QoS Management-Related Tasks: Beyond Policy-Based Management

Lisandro Zambenedetti Granville, Liane Margarida Rockenbach Tarouco,
Márcio Bartz Ceccon, Maria Janilce Bosquiroli Almeida

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

Abstract. 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, policy-based network management (PBNM) addresses this need, but such management is not enough either. Network managers often deal with QoS tasks that cannot be performed using only PBNM. This paper describes six important QoS management-related tasks (QoS installation, operation maintenance, discovery, monitoring, analysis and visualization) and shows solutions that can help managers proceed with these tasks. Unfortunately, these solutions are independent from each other, leading to a scenario where integration is difficult. To solve this lack of integration, QAME (QoS-Aware Management Environment) is proposed. QAME provides support to allow the execution of the defined QoS tasks in an integrated fashion.

Keywords. QoS management, policy-based network management, management environment

1 Introduction

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

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 (router, switches, hosts, etc.), managers must also manage QoS aspects. In this scenario, it's not a surprise to realize that the overall management will be more complex.

Effective QoS management can only be achieved if QoS management tasks are properly identified. Based on these tasks, management mechanisms can be defined to help managers to deal with QoS elements. Besides, such mechanisms must allow the replacement of the current device-oriented management approach by a network-oriented approach. This approach replacement is necessary because the amount of management information is even greater when QoS-related data are present [1]. Device-specific management takes too long and does not provide a global view, which is specially needed when monitoring QoS parameters in critical flows.

QoS management must also take place within the same environment used to manage standard network elements. Management platforms should be aware of QoS to help managers

proceed with QoS-related tasks. Unfortunately, today's management platforms are not QoS-aware and managers have to investigate each QoS-enabled device to check QoS parameters. Again, a network-oriented approach is needed, with features that explicitly present QoS information to managers in an intuitive fashion.

This work presents our ongoing QoS management project named QAME (QoS-Aware Management Environment). The paper shows current analysis on QoS tasks and related software developed. The final goal is to provide an environment where managers are able to deal with explicit QoS information in a higher abstraction level, and with a global network view, implementing the network-oriented approach mentioned before.

This paper is divided as follows. Section 2 presents QoS-related tasks that should be performed to allow the deployment and maintenance of QoS architectures and section 3 introduces our proposed environment. Section 4 concludes the paper and also shows future work to be developed.

2 QoS Management-Related Tasks

QoS-enabled networks can only present reliable QoS services if managers are aware of QoS. Each network can use one or more QoS solutions and protocols. For instance, one could use MPLS [2] within an administrative domain, and have agreements with neighboring domains using DiffServ [3]. Another could use RSVP [4] to dynamically reserve network resources for a scheduled videoconference. Every QoS-related element (routers, brokers, hosts, classifiers, markers, etc.) must be managed to guarantee proper QoS service operation.

In order to manage QoS elements, managers must perform several tasks, besides the tasks applied to traditional management. QoS-related tasks must be performed by using facilities provided by the network management environment, but today's platforms do not provide any explicit QoS facility. To start providing such facilities, firstly we need to identify QoS tasks and then check how facilities should be created to help QoS management. Thus, we have classified QoS management-related tasks as described in the following subsections.

2.1 QoS Installation

We call "QoS installation" the task of choosing QoS solutions and installing such solutions into a network. The result of QoS installation is a unique QoS architecture that provides QoS services to a particular network. The resulted architecture is a collection of QoS solutions and protocols that, together, offer services to network users.

Nowadays, the main QoS solutions offered are those supported by the IETF (e.g. MPLS, DiffServ and IntServ [5]). Actually, each solution can be deployed by using different configurations. For instance, one could use bandwidth brokers [6] to allow dynamic DiffServ resource reservation. Another could use DiffServ with no brokers installed on the network.

There is a need for software that advises managers about possible QoS solutions, configurations and problems. One solution can be appropriate for a particular network, but totally inappropriate for another network. The final architecture must result from the analysis of the following elements:

- **Network topology**. Each network has a particular topology and for each topology a particular architecture is more suitable;
- **Network traffic**. Even with identical topologies, each network faces different traffic. One could have more downstream than upstream traffic, or more internal than external traffic;

- **Network application priorities**. Network traffic is mainly generated by applications (network itself doesn't generate too much traffic). The level of desired application priorities vary from network to network;
- **Network available solutions**. The final architecture can only be applied if selected solutions are supported by network devices. Otherwise, the final architecture has to be computed again.

