Network Executive: A Policy-Based Network Management Tool

Gustavo Augusto Faraco de Sá Coelho, 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
Telephone: +55.51.3316.6235
{gcoelho, granville, janilce, liane}@inf.ufrgs.br

Abstract. The development of new applications that require QoS support makes computer networks even more complex and heterogeneous. Current network management is not capable of meeting the new needs network managers have. This is the scenario where Policy-Based Network Management (PBNM) emerges as an efficient alternative to warrant the operation of QoS services. However, the existing commercial tools do not meet the IETF standards completely and do not have suitable interoperability functions. Therefore, this article presents a solution developed according to IETF concepts that meets this need for new tools capable of managing the new upcoming network services.

Keywords: *policy-based network management, QoS management, network management.*

1. Introduction

Nowadays, computer networks can be found practically everywhere and their complexity and dimensions vary a lot, from small local networks to large metropolitan and national networks. Although the usual traffic on such networks used to be only the transmission of data, today's traffic is rather heterogeneous. Current applications, such as voice over IP (VoIP), videoconferencing, enterprise resource planning (ERP), mission-critical systems, and others, demand more appropriate services from the networks. Even though the IP protocol best-effort approach was once enough, today's network services need to make sure new applications run properly. This is when QoS (Quality of Service) parameters become necessary in order to ensure throughput, delay, jitter and lost rate limits. Therefore, in order to support new applications, IP networks must migrate from the best-effort paradigm to a QoS-enabled paradigm.

As on one hand such new applications are an appealing option for the final user and generate new business opportunities, on the other hand they present new challenges that network managers face to run operations properly. Implementing these new applications and

adding new mechanisms and techniques to support them make the network a heterogeneous, complex and consequently more difficult to manage environment.

From that standpoint, traditional management is no longer enough. The amount of management information generated by a QoS-enabled network is so much that even if the manager can obtain all that information, it will not be possible to deal with such amount. Policy-Based Network Management (PBNM) [1] helps manage complex networks. Through the concept of policies, the network manager can define how the network should operate by using a more abstract language than the one used in traditional management. It should be noted that managers will only inform "what" is expected, but not "how" it should be obtained.

Because it is relatively new technology, the existing PBNM tools available on the market differ greatly concerning protocols and data schemas, which makes interoperability harder [2]. As opposed to the traditional management applications, which use SNMP as a current standard, there is not a PBNM solution that stands out, and consequently there is no market standard [3]. The tools available at present, such as HP PolicyXpert [4], ExtremeWare Entreprise Manager [5] and Cisco QoS Policy Manager [6], do not completely follow IETF (Internet Engineering Task Force) works, such as the use of open protocols, standard data schemas and natural language to define policies [7] [8].

The work this paper presents is the result of the study, development and implementation of a prototype PBNM tool [9] [10]. By using IETF-defined standards, such as LDAP and SNMP, this tool, called Network Executive, has a Web interface that allows the use of policies over several network devices. Network Executive was implemented in Java and C, and designed by using a layered architecture. The system was developed so as to be easily expandable, since new QoS technologies may be available at any time. Thus, it is important that the environment can acknowledge and use new network resources as soon as possible.

Network Executive is part of the QAME (QoS-Aware Management Environment) project. QAME and Network Executive share common databases such as the policies, the targets and the associations. This information is stored on an LDAP directory, using a data schema similar to the recommended by IETF and defined by Snir et al. [11].

This paper is structured as follows: the second section presents QAME, in which Network Executive is responsible for policies. Network Executive details and its architecture are explained in section three. The elements that apply these policies, called Policy Consumers, are presented in section four. The fifth section discusses the data scheme used by Network Executive to store information on the system. A brief example of the use of the tool is presented on the sixth section and, finally, section seven presents conclusions and possible future work.

