in the monitors. The experienced delay and loss are compered with the max delay and loss defined by the network administrator. If violation is detected, again, the controller updates the status database and optionally notifies the management application. Comparing QoS monitoring controllers and QoS monitors regard QoS parameters verification, the former checks for delay and loss problems, while QoS monitors take care about jitter and throughput. Section 4 provides details about these verifications inside QoS monitoring controllers and monitors.

3.3. Policy Repository, Status Database and Management Application

The policy repository is the storage for network policies. Accessing the policy repository the network administrator is able to define and store new policies, to retrieve and modify previously defined policies, and to remove old policies not used anymore. Once a policy is defined and placed in the repository, it is ready to be used.

As seen before, the policy deployment process may fail. Also, a policy may fail even after its deployment, due to QoS degradation. Thus, policies being used have a policy status that should be accessed by the network administrator to check the policy successfulness. Such policy status is supported by the status database. This database is accessed by either PDPs and QoS monitoring controllers. PDPs updates the policy status when problems are detected in the policy deployment. On the other hand, QoS monitoring controllers update the policy status when QoS degradation is detected by the monitoring system after the policy deployment.

Finally, the top element of our architecture is the management application. It is the central point from where the policy deployment and QoS monitoring processes are controlled. In order to proceed with such control, the management application uses both policy repository and status database, and have direct access to PDPs and QoS monitoring controllers. Next section presents the system's operations, that are executed under the control of the management application.

4. System's Operational Issues

Using the management application, the following steps are typically taken in the policy deployment and monitoring processes.

1. The network administrator defines policies to be used in the network and stores these policies in the policy repository;
2. The administrator selects PEPs where some policy is going to be deployed;

3. The management application, based on the selected PEPs, determines which PDPs have to be used in the policy deployment and transfers to those PDPs the policy definition and a list of PEPs previously selected by the administrator. It means that in the policy deployment process the network administrator is unaware about the PDPs, but the PEPs;
4. To check the evolution of the policy deployment process the administrator accesses the policy status database;
5. Critical policies have to be monitored. The network administrator informs to the management application some critical policies and monitoring thresholds and, based on its previous deployment information, the management application selects QoS monitoring controllers and QoS monitors that verify the critical selected policies;
6. Optionally, the administrator can select further controllers and monitors by hand, or deselect previously selected ones;
7. Then, the management application transfers the critical policies and thresholds to the monitoring system, sending also a list of QoS monitors to the selected QoS monitoring controllers;
8. Finally, the administrator checks if the critical policy is being respected in the network accessing the policy status database, or optionally the monitoring system notifies the administrator directly.

The next subsections present further considerations about the system that allows the execution of the steps above.

4.1. Relevant QoS Monitors Selection

A key aspect of the QoS monitoring system is the selection of relevant QoS monitors. Tham, Jiang and Ko [Tham et al., 2000] proposes the use of monitors that register themselves in a real-time application name server (RTANS) for each perceived flow that passes through the monitors. The management application can determine the relevant monitors of a flow accessing the RTANS. These procedures ease the relevant monitors selection in the management application, but force each monitor to keep track of every single flow perceived. In our approach, on the other hand, the determination of relevant monitors is not so easy, but it allows the selected monitors to store and analyze few specific flows provided by the management application. We use the following scenario to illustrate the monitor selection question.

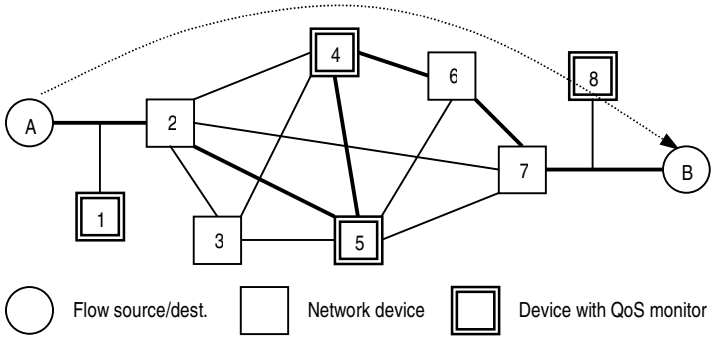

Figure 2: Relevant QoS monitors selection example

Figure 2 presents a policy-based network with some QoS monitors. Host A generates a flow that is sent to host B. This flow is routed through the network devices ♯2, ♯5, ♯4, ♯6 and ♯7. Some devices have an internal QoS monitor (devices ♯4 and ♯5); others don't (devices ♯2, ♯6 and ♯7). There are also devices that do not route the generated flow, but are used in the monitoring process (devices ♯1 and ♯8). Two approaches can be used to select these monitors. The first one consists in providing a user interface that allow the network administrator to explicitly declare the monitors to be used. The second one is more sophisticated and consists in providing automated facilities that investigate the managed network and provides the identification of the QoS monitors bypassed by a flow. For example, the network administrator could inform the flow to be monitored and the management application would proceed with automatic detection of the flow path and its associated monitors. In both cases, a list of QoS monitors to be used is generated, either manually (in the first case) or automatically (in the second case).