Deriving the final architecture could be helped by QoS simulation solutions. Managers would describe its network topology, traffic and application priorities, and start checking each available solution. Different solutions could exist in the same environment at the same time, to collaborate with each other. For instance, a manager could use RSVP reservation internally and DiffServ on the network boundaries.

After deciding on a final architecture to be used, managers should start configuring devices, changing inappropriate ones, updating software on others, etc. Such procedures can be done in two different ways: locally or remotely. Local procedures are time consuming. Thus, we must provide tools able to keep local procedures minimal. Configuring and updating software can be remotely done on almost all cases. Except for a few operations, configuration is remotely driven, mainly through Telnet command line interface, HTTP in modern devices, and by using SNMP [7] agent/manager interaction. Changing equipment is an intrinsically local operation, and cannot be done remotely.

2.2 QoS Operation Maintenance

After QoS architecture is defined and installed, QoS services are ready to be offered to network users. At the same time QoS architecture is serving user needs, the manager must define appropriate operational parameters. We qualify any procedure taken to define QoS service behavior "on the fly" as "QoS operation maintenance".

Procedures often taken in QoS operation maintenance are those related to traffic classification, marking and prioritization. Bandwidth static reservation, SLAs management [8] and policing are also examples of operation maintenance.

Today, the promising solution in QoS operation maintenance is policy-based network management (PBNM). By using policies, managers can determine, at a higher abstraction level, how the QoS architecture should proceed to meet a desired behavior. PBNM actually constitutes a whole architecture itself, with dedicated elements (e.g. PEPs - Policy Enforcement Points and PDPs - Policy Decision Points [9]). One important procedure in PBNM is, for example, the location of Policy Enforcement Points. Tools should also help managers in defining such points.

2.3 QoS Discovery

Some network equipments are overfilled with features. It is not rare to see complex routers, with several services, being used only for packet forwarding. Although several features could be used to help QoS provisioning, they are not, since managers cannot handle so many device features in large and complex networks.

We define "QoS discovery" as the task of searching the network for features that can help or improve QoS provisioning. Normally, QoS discovery is done through SNMP messages sent to devices, or by using proprietary methods. QoS discovery can also be performed checking equipment documentation, but that is not an operation that could be automated.

QoS discovery is helpful in two important moments: in QoS installation and in QoS operation maintenance. In QoS installation the new discovered features can be used to decide among QoS architectures. In QoS operation maintenance, discovery can report new added equipment with QoS features while the QoS architecture is running.

QoS discovery works only if discovering mechanisms are installed. Such mechanisms can be simple (polling devices trying to find MIBs related to QoS architectures) or complex (distributed network monitoring to check traces of critical protocols, such as IGMP [10] and RSVP [4]).

2.4 QoS Monitoring

Managers have to be up-to-date about the difference between the desired QoS and the faced QoS. This difference can never be greater than a defined critical value. If the faced QoS is too different from the desired QoS, user applications would degrade indicating that the contracted services are not running properly.

To check the current QoS, managers should collect data in the network through QoS monitoring. This task has to be able to determine two related pieces of information:

- **End-to-end achieved QoS**. The QoS parameters of particular segments are important, but end-to-end QoS is crucial. If end-to-end achieved QoS is not correct, managers must be warned to fix the problem. End-to-end QoS degradation is the sum of the degradation of each segment on the end-to-end path. If just one segment is introducing degradation, that will be noticed in the end-to-end QoS.
- **Points of degradation**. If end-to-end QoS degradation is noticed, QoS monitoring should be able to determine where in a flow path the degradation points are.

Today, most QoS monitoring solutions are only able to satisfy the first item. It is simple to check if there is end-to-end degradation, by using RTP/RTCP [11] protocols, for example. However, identifying degradation points is a more complex procedure, and requires more complex processing on the network [12].

2.5 QoS Analysis

In a proactive management, managers should anticipate future problems and attack them as soon as possible. To achieve proactive QoS management, QoS analysis tasks must be performed.

Cataloged historical behavior can show, for example, the number of refused RSVP sessions due to lack of resources. If the number of refused sessions increases too much, it indicates that the manager should update network resources. Analysis of a QoS-dependent monitored client/server operation could show the frequency of QoS degradation, and the frequent degradation points. In this case, the manager should check the critical points to see link problems.

