# An Approach for Integrated Management of Networks with Quality of Service Support Using QAME

Lisandro Zambenedetti Granville,
Márcio Bartz Ceccon, Liane Margarida Rockenbach Tarouco,
Maria Janilce Bosquiroli Almeida and Alexandre da Silva Carissimi

*Federal University of Rio Grande do Sul - UFRGS*
*Institute of Informatics - II*
*Av. Bento Gonçalves, 9500 - Bloco IV*
*Porto Alegre,RS - Brazil*
{*granville, ceccon, liane, janilce, asc*}*@inf.ufrgs.br*

Providing QoS-guaranteed services in current installed networks is an important issue, but only deploying QoS services is not enough to guarantee their success: QoS management must also be provided. Nowadays, police-based network management (PBNM) addresses this need, but such management is not enough either. Network managers deal with QoS tasks that cannot be performed using only PBNM. Other solutions, besides PBNM, have to be used to proceed with QoS management-related tasks. Unfortunately, these solutions are independent from each other, leading to a scenario where integration is difficult. This paper introduces QAME (QoS-Aware Management Environment) which main goal is the provisioning of facilities to allow a common and integrated Web-based management of QoS-enabled networks.

**Keywords:** QoS management, policy-based network management, Web-based management environment

## 1 Introduction

The great majority of today's networks operate based on the IP best-effort approach. There is no warranty concerning traffic delay, jitter, throughput or lost rate. Several applications operate properly in this environment, but several others can only be delivered if network QoS (Quality of Service) warranties are present [1].

Managers should be aware of QoS features present on the network. QoS architectures can only be effective and provide guaranteed services if QoS elements are adequately configured and monitored. Thus, in addition to the management of traditional elements (routers, switches, hosts, etc.), managers must also manage QoS aspects. In this scenario, it is not a surprise to realize that the overall management will be more complex [2].

QoS management must take place within the same environment used to manage standard network elements and management platforms should be aware of QoS. A common environment is required to allow managers to proceed with QoS and standard management-related tasks in an integrated fashion [3]. Management platforms should be aware of QoS to facilitate the visualization and manipulation of QoS information, at least as easy as it is currently possible dealing with standard management information (e.g. routing tables, interface monitoring, etc.). Unfortunately, today's network management platforms are not QoS-aware and managers are forced to investigate

*O. Festor and A. Pras (Eds.) : 12th International Worshop on Distributed Systems: Operations and Management*
*DSOM'2001 Nancy France, October 15-17, 2001.*

each QoS-enabled device to check QoS parameters. Thus, a network-oriented approach that complements current device-oriented management is needed, with features that explicitly present QoS information to managers in an intuitive fashion [2].

In this context, we propose a Web-based management environment able to proceed with both standard and QoS management-related tasks. We introduce QAME (QoS-Aware Management Environment) which is based on the IETF PBNM works, and uses open standards (e.g. SNMP, LDAP, COPS, ScriptMIB) to proceed with management tasks. Managers first ask for QAME topology and QoS discovery services to map the environment to be managed. Then, policies are defined and scheduled to be used in specific devices. QAME QoS monitors can be activated to check critical network flows/aggregates in the most important network segments, while analysis and visualization features show information about experienced and expected network QoS.

Since QAME is a Web-based environment, the graphical user interface is the manager's favorite Web browser. QAME is built with PHP4 script engine that allows an extension of the environment "on the fly". New modules can be easily added throughout an upload mechanism, without having the QAME execution to be stopped. We have used XML to describe both management information and presentation data. XSL transformation, on server-side, is used to allow the development of further user interfaces, besides QAME standard user interface. Intermediate scripts are responsible for other transformations (e.g. from MIB to XML and from LDAP to XML).

The main contribution of our proposal is that QAME allows the execution of QoS management-related tasks in an integrated fashion that is absent in current management platforms. Another benefit is that QoS-related information are shown by the environment in more explicit ways, easing QoS visualization and QoS-related information treatment. Also, managers that require the incorporation of more specific features, not found in the original environment, can easily extend QAME using open standard frameworks (e.g. XML) and free software engines (e.g. PHP).

This paper is divided as follows. Section 2 discusses related work about QoS and Web-based management. Section 3 presents QAME architecture describing its components and the elements location in the managed network. QAME implementation and associated technologies are presented in section 4, while section 5 explains QAME operations throughout an example. Finally, section 6 concludes the paper and also shows future work to be considered.