If end-to-end QoS is more important than core network QoS, than only monitors ♯1 and ♯8 could be used. If a specific hop QoS is important, one could use monitors ♯4 and ♯5. It is important to notice that in some cases it is not possible to retrieve the QoS of a segment. For instance, it is not possible to retrieve the QoS between devices ♯6 and ♯7. It means that core network QoS monitoring depends on the QoS monitors' location and availability.

In order to start the monitoring process, the next step is to transfer the policy to be monitored to the monitoring system. Since the QoS monitors do not understand network policies, QoS monitoring controllers have to be used. Although QoS monitor selection includes an interaction with the network administrator, QoS monitoring controller selection don't.

4.2. QoS Monitoring Controllers Selection and Operations

QoS monitoring controllers selection is totally coordinated by the management application. Virtually, no network administrator interaction is required. The algorithm for controllers selection is quite simple: find a set of controllers that are able to access as much as possible selected QoS monitors and are also able to be notified by such monitors.

With this assumption, we can conclude that a single flow could be monitored using a single controller, but a single controller can control the monitoring of several flows using several different monitors. Moreover, a single QoS monitor can be controlled by different controllers in the monitoring process of different flows that bypass the QoS monitor.

After the selection of the QoS monitoring controllers to coordinate the monitoring of a flow, the management application is ready to start the monitoring process. In order to proceed with that, the management application contacts the selected controllers and sends (using push or pull approaches discussed in the previous sections):

- The policy to be monitored;
- The list of selected QoS monitors;
- The thresholds for the QoS parameters.

Once these informations are available in a selected QoS controller, the first action executed is the derivation of QoS parameters to be monitored, based on the policy transferred. Then, the controller contacts the QoS monitors from the list and configure

them to monitor throughput and jitter based on the QoS values and thresholds. In order to verify delay and loss problems, the controller also programs the monitors to periodically send back to the controller quantitative information about the observed flow.

4.3. Prototype Implementation

We have implemented a system that uses the architecture proposed in the Section 3. We have used an SNMP-based approach to build the functional elements, which includes the definition of management objects to configure and control the QoS monitors and QoS monitoring controllers, and traps (actually, SNMPv2 InformRequest messages) that implements notification mechanisms used by the elements.

The QoS monitor was coded to work inside a Linux-based router, and uses `libpcap` [McCanne et al., 2002] in order to implement a network sniffer and access network traffic. The monitor is controlled through the QoSMonitor MIB (fig. 3) that allows the definition of the flows to be monitored in the *qmFlowTable*. Table programming follows the RMON approach for control tables programming, which includes, for example, objects for table control (*qmFlowStatus*) and for owner identification (*qmFlowOwner*). Flows throughput and jitter observed by the monitor are stored in the *qmObservedQoSTable*. Each valid row in the *qmFlowTable* corresponds to another row in the *qmObservedQoSTable*.

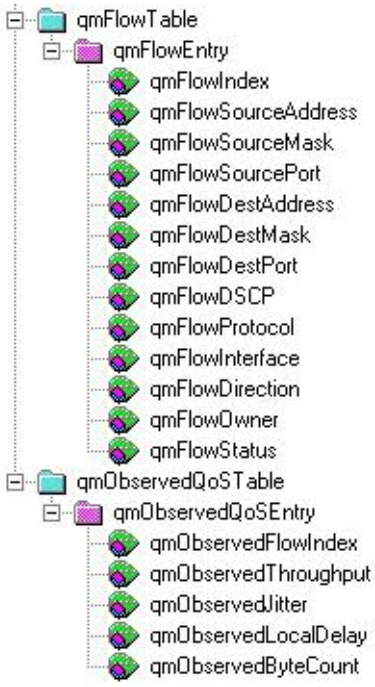

Figure 3: QoSMonitor MIB

We have also implemented the alarm and event groups from the RMON MIB. Such groups allow the automation of the qmObservedQoSTable objects verification and permits the definition of thresholds for jitter and throughput. Once such thresholds are crossed by a flow with QoS degradation, RMON events are sent to external managers (in this case, such managers are the QoS monitoring controllers). Although RMON events are originally implemented through SNMP traps, in our QoS monitor we have used SNMPv2

InformRequest messages due to robustness, since InformRequest can be acknowledge by the receiver, and events retransmission mechanisms can be provided to handle events lost in a congested network.

A MIB related to Monitor Controller has been defined (fig. 4), it serves as entry point to system monitor the QoS of the net. In order to work with several QoS Monitors, there is one table that serves to register those QoS Monitors making them available to be used by Monitor Controller *(qmMonitorControllerTable)*.