One crucial part of QoS analysis is QoS visualization. We consider QoS visualization such an important procedure that it is defined separately from QoS analysis.

2.6 QoS Visualization

Today's network management platforms are topology-oriented, i.e., they show information from a network topological perspective. Managers browse network topology and check each

desired device. Some facilities can be found, allowing managers, for example, to ask the platform for a map listing every printer.

For QoS management tasks, current visualization is poor. Managers often search for each important device in maps, and check to see if such device is QoS-enabled. This is a time-consuming task, and should be replaced with an automated search procedure.

QoS visualization is a task that helps managers to proceed with other QoS tasks. Tools should provide visualization based on QoS characteristics. We list here helpful QoS visualizations examples.

- **Colored link utilization**. Each link shows, instead of a single black connecting line, a colored set of lines describing link utilization by each flow/aggregate;
- **Selected end-to-end sessions**. A topology map shows end-to-end sessions highlighted, through colored lines. Managers could ask to visualize only sessions that match some pre-determined features;
- **QoS enabled devices**. A topology map highlights QoS-enabled devices. Different colors show devices that implement different solutions. For example, green boxes indicate DiffServ-enabled routers, whereas red boxes indicate RSVP-enabled ones;
- **Segments with QoS degradation**. Segments with QoS degradation are shown in red or are highlighted, to indicate degradation. Orange segments indicate probable degradation coming.

Several other visualization facilities could be created to help managers identify QoS-related information. Today, QoS visualization can only be found on separate software that has no integration. The next section shows some current solutions used in QoS visualization and other tasks.

3 QAME

Previous sections presented QoS-related management tasks and some solutions that help managers perform such tasks. Each solution provides functionalities to attack a particular problem, but that is done independently from other solutions. Thus, there is not any QoS task integration.

In addition to the lack of integration, current network management platforms are device-oriented and not QoS-aware, i.e., even if they have QoS support they do not show QoS information properly. A more appropriate environment should allow a network-oriented view to the management, also allowing explicit QoS information.

This section presents a QoS-integrated management environment, called QAME (QoS-Aware Management Environment). The environment is QoS-aware in the sense that it takes QoS information explicitly and shows that information more properly. Besides, QAME provides support for managers to proceed with the six QoS tasks previously defined (QoS installation, operation maintenance, discovery, monitoring, analysis and visualization).

3.1 QAME Architecture

QAME architecture extends the policy-based solution defined by [13] introducing some new elements. The architecture is initially divided into active elements and databases. Active elements perform management and QoS provisioning tasks. They also store, retrieve and update information in databases. Active elements are sub-divided in an upper element (the User Environment), intermediate elements (Policy Consumer, QoS Monitor and Target

Finder) and lower elements (Targets). Figure 1 shows QAME elements and databases, which, on their turn, are presented in the following subsections.

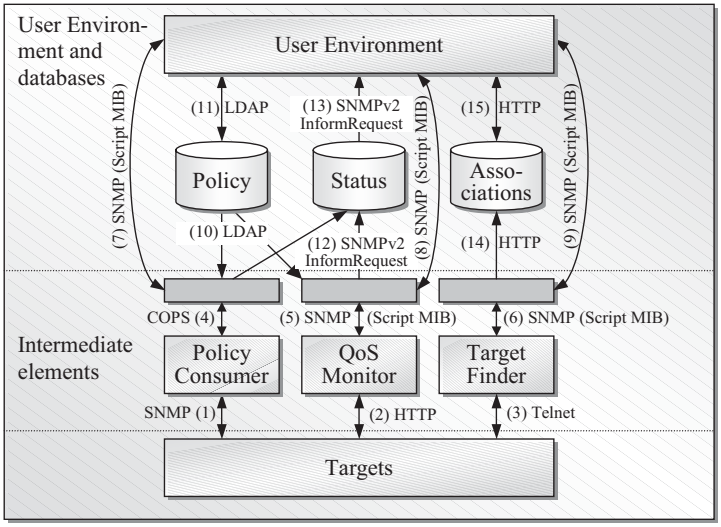

Fig. 1. QAME architecture

Targets

Targets are active elements that influence the final QoS observed in the network. Each device can have several Targets that can influence the network parameters. For example, in a router each interface is a Target. Targets are the final elements that effectively implement a QoS architecture.