## 2 Related Work

Solutions created to manage QoS in modern networks gained recent highlight mainly because of the PBNM proposals. The main players in the IP network management market issued their PBNM solutions, often integrated in their standard management platforms. Hewllett-Packard created its PolicyXpert [4] that is part of the HP OpenView management suite. Cisco also produced its PBNM solution and released, in 1999, the QPM (QoS Policy Manager) as part of the CiscoAssure policy management initiative [5]. Extreme Networks did the same and created the EPICenter (formerly the ExtremeAware Enterprize Manager - EEM) [6]. Other industry players have issued their solutions too (e.g. Nortel Networks, Lucent Technologies, Orchestream) [7]. Although these solutions can have market success, they suffer from a lack of integration from each other. Since current PBNM systems are not entirely based on IETF open standards, costumers will face serious problems when different solutions will start to be used in the same managed network [8].

In research areas that investigate QoS management some important work can be found. Imperial College London have been an active PBNM research player, with pioneer efforts, mainly through Emil Lupu and Morris Sloman works [9]. Mahon et. al. are working on the definition of general requirements for PBNM systems [10]. In their IETF work, a general architecture has been defined to guide the development of PBNM solutions. However, as Mahon states, that proposals don't address other critical aspects of QoS management, such as QoS monitoring, analysis and discovery.

Such other QoS management aspects are addressed, on the other hand, by other research projects. Hong et al. [11] proposed, in 1998, a CORBA-based management framework for managing QoS of distributed multimedia services and applications of the MAESTRO system. The

layered architecture enabled an end-to-end management of QoS provisioning resources, including services for QoS specification and mapping, admission control, negotiation and renegotiation. A generic QoS MIB was developed to access QoS parameters in such layered architecture. In 2000, we have also proposed SNMP as a mean to program DiffServ marking processes on end-systems and to allow users to reserve QoS resources asking for such resources to Bandwidth Brokers (BB) [12]. QoS monitoring has been studied by Jiang et al. [13], and Joshi et al. [14] presented, also in 2000, a solution that integrates traditional network management and QoS monitoring.

All previous proposals and solutions are important for QoS management, but they are independent from each other. In this scenario, network managers are forced to deal with several tools to manage different aspects of the whole QoS issue. Recently, we have classified QoS management-related tasks [15], trying to organize, from a network management point-of-view, the different aspects involved in QoS management. We have divided QoS management in six different tasks (installation, operation maintenance, discovery, monitoring, analysis and visualization) and have argued that an effective QoS management system is the one able to provide, in an integrated environment, facilities to proceed with the execution of such tasks. Also, we have argued that Web-based management should also be applied to QoS related tasks, in a way to allow network managers to control QoS aspects using Web widely known facilities.

Web-based management has been investigated for some years, but its first significant move occurred in 1996, when WBEM (Web-Based Enterprize Management) first came to life [16] as part of the DMTF (Distributed Management Task Force) solution. Martin-Flatin et al. have also done investigation on Web-based management and proposed the JAMAP management platform, based mainly on Java technology [17]. Using Java Applets, managers can interact with the managed network and the associated management structures.

XML technology used on network management found its way when XML was incorporated to the WBEM framework. From a different approach, John et al. presented, in 1999, XNAMI [18], that uses XML to reduce management traffic allowing the transfer of MIBs between managed devices and management stations. Unfortunately, none of the previous Web-based proposals took QoS into account directly, i.e. QoS management were not the main focus.

# 3 QAME architecture

QAME architecture (fig. 1) is divided in three different sets of elements: upper elements (which include manager Web browser, the Web-based user interface and databases), intermediate elements (policy consumers, QoS monitors and target finders) and lower elements (targets). Three different databases are part of the QAME architecture's upper elements: policy, status and associations databases.

## *3.1 Targets*

Targets are active elements that influence the final QoS observed in the network. Each device can have several targets that influence in the QoS provisioning. For example, in a router, each interface's queueing discipline is a target. Thus, targets are the final elements that effectively implement a QoS architecture.

Managers access the networks' targets indirectly throughout QAME intermediate elements (policy consumers, QoS monitors and target finders). The interface between these elements and targets depends on the devices that own the targets. Here, different protocols have to be used to access different targets on different devices. A router from vendor A, for example, can have its interfaces programmed via Telnet, while another router from vendor B requires HTTP interactions to change its interfaces settings.

## *3.2 Policy consumer*

Policy consumer is the element responsible for the installation of management policies into targets. When ordered (label 1), the policy consumer retrieves policies definitions from the policy

database (label 2). These policies are then translated into device-specific instructions to program the appropriate targets to conform the policies definitions (label 3).

After policy installation, the policy consumer is also responsible for checking the success of the policy deployment. If a policy can not be installed, either due to a failure in the target or to a lack of target capabilities, the policy consumer notifies the network manager by sending messages to the user environment (label 1).