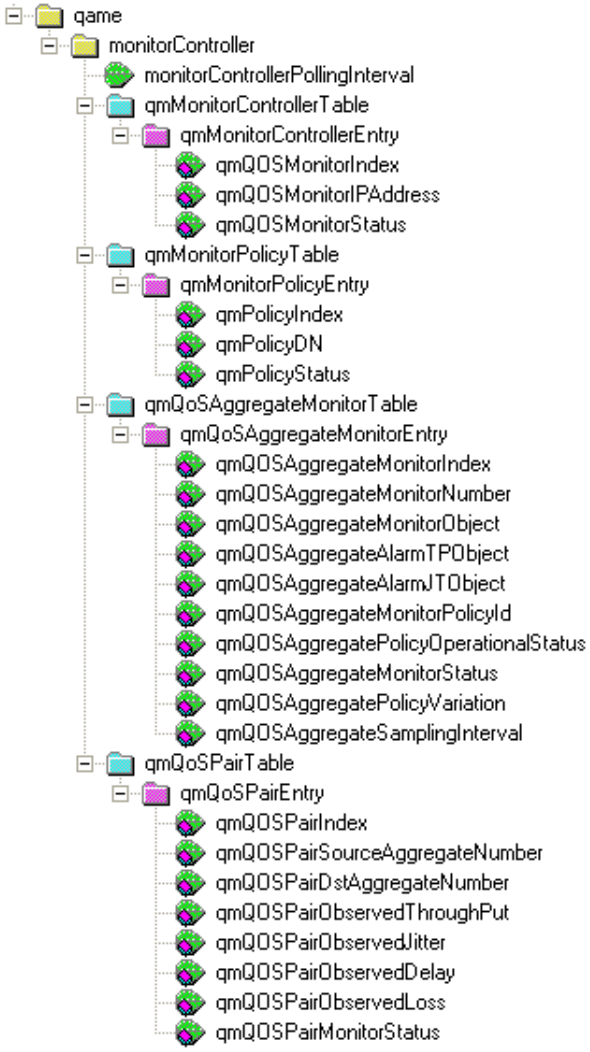

Figure 4: Monitor-Controler MIB

The next step is to register an entrance for the repository of policies where there is the policy that will be monitored *(qmMonitorPolicyTable)*. After that, it only remains to fill the next table that is responsible for the junction of a QoS Monitor with one policy *(qmQoSAggregateMonitorTable)*. This table, still has an variable *(qmQOSAggregatePolicyVariation)* that serves to specify the tolerance in the thresholds that will be attributed to RMON agent for an aggregate in particular. For example, one policy defines an outflow of 10Kbit/s and the Network Manager can define, for a particular QoS Monitor, a variability of 10% in this value is accepted, in other words, an inferior

limit of 9Kbit/s and a superior limit of 11Kbit/s. After this part, the monitoring process can initiate in the selected QoS Monitor. The last table *(qmQoSPairTable)* serves to be specified, two by two, QoS Monitors with these respective aggregates, through the entrances *qmQoSPairSourceAggregateNumber*, to indicate QoS Monitor of origin and *qmQoSPairDstAggregateNumber*, to specify a destination QoS Monitor, both responsible for monitoring one segment of the added flow. In order to verify the monitoring result, there are defined these remaining variables (with read-only purposes).

In order to transfer a policy to a QoS monitoring controller, we have implemented the pull transfer model. A QoSMonitoringController MIB allows the identification of the URL from where the controller can download a policy using LDAP. Since QoS parameters of a flow are defined within a policy, the QoSMonitoringController MIB, differently from the QoSMonitor MIB, does not need to implement objects for the definition of QoS delay or loss rates: this information is already coded in the downloaded policies.

5. Conclusions and Future Work

In this paper we have presented an architecture designed to integrate policy-based management with the monitoring of QoS-enabled networks. The main contribution of this work comes from the fact that policies originally defined to configure network devices can now also be used in the QoS monitoring process. The advantages of this approach can be verified from two different perspectives: from a QoS monitoring perspective, and from a policy-based management perspective.

Checking the current solutions for QoS monitoring, one can verify that the network administrator is often supposed to: (a) define monitoring flows, (b) define expected QoS, (c) monitor the network and discover the provided QoS, (d) compare the provided QoS with the expected QoS trying to identify QoS degradations, and finally, (e) proceed with action if degradations are detected. All these actions are manually executed. With the proposed architecture, on the other hand, (a) the network administrator uses the flows and expected QoS definition from previously defined policies, (b) the system itself checks the provided QoS and compares with the QoS specified in the policies, and (c) in case of degradation, the system notifies the network administrator. In this new scenario two main advantages arise: (1) the network administrator is freed to proceed with other actions while the provided QoS is automatically verified, and (2) the definition of flows and expected QoS is done just once, reducing multiple definition of the same information across different solutions (in this case, across a policy-based system and a QoS monitoring system).