Network manager has indirect access to Targets throughout Policy Consumer, QoS Monitor and Target Finder elements. The interface between these elements and Targets is device-specific, and different protocols are used to access different Targets. A router from vendor A, for instance, can be accessed via a Telnet client, while another router from vendor B can be accessed via HTTP.

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 cataloged for use. Finally, if QoS discovery is an important task, automatic Target finding is necessary.

Target Finders are special agents that search the network for current and new Targets. Each Target Finder recognizes at least one specific Target, using a specific Target finding algorithm. 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 or check for Target DiffServ MIB implementations.

Since each Target can have different capabilities, Target Finders are also responsible for classifying new discovered Targets and storing any Target information on the Associations database.

Policy Consumer
Policy Consumer is responsible for installing a policy into Targets. Each Policy Consumer, when ordered to install a policy, retrieves the policy by accessing the Policy database. The new policy is then translated into device-specific instructions to program Target to conform that policy.

After policy installation, Policy Consumer is also responsible for checking the success of the policy in the Targets. If a policy could not be installed either due to failure in the Target or to lack of Target capabilities, the Policy Consumer notifies the network manager by sending messages to the User Environment.

A Policy Consumer can install policies in several Targets at the same time. On the other hand, each Target can be associated to several Policy Consumers, even though only one can be the current active consumer.

QoS Monitor
Installed policies might 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 policies must be checked and QoS Monitors are then associated to the Targets that implement those policies. QoS Monitors access policy definitions also in the Policy database and compare the effective behavior on the network with the one defined in the policy. If degradation is observed, QoS Monitor notifies the network manager by sending special messages to the User Environment, too.

The greater the number of QoS Monitors used, the more accurate the monitoring process will be. Also, the greater the number of QoS Monitors used, the greater the information analysis overhead will be.

User Environment
QAME graphic user interface is implemented in the 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, Target Finder and QoS Monitor. For example, User Environment receives special messages from Policy Consumer telling a policy could not be installed, and messages from QoS Monitor when the observed QoS is different from the expected QoS.

User Environment also interacts with the three databases in order to define their contents. Users define policies that are stored in the Policy database by using the environment interface. Policies can also be modified or removed from the database. Network topology is shown by accessing the Associations database information. Users check network topology on their Web browsers and order actions to be executed in Targets.

Databases
The three databases shown in figure 1 are defined to store different classes of information. Every Target, Policy Consumer, QoS Monitor and Target Finder is registered in the Associations database. With appropriate searching of the base, the User Environment can derive the network topology, existing QoS solutions, and resource capabilities. Furthermore, the Associations database stores the associations between Targets and the other elements. For example, a Target named A uses Policy Consumer B for policy translation and installation, QoS Monitor B for policy performance checking and Target Finder C for discovering possible new capabilities in Target A.

The policies are stored in the Policy database. Since policies themselves do not define on which Targets they should be installed, policies can be stored separately from Targets. We do that because the database needed for Targets and the database needed for policies have different requirements. Policies once defined have little or no change. On the other hand, Targets can have their capabilities extended, associations updated, and have more changeable data overall.

Even more changeable are data used to represent the status of a deployed and/or monitored policy. QoS Monitors and Policy Consumers change data in the Status database every time a deployed policy has its status altered. Thus, Status database binds information stored in the Policy database (the policies) with that stored in the Associations database (the Targets), and introduces new information about this relationship: the status of a policy.

3.2 Elements location

The previous subsection described each element of the QAME architecture. This present subsection explains where in the network infrastructures these elements are located and how many elements of the same type can be used.

The more obvious element location is the Target location. Targets are located within network devices that play any active role in QoS provisioning. Target examples are routers and switch interfaces in their queuing 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 [14] are Targets, since they influence the end-to-end QoS.

We leave intermediate elements location for the next paragraphs, since this is a more complicated issue. On the other hand, User Environment location is almost as obvious as Target location was. We use a central point that runs QoS analysis processes and generates HTML pages showing the results. The central point could generate too much traffic on larger networks. In this case, a distributed User Environment can be used since databases are detached from the User Environment. Several environments would access the same databases.

Databases can be located on the same device that implements 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 QoS Monitors, Policy Consumers and Target Finders when they need to update databases information.