The status database stores the deployment and operating status of each active policy associated to targets. Failures in policy deployment make the policy consumer update the status database indicating the installation problem (label 4).

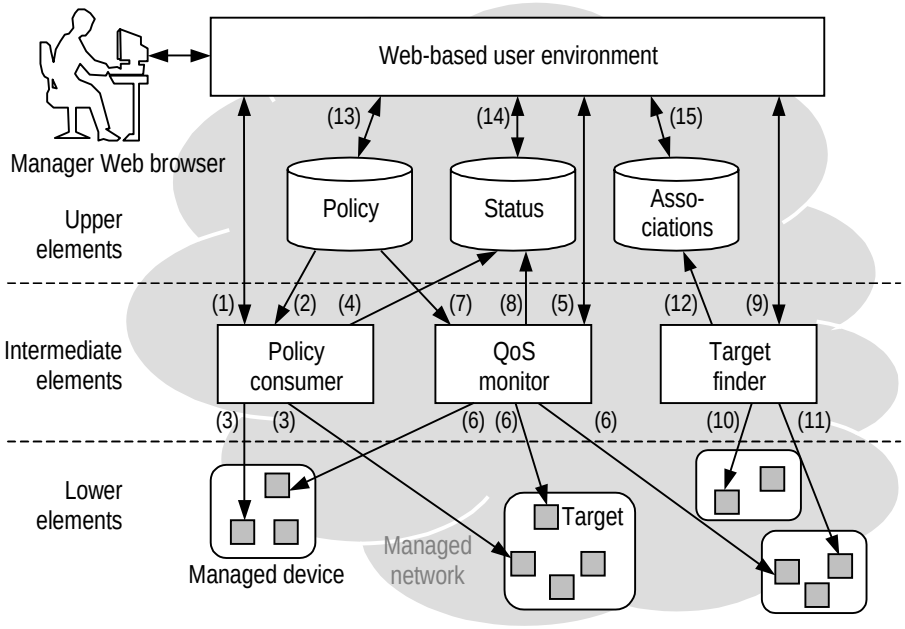


**Fig. 1:** QAME architecture

## 3.3 QoS monitor

Installed policies may not behave as stated in the policy definition. The QoS resulted from a policy installation can be different from its specification. Critical policies must then have their expected QoS monitored. The element responsible for doing that is the QoS monitor.

The network manager defines which critical segments must be checked (label 5) and QoS monitors are then activated to the check the targets that make part of those segments (label 6). QoS monitors access policy definitions also in the policy database (label 7) and compare the observed behavior of the network with the one defined in the policy. If degradation is verified, QoS monitor notifies the network manager by sending special messages to the user environment (label 5) and updates the status database to be consistent with the problem observed (label 8).

## 3.4 Target finder

In the network, searching each device to identify its targets is a time-consuming task. Also, new devices just attached to the network must have their targets identified for future programming. To do that, target finder element is the one trigged (label 9) to run QoS discovery services.

Target finders search the network for current and new targets. Each target finder recognizes at least one specific target using a specific target finding algorithm and a specific device access protocol. For example, a DiffServ target finder is the one that looks within routers and checks the

existence of packet prioritization based on the IP DS field. To do that, the DiffServ target finder can open a Telnet session (label 10) or check for target DiffServ MIB implementations (label 11).

Every discovery target is identified, classified and stored in the associations database (label 12). Target's device is also stored and relations between targets and their corresponding devices are created. Target finders are responsible for coordinating all this procedures.

### 3.5 Web-based user environment

QAME graphical user interface is implemented in the Web-based user environment, which uses Web technology to show management information. User environment is responsible for running analysis processes that complement the functionality presented in the policy consumer, QoS monitor and target finder. For example, user environment receives special messages from policy consumer telling that a policy could not be installed (label 1), and messages from QoS monitor when the observed QoS is different from the expected QoS (label 5).

User environment also interacts with the three databases in order to define their contents. Users define policies that are stored in the policy database (label 13). Policies can also be modified or removed from the database. Network manager can check the status of a deployed policy accessing the status database (label 14) and network topology is shown by accessing the associations database information (label 15).

### 3.6 Elements location

The previous subsections described each element of the QAME architecture. The present subsection explains where, in the network infrastructures, these elements are located.

Targets are located within network devices that play any active role in QoS provisioning. Examples of targets are routers and switches interfaces and their queueing disciplines. Marking, policing and traffic shaping processes are also examples of targets. Targets can be located in hosts, too. RSVP-enabled applications, or DiffServ marking processes in end systems [12] are targets, since they influence the end-to-end QoS.