From a policy-based management point-of-view, the system introduces the verification of policies after their deployment. Although some policy-base management approaches consider policy monitoring, the IETF approach does not. In this case, if IETF-based policies are used, the network administrator is forced to use complementary systems to manually monitor policies. However, with the proposed system the monitoring of policies is implemented through the QoS monitoring process. It also means that another contribution of our work is the fact that it compliments the IETF approach introducing the monitoring of policies, which is a feature absent in the original IETF definitions. Future work to be developed includes the improvement of the system's graphical user

interface. Currently, we are using command line oriented interface to control the elements of the system. We started to develop a Web-based interface to enhance the user interaction. Another improvement that should be coded is the implementation of pro-active management algorithms inside QoS monitoring controllers and QoS monitors. Such algorithms can detect future QoS policy degradations and notify the network administrator before the actual degradations occur, allowing the administrator to pro-actively tune the managed network and its policies.

References

Ballard, J. (2001). Netsaint Watches Over Your Network. Network Computing, 11 Jun. 2001.

Brownlee, N. (1997). Traffic Flow Measurement: Experiences with Netramet. RFC 2123, 1997.

Chan, K., Seligson, J., Durham, D., Gai, S., McCloghrie, K., Herzog, S., Reichmeyer, F., Yavatkar, R., and Smith, A. (2001). Cops usage for policy provisioning (COPS-PR). RFC 3084, Mar. 2001.

Coelho, G. A., Granville, L. Z., Almeida, M. J., and Tarouco, L. M. (2001). Network executive: A policy-based network management tool. IEEE Latin America Network Operations and Management Symposium (LANOMS), 2001.

Deri, L., Carbone, R., and Suin, S. (2001). Monitoring networks using nTop. IFIP/IEEE International Symposium on Integrated Network Management (IM), 2001.

Eder, M. and Nag, S. (2001). Service management architectures issues and review. RFC 3052, Jan. 2001.

Granville, L. Z., Ceccon, M. B., Tarouco, L. M., Almeida, M. J., and Carissimi, A. S. (2001). An approach for integrated management of network with quality of service support using qame. IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM), Nancy-France, 2001.

Jiang, Y., Tham, C., and Ko, C. (2000). Challenges and approaches in providing QoS monitoring. Wiley/ACM International Journal of Network Management, 10(6):323-334, Nov./Dec. 2000.

Lobo, J., Bhatia, R., and Naqvi, S. (1999). A policy description language. AAAI, 1999.

Lupu, E. and Sloman, M. (1999). Conflicts in policy-based distributed systems management. IEEE Transactions on Software Engineering, 25(6): 852-869. Nov.1999.

Martinez, P., Brunner, M., Quittek, J., Strauss, F., Schoenwaelder, J., Mertens, S., and Klie, T. (2002). Using the Script MIB for policy-based configuration management. Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS), 2002.

McCanne, S., Leres, C., and Jacobson, V. (2002). libpcap. http://www.tcpdump.org, 2002.

Moore, B., Elleson, E., Strassner, J., and Westerinen, A. (2001). Policy core information model - version 1 specification. RFC 3060 Feb. 2001.

Nichols, K., Blake, S., Baker, F., and Black, D. (1998). Definition of the Differentiated Services Field (DS field) in the ipv4 and ipv6 headers. RFC 2474, Dec. 1998.

Oetiker, T. (1998). MRTG, Multi Router Traffic Grapher. Systems Administration Conference (LISA), 1998.

Systems, C. (2001). Cisco catalyst 6000 series network analysis software version 1.2 (new features - application response time mib). Product Bulletin No. 1398, Oct. 2001.

Tham, C., Jiang, Y., and Ko, C. (2000). Monitoring QoS distribution in multimedia networks. Wiley/ACM International Journal of Network Management, 10(2):75-90, March/April 2000.

Wahl, M., Howes, T., and Kill, S. (1997). Lightweight directory access protocol (v3). RFC 2251, Dec. 1997.

Waldbusser, S. (1997). Remote network monitoring management information base version 2 using SMIv2. RFC 2021, Jan. 1997.

Waldbusser, S. (2001). Application performance measurement mib. draft-ietf-rmonmib-apm-mib-06.txt, Feb. 2001.

Waldbusser, S., Saperia, J., and Hongal, T. (2002). Policy based management mib. draft-ietf-snmpconf-pm-10.txt, Feb. 2002.

Warth, A. and McQuaid, J. (1999). Application response time mib. draft-warth-rmon2-artmib-01.txt (expired draf), Oct. 1999.

Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., and Waldbusser, S. (2001). Terminology for policy-based management. RFC 3198 Nov. 2001.