A trickier aspect 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, it is expected that QoS Monitors are 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 consumer implementations. On the other hand, Policy Consumers can be located together with the User Environment, thus improving User Environment and Policy Consumer communication.

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 segment. They would act as segment monitors, looking for QoS traffic generated by QoS-enabled devices. The less suitable location for a Target Finder is within devices, since devices and their Targets are the objects of the finding process. One exception is when devices are able to announce themselves and their capabilities to the network.

One important issue about location is the management interface between the User Environment and Policy Consumer, QoS Monitor and Target Finder. This interface is depicted in figure 1 by the gray rectangles connecting User Environment and intermediate elements. These interfaces can be found together with the elements if the elements implement such interfaces. Otherwise, the interfaces are located in the User Environment and translate requests into element-specific commands. This separation between element implementation and interface is important because modern devices could implement elements with interfaces different from those used by QAME. In this case an interface translation is needed to allow the use of built-in device elements. Table 1 summarizes the possible location of QAME elements.

	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

Table 1. QAME elements location. Rows list QAME elements and colums list possible locations. Cells marked with an "x" denote that the QAME element in the row can be present in the equipment of the colum. "Devices" are network equipment (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 User Environment and databases are placed.

3.3 Protocols

This subsection describes the protocols used to provide communication between QAME elements. In the protocol definition phase of the project, we decided to use standard and open protocols to implement such communication. Even though some standard protocols might not be the best choice for some critical tasks, we believe that this choice makes the architecture open, making future implementation of new modules easier.

Targets Protocols

The protocols used to communicate with Targets are actually defined by the devices that contain the Targets. These protocols are then dependent on the device's provider, which could choose to use standard protocols or implement its own proprietary protocol.

The most common Targets protocols available are Telnet and SNMP, but modern devices also use HTTP for configuration management (figure 1, labels 1, 2 and 3). Fortunately, proprietary protocols are more rare.

Despite the diversity of possibilities, Targets protocols are not a critical issue since intermediate elements are responsible for protocol translations. Thus, when two different routers, using different protocols, should be programmed to prioritize a defined flow, that programming task "sees" the different routers as equal because of the protocol translation executed in the Policy Consumer. This translation also occurs in the QoS Monitor and Target Finder elements (figure 1, labels 1 and 4, 2 and 5, 3 and 6).

QoS Monitors, Policy Consumers and Target Finders Protocols

Protocols used to communicate with intermediate elements can be divided into two groups: those implemented by the elements, and those used as interface with elements (gray rectangles in figure 1). Protocols implemented by the elements are element-dependent and defined by the element developer. For example, a vendor implementing a Policy Consumer within its new router could use COPS [15] to transfer policies. Another vendor could choose Script MIB [16] definitions to have access to the policies (figure 1, labels 4, 5 and 6).

On the other hand, protocols used in the access interface are always the same. This is a requirement to allow access to the same interface in the User Environment. Thus, interface actually implements only protocol translation, from User Environment interface access into the intermediate elements-specific interface (figure 1, labels 4 and 7, 5 and 8, 6 and 9). The current QAME implementation uses Script MIB to communicate with intermediate elements interface.

Databases Protocols

The protocols used to access database information are different because the nature of each base is different. Policy database is reached using the LDAP [17] protocol (figure 1, labels 10 and 11). Since policies are information that has few updates but can be accessed several times, a write-once read-several times protocol like LDAP is more suitable.

Status and association information are more dynamic, and LDAP protocol should be avoided. Thus, SNMPv2 InformRequest messages are used to update status information in the Status database (figure 1, label 12). InformRequest messages can be faced as an SNMPv1 trap message with confirmation. QoS Monitor perceiving QoS degradation can update the status of a monitored flow or aggregate trapping the Status database, and still have confirmation of the update operation due to the reply of the InformRequest message. Policy Consumers can also notify the Status database when a policy deployment fails. The InformRequest message can be forwarded to the User Environment when critical monitoring

tasks are performed (figure 1, label 13). Finally, User Environment and Target Finders reach Associations database through HTTP and queries to a PHP4 engine [18] (figure 1, labels 14 and 15).

4 Conclusions and Future Work

Providing QoS services in networks is currently very important because time-dependent application cannot be deployed using best-effort based services. QoS architectures must be installed, but should also be properly managed to be effective. Traditional network management cannot be applied to QoS management because the amount of available information is much larger and complex.