Web-based user environment location is almost as obvious as target location was. We use a central point that runs QoS analysis processes and generates, via PHP4 engine, HTML pages showing the results. Databases can be located on the same device that implements Web-based user environment, or on separate devices. Since there are three databases, some can be found together with user environment, and others separately. Although figure 1 shows only one copy of each database, for security reasons we could have more copies of the same base and use database replication for security. Also, more copies of the same database would facilitate the distribution of network traffic generated by policy consumers, QoS monitors and target finders when they need to update databases information.

A trickier aspect in elements location is the location of target finders, QoS monitors and policy consumers. First of all, since they are independent elements they can be located in different places. QoS monitors are very tightly related to their targets. Thus, QoS monitors are expected to be located within the same devices that contain the monitored targets. However, depending on the installation of the QoS monitors, they can also be located close to devices, but not inside. For example, a monitor created to check the bandwidth traffic of a router interface could access the MIB-II interface group and realize that an interface is facing overflow, even though the monitor is not located within the router.

Policy consumers are often located outside devices, but modern equipments are expected to have built-in policy consumers. On the other hand, policy consumers can be located together with the user environment. Finally, target finders are often located together with user environment, acting as special plug-ins that search the network for QoS-enabled devices. Target finders can also be located in network segments other than the user environment. The less suitable location for a target finder is within devices, since devices and their targets are the objects of the finding process.

Table 1 summarizes the possible location of QAME elements.

**Tab. 1:** QAME elements location. Rows list QAME elements and columns list possible locations. Cells marked with an "x" denote that the QAME element in the row can be present in the equipment of the column. "Devices" are network equipments (routers, switches, bridges, etc.). "Proxies" are network equipment used to host some active elements that act on different equipment (e.g., a QoS monitor located within a host used to monitor a router). "Hosts" are listed to explicitly define elements located and acting in a host. Finally, "management stations" are used to denote the hosts where QAME Web-based user environment and databases are placed.

| | **Devices** | **Proxies** | **Hosts** | **Management stations** |
|---|---|---|---|---|
| Targets | x | - | x | Only if target plays active role in QoS provisioning |
| Qos Monitors | x | x | x | x |
| Policy Consumers | x | x | x | x |
| Target Finders | - | x | - | x |
| User Environment | - | - | - | x |
| Databases | - | - | - | x |

# 4 QAME technologies

In this section we will present the technologies involved in the implementation of the QAME prototype. These technologies are present in two important aspects of the QAME communications: among QAME elements and the Web-based user environment/manager Web browser communication.

## *4.1 Techonologies of the QAME elements*

As stated before, the communication with targets depends on the devices that own the managed targets. Each device can implement different ways to monitor and program its targets. For example, several commercial routers allows programming of their internal structures through Telnet (the standard old method), SNMP (for standard management platforms) and HTTP (for Web-based management). COPS protocol is expected to be found more and more frequently, supporting PBNM in modern equipments.

This diversity of access methods introduces several complexities in targets communications. QAME intermediate elements are then forced to own the following properties when communicating with targets:

- *Intermediate elements know which kind of information they are dealing with.* A policy consumer, for example, knows if an associated target supports DiffServ or IntServ. When a policy is installed, the policy consumer translates the policy definition to device-specific programming (DiffServ or IntServ). When a QoS monitor checks for performance issues, it knows if it must watch for aggregate performance (in the case of DiffServ) or flow performance (in the case of IntServ).

- *Intermediate elements know how to access targets information.* A policy consumer, for example, knows if it must use Telnet session, SNMP or HTTP to program a target. A QoS monitor knows if DiffServ information, in a specific target, are reached with COPS, SNMPv1, v2 or v3. Also, a target finder looking for targets in router from vendor A knows that those targets are found if COPS is used.

Figure 2 depicts three examples of different policy consumers acting in different devices' targets. The devices are routers from three different vendors. The targets within routers A and C are accessed via SNMP, while targets from router C are accessed via COPS. Although routers A and

B support DiffServ, they are accessed with different protocols. It means that any combination of access protocol and supported QoS mechanism requires a different policy consumer. This feature is also valid for QoS monitors and target finders. In our prototype, we have created intermediate elements for DiffServ and IntServ using SNMPv1 and Telnet, i.e we have 4 different policy consumers, 4 different QoS monitors and 4 different target finders.

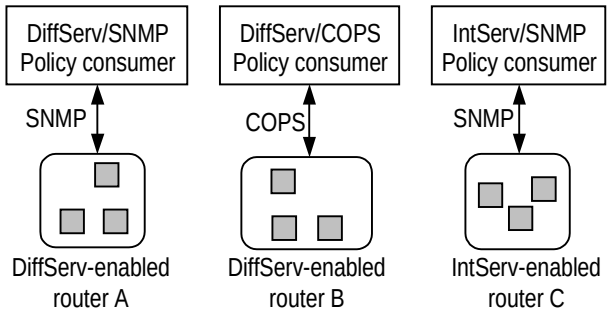


**Fig. 2:** Targets communication examples

Despite the complex communication with targets, the communication between intermediate elements and the Web-based user environment is simple. We implemented this communication using the Script MIB [19] definitions. When the network manager wants some actions to be executed in the managed network, an appropriate script is selected and sent to an intermediate element (figure 3, labels 1, 2 and 3). Each intermediate element provides facilities that allows the script to deal with the targets to be accessed and the databases to be updated or consulted. We have used the Jasmin implementation to use Script MIB definitions in QAME environment. Figure 3 shows the communication with the QAME intermediate elements and the communication with the databases.

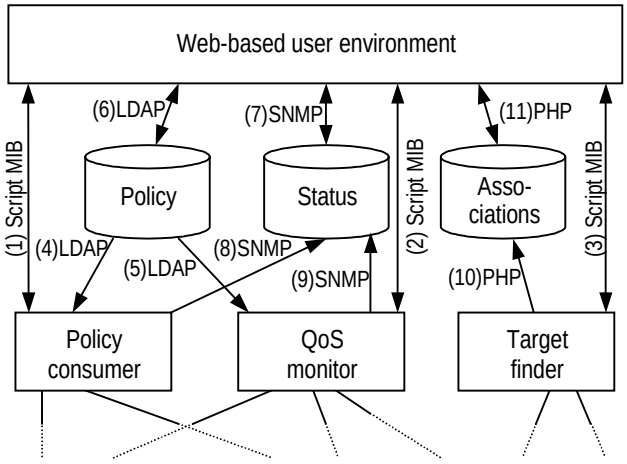


**Fig. 3:** QAME elements communication

Policy database is implemented with LDAP. We have use OpenLDAP package that includes the LDAP server and protocol API in the implementation of the policy consumers (label 4) and QoS monitors (label 5). For Web-based user environment communication the OpenLDAP API was hidden by the LDAP API provided by the PHP4 engine (label 6).

The status database interface was implemented as a MIB (label 7). We used NET-SNMP package that supports SNMPv3. A key aspect of status database communication is related to the notification messages. We used SNMP InformRequest message to update the status database because the

InformRequest works as a trap message with reply. This allows the policy consumers (label 8) or QoS monitors (label 9) to notify the status database and still be sure that the notification reached the destination.

Finally, the associations database is implemented using the MySQL solution. Thus, intermediate elements act as MySQL clients to access the needed information using PHP4 scripts (label 10). Also, Web-base user environment access the MySQL using the PHP4 (label 11). Although PHP4 has specific MySQL functions, we used the DB class that provides database abstraction, which allows an easier replacement of MySQL, if required in the future.

## *4.2 QAME Web-based technologies*

A key aspect in the QAME communication is the technologies involved in the interaction between the Web-based user environment and the manager Web browser. We have balanced the presentation overhead of the managed network between server and browser. This is done in a way to allow a richer user interaction, without introducing too much network messages exchange.

Web browser is responsible for asking network topology information for the server, and the respective presentation. We use Flash technology to build the network topology in the browser. The first advantage of using Flash is that the network traffic is reduced, since no pictures are passed from server to browser; only presentation information are exchanged. Second, Flash allows an effective user interaction, where network manager can change topology layout directly on the Web browser.

Flash itself does not provide all the facilities required in the browser interaction. JavaScript technology is then applied to complement the Flash presentation. Figure 4 shows an example of a network presented in the QAME user interface using the Flash facilities. The figure show QAME menu on the right-hand side and a topology built with Flash on the left. Each managed element has a particular set of management operations that can be applied to. This operations can be accessed through the context-menus (also shown in the figure 4) trigged when user clicks some managed device with the CTRL key pressed.