Current efforts try to find a way to allow complexity abstraction by using policy-based network management (PBNM). But policies are not sufficient to allow total QoS management. Other QoS management-related tasks are performed by network managers, and support should be available for those tasks.

This paper has defined six main QoS management related tasks: QoS installation, operation maintenance, discovery, monitoring, analysis and visualization. PBNM only addresses QoS operation maintenance. Other tasks have no advantage of PBNM. For these tasks we have presented current solutions that help managers install them. Unfortunately, there is no integration between solutions, and managers often have to deal with too many tools, increasing the management complexity.

To make management easier we have presented the QoS management environment called QAME where the six important QoS tasks can be performed in an integrated fashion. We have described the QAME architecture, its elements, and where these elements can be located in the network (e.g., within routers, switches, hosts, or integrated in the management station). QAME elements exchange information to allow integration. This information exchange is done by communication protocols. We have shown that the QAME environment uses LDAP, SNMPv2, Script MIB and proprietary protocols to implement communications.

Protocol translation is a required feature to allow better interaction between user environment (where management information is presented) and lower-level elements (where management information is gathered and processed). The protocol translation is possible because QAME architecture detaches lower-level elements implementation from their interface. The element implementation can be located, for example, in a router, while the element interface can be located in the management station.

Future work may address security, database replication and distributed management issues. Since SNMP messages are not encrypted, the use of SNMP version 3 is a natural choice. Database replication deserves a more accurate research because single database instances could be inadequate to manage very large networks. Because management traffic will be greater, database access at a single point could prevent better management performance. Also, in larger networks, a central point of management should be avoided, and distributed management with more than one User Environment would be preferable.

References

1. Eder, M.: Service Management Architecture Issues and Review. RFC 3052 (2001)
2. Rosen, E et al.: Multiprotocol Label Switching Architecture. RFC 3031 (2001)
3. Blake, S. et al.: An Architecture for Differentiated Services. RFC 2475 (1998)
4. Braden, R. et al.: Resource ReSerVation Protocol (RSVP). RFC 2205 (1997)
5. Shenker, S., Wroclawski, J.: General Characterization Parameters for Integrated Service Network Elements. RFC 2215 (1997)
6. Nichols, K. et al.: A Two-bit Differentiated Services Architecture for the Internet. RFC 2638 (1999)
7. Case, J. et al.: A Simple Network Management Protocol (SNMP). STD 15, RFC 1157 (1992)
8. McBride. D.: The SLA Cookbook: A Recipe for Understanding System and Network Resource Demands. Hewlett-Packard Company. Available at http://www.hp.com/openview/rpm (1996)
9. Yavatkar, R. et al.: A Framework for Policy-Based Admission Control. RFC 2753 (2000)
10. Fenner, W.: Internet Group Management Protocol, Version 2. RFC 2236 (1997)

11.Schulzrinne, H. et al.: RTP: A Transport Protocol for Real-Time Applications. RFC 1889 (1996)
12.Jiang, Y., Tham, C.K., Ko, C.C.: Providing Quality of Service Monitoring: Challenges and Approaches. NOMS 2000 - IEEE/IFIP Network Operations and Management Seminar (2000)
13.Mahon, H. et al.: Requirements for a Policy Management System. Internet draft <draft-ietf-policy-req-02.txt>. Work in progress (2000)
14.Granville, L. et al.: Managing Differentiated Services QoS in End Systems using SNMP. IPOM 2000 - IEEE Workshop on IP-oriented Operations & Management (2000)
15.Chan, K. et al.: COPS Usage for Policy Provisioning (COPS-PR). RFC 3084 (2001)
16.Quittek, J., Kappler, C.: Remote Service Deployment on Programmable Switches with the IETF SNMP Script MIB. DSOM 1999 IFIP/IEEE International Workshop on Distributed Systems: Operations and Management. Springer Verlag (1999)
17.Arlein, R., Freire, J., Gehani, N., Lieuwen, D., Ordille, J.: Making LDAP active with the LTAP gateway: Case study in providing telecom integration and enhanced services. In Proc. Workshop on Databases in Telecommunication (1999)
18.PHP Hypertext Preprocessor. Available via WWW at URL: http://www.php.net (2001)