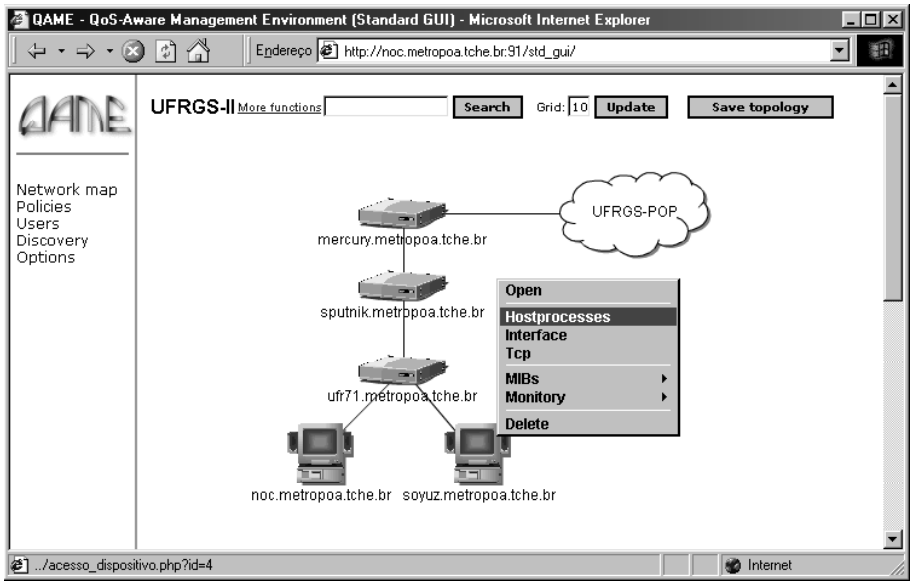


**Fig. 4:** Network topology presentation in QAME user environment

In the server-side, we use PHP4 engine, as mentioned before, and XML translations. PHP4 orchestrates several tasks, including databases access, intermediate elements communication and

user-environment graphical presentation. XML, on its turn, is used to present several information in a standard way. Table 2 shows examples of PHP4 scripts that retrieve information from different sources and produce XML results. Frequently, the source accessed does not exposes its data in XML, leaving to PHP4 the task to translate from a source representation (e.g. MIB) to XML. In the case the source already exposes its data using XML, the scripts only copy the results to the user-environment.

**Tab. 2:** Examples of PHP4 scripts that translates information from different sources to XML.

| **PHP4 Script** | **Source** | **Output** | **Description** |
|---|---|---|---|
| route.php?ip=w.x.y.z | hosts and routers | route.xml | Retrieves a devices' route table accessing the ipRouteTable objects from the 'w.x.y.z' device |
| procs.php?ip=w.x.y.z | hosts | procs.xml | Retrieves the list of processes from the host 'w.x.y.z' as specified in the HostResources MIB |
| policy.php?pol=p | policy database | policy.xml | Retrieves the policy 'p' from the policy database |
| status.php?pol=p.t.w.x.y.z | status database | status.xml | Retrieves the operational status of policy 'p' applied to the target 't' within device 'w.x.y.z' |
| targets.php?ip=w.x.y.z | association database | targets.xml | Retrieves the targets from device 'w.x.y.z' |

When the resulted XML is about to be sent to the manager Web browser, we use XSL to transform the original XML to a HTML standard page. We use Sablotron solution to proceed with the XML/XSL transformation in the server-side, allowing not XML-enabled browsers to access the retrieved information. One XML file can be differently transformed using different XSL files. With this feature we implemented, until now, three different QAME skins: advanced skin, standard skin and text only. Advanced skin produces HTML pages with several graphical elements and is preferable used when the manager operates his/her Web browser in the local network, near to the Web server. The standard skin produces fewer graphical elements and should be used when manager operates far away from the managed network. Although, if the communication between manager Web browser and QAME Web server is too congested, the text only skin can be used, since almost no graphical information is generated. The desired skin is selected when manager logins into the QAME environment.

Table 3 summarizes the technologies used in the current QAME prototype implementation.

**Tab. 3:** QAME technologies.

| **Technology** | **Interaction** |
|---|---|
| Telnet, SNMPv1 | Used to access DiffServ and IntServ-enabled devices |
| Script MIB | Script MIB is the interface of the intermediate elements |
| LDAP | Used to access the policy database |
| SNMPv2(v3) MIB | Implements the access to the status database |
| PHP + MySQL | Used to access the associations database |
| Flash + JavaScript | Dynamic topology interaction on Web browser |
| PHP + XML | Translates information from different sources to XML |
| XML + XSL | Implements the QAME skins feature |
| Sablotron | Transforms XML/XSL to HTML in server-side |

# 5 An example of QAME usage

In this section we will present an example of how QAME can be used to proceed with a simple set of management tasks. In this example we are dealing with a network composed by two connected segments that support DiffServ in their border routers (figure 5). This network is not yet mapped, an thus QAME knows nothing about the network to be managed, expect the IP addresses of its intermediate elements. Two policy consumers are used: one is implemented within the router belonging to the right-hand segment and the other is implemented by a host that controls the other router from the left-hand segment. Just one QoS monitor and one target finder are applied. The network experience hard HTTP traffic between the two segments. This hard traffic must be controlled to allow the videoconferencing to run between the hashed hosts in the figure 5.

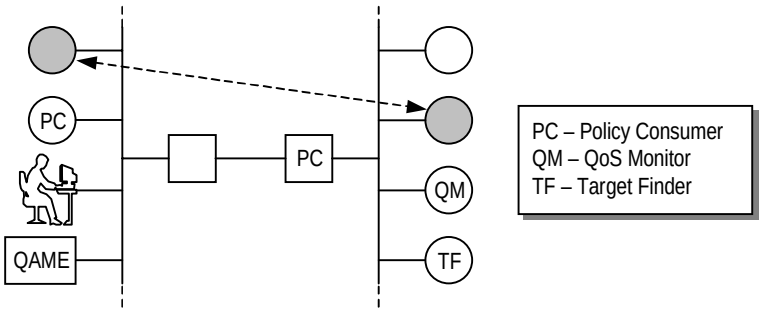


**Fig. 5:** A network example

The first step is the definition of a discovery rule. This rule is divided in topology discovery and QoS discovery. Figure 6a shows a rule defined to search the network 143.54.0.0 with netmask 255.255.0.0. We will search for DiffServ-enabled devices. The rule will be activated "now" and reactivated every 24 hours, until December 24, 2003. More specific discovery parameters, although not shown in figure 6a, can also be specified (for example, the number of ICMP retransmission in case of errors and the timeout of ICMP replies).

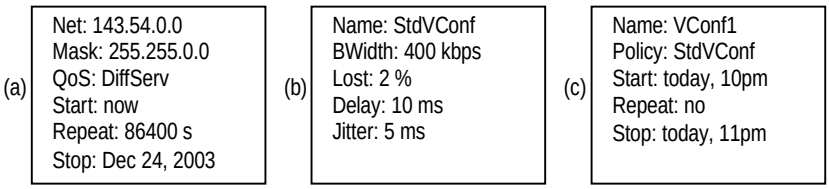


**Fig. 6:** Examples of discovery rule, policy definition and deployment

QAME transfers the discovery rule to the appropriate target finder based on the IP address, netmask and QoS mechanism specified in the rule, although in our example there is no choice since just one DiffServ target finder exists. The rule is kept in the target finder until its end (December 24, 2003) and started just after its transfer. The target finder than first discover the network topology using ICMP messages. The just discovered devices are then investigated, in our current prototype via SNMP, to determine its DiffServ capabilities. Then, target finder stores the discovered devices and the associated targets in the associations database that is located together with QAME. Visually, network manager can perceives the evolution of the discovery by looking at the Flash topology built in the user browser. Since Flash updates its presentation periodically, new discovery devices are shown in next updates.

The next step requires the manager to determine some relationships among QAME elements and the managed network. The QoS monitor, from the right-hand segment, has to be associated to the right router. The policy consumer from the left-hand segment has to be associated to the

left router. The policy consumer from the right-hand router is automatically associated to the router by the target finder. All associations are then stored in the associations database. After that, network is ready to be programmed.

Policies must be defined. Figure 6b shows the definition of a simple policy used to guarantee enough network resources to the videoconferencing we are dealing with. This policy is stored in the policy database for future use, and indexed by its name. Let's suppose the videoconferencing should gain network resource for only one hour beginning at 10am. Figure 6c shows the deployment rule for this policy stored in the status database and indexed by its name too.

To deploy the policy on the network, manager consults QAME topology facilities to determine the routers in the path from one videoconferencing host to the other. This routers are highlighted on the topology presentation and manager is able to deploy the policy. Internally, QAME consults the associations database to retrieve the IP addresses of associated policy consumers for each router. Finally, policy is sent to these consumers that translate the policy and program the network devices when the policy is activated.

To watch the experienced QoS in the network, manager again consults QAME topology facilities to determine which routers can be monitored. In this case, only the router from right-hand segment has a QoS monitor associated. The policy definition and the deployment rule are then accessed by the QoS monitor to verify when monitoring should begin.

## 6 Conclusion and future work

In this paper we have argued that QoS provisioning architectures can deploy guaranteed services only if such architectures are managed by QoS management processes. In the related work section we saw that several aspects involved in QoS management are addressed by different solutions, leading to a scenario where managers are forced to deal with different tools.

The presented QAME solution integrates, in the same environment, QoS management-related tasks that were separately found before. An advantage of this integration can be observed, for example, in policy deployment followed by QoS monitoring. In a standard environment, managers could use HP PolicyXpert to define a policy and deploy it. Then, using a separated monitoring tool the managers proceed with another definition to determine which flows have to be observed. In QAME, however, only one definition is necessary to program the network and to observe the network behavior, since policy consumers and QoS monitors use the same policy definition and deployment rule to proceed with their tasks.

Another important aspect of QAME is that, even though it is a Web-based management environment, the network manager is not limited by static network bitmaps or heavy loaded Java applets. We combined Flash and JavaScript to provide, with low bandwidth usage, levels of user interaction that are normally found only in standard not Web-based platforms.

Future works are related to the definition, in a higher abstract language, the policies to be applied. Today, policy definition still requires several details that could be abstracted from the network manager. More investigation is also required in the topology representation of very large networks (more then 400 devices) using the JavaScript/Flash solution. We have used QAME to manage small (less then 100 devices) and medium networks (between 100 and 400 devices). Its behavior have shown that QAME integration of QoS management facilities eases, in that networks, the management. On very large networks, on the other hand, the amount of information represents a very challenger management task.

## References

[1] Stardust.com, Inc.: The Need for QoS. White paper. Available at: <http://www.qosforum.com/white-paper/Nedd_for_QoSv4.pdf>. QoS Forum (1999)

[2] Eder, M., Nag, S.: Service Management Architecture Issues and Review. RFC 3052. IETF (Jan. 2001)

[3] Huston, G.: Next Steps for the IP QoS Architectures. RFC 2990. IETF (Nov. 2000)

[4] Hewllett-Packard: HP OpenView PolicyXpert. Homepage. Available at: <http://www.openview.hp.com/products/policyexpert/>. Hewllett-Packard Company (2001)

[5] Cisco Systems: Cisco QoS Policy Manager (QPM). Homepage. Available at: <http://www.cisco.com/warp/public/cc/pd/wr2k/qoppmn/>. Cisco Systems (2001)

[6] Extreme Networks: EPICenter 3.0 Technical Specification. Available at: <http://www.extremenetworks.com/products/prod_pdf/EPICenter.pdf>. Extreme Networks (2001)

[7] Saunders, S.: The Policy Makers. Data Communications Magazine (May 1999) 34-56

[8] Clark, R.: The Mechanics of Policy-Based Management. Network Magazine (Mar. 2000) 44-51

[9] Sloman, M.: Policy Driven Management for Distributed Systems. Journal of Network and Systems Management, Vol. 2, Plenum Publishing (Dec. 1994) 333-360

[10] Mahon, H., Bernet, Y., Herzog, S., Schnizlein, J.: Requirements for a Policy Management System. Internet draft <draft-ietf-policy-req-02.txt>. Work in progress. IETF (Nov. 2000)

[11] Hong, J.W.K., Kim, J.S., Park, J.K.: A CORBA-Based Quality of Service Management Framework for Distributed Multimedia Services and Applications. IEEE Network, Vol. 13, No. 2, (1999) 70-79

[12] Granville, L.Z., Fleischmann, R.U., Tarouco, L.M.R., Almeida, M.J.B.: Managing Differentiated Services QoS in End Systems using SNMP. In Proceedings of the 2000 IEEE Workshop on IP-oriented Operations & Management (IPOM 2000). Cracow, Poland (2000) 191-198

[13] Jiang, Y., Tham, C.K., Ko, C.C.: Providing Quality of Service Monitoring: Challenges and Approaches. In Proceedings of the 2000 IEEE/IFIP Network Operations and Management Symposium (NOMS 2000). Honolulu, USA (2000) 115-128

[14] Joshi, R., Tham, C.K.: Integrated Quality of Service and Network Management. In Proceedings of the 2000 IEEE International Conference on Networks (ICON 2000). Singapore (2000) 497

[15] Granville, L.Z., Tarouco, L.M.R.: An Environment to Support QoS Management-Related Tasks on IP Networks. In Proceedings of the 2001 IEEE International Conference on Telecommunications (ICT 2001). Bucharest, Romania (2001)

[16] Distributed Management Task Force. WBEM Initiative. Homepage. Available at: <http://www.dmtf.org/wbem/>.

[17] Martin-Flatin, J.P., Bovet, L., Hubaux, J.P.: JAMAP: a Web-Based Management Platform for IP Networks. In Proceedings of the 10th IFIP/IEEE Internation Workshop on Distributed Systems: Operations & Management (DSOM'99). Zurich, Switzerland. Lecture Notes in Computer Science, Vol. 1700. Springer-Verlag, Berlin Heidelberg New York (1999) 164-178

[18] John, A., Vanderveen, K., Sugla, B.: An XML-based Framework for Dynamic SNMP MIB Extension. In Proceedings of the 10th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM'99). Zurich, Switzerland. Lecture Notes in Computer Science, Vol. 1700. Springer-Verlag, Berlin Heidelberg New Your (1999) 107-120

[19] Schoenwaelder, J., Quittek, J., Kappler, C.: Building Distributed Management Applications with the IETF Script MIB. IEEE Journal on Selected Areas in Communications, Special Issue on Network Management and Operations, (2000) Vol. 18, Number 5