2. Context

Managing a QoS-enabled network involves analyzing a much greater deal of information than that found on the management of networks that are not QoS-enabled. Traditional management can not properly manage such a network because of the excessive time spent by the manager to analyze management information. Thus, there is a clear need for a new management model in which the large amount of data can be reduced to acceptable levels so as to allow the network manager to solve problems promptly.

Policy-Based Network Management (PBNM) is a feasible alternative because it allows the manager to set actions to be carried out by the network without worrying too much about network details. A new abstraction level is presented to managers so that they can define suitable actions in due time and still have a global view of the network.

As mentioned earlier, Network Executive is part of the management environment called QAME. QAME (QoS-Aware Management Environment) [12] attempts to provide the manager with the ability to use QoS management more effectively. QAME supports the execution of six main QoS management tasks: QoS installation, monitoring, operation maintenance, discovery, visualization and analysis. Policy definitions and their deployment on the network are part of QAME's QoS operation maintenance. In order to perform such management tasks effectively, QAME architecture features distinct but related software modules that can be located on several critical parts of the network (figure 1).

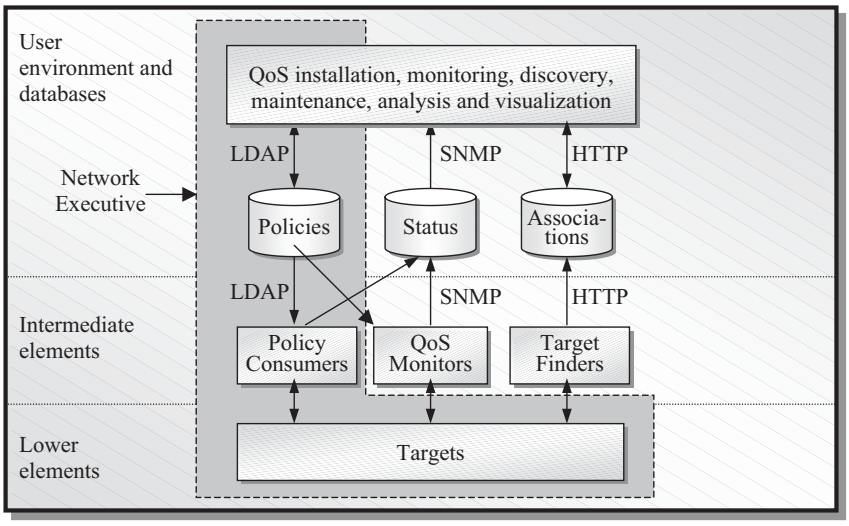

Figure 1 – Network Executive's Scope on the QAME Architecture.

The lowest level of the architecture features the *targets*, which are the elements that actually implement a QoS solution. Some examples of targets are: queue priority algorithms on the output interfaces of a router, traffic shaping and congestion avoidance processes and the marking packets algorithms in differentiated services on the input interfaces.

On the layer above, there are the intermediate architecture elements: target finders, policy consumers and QoS monitors. *Target finders* are special agents that search network devices for new targets. Every time a new target is found, a record is stored on the *associations* database. The *policy consumers* install policies on targets by translating policy descriptions into specific actions for each target. To do so, the *associations* database has information to match the stored targets and the defined policies. Finally, to make sure the installed policies are performing as expected, *QoS monitors* analyze traffic on critical network segments and compare their actual behavior to their expected behavior. If the disparity of performance exceeds a critical point, the QoS monitor will trigger a notification to inform the manager about a QoS degradation.

The system features three main databases. The previously mentioned associations database, the policies database and the status database. The *associations* database stores the targets and intermediate elements of the system (policy consumers, target finders and QoS monitors), as well as the information to match them to one another. It also stores information

to match the policies stored on the *policies* database to the targets on which such policies are installed. The *status* database stores the current state of defined and installed policies, so that the network manager can assess the effective use and operation of a policy.

Finally, the higher level of the architecture features the *user environment*, where the network manager interacts with QAME through a Web interface in order to manage the six QoS-related aspects mentioned before. As figure 1 shows, QAME has a distributed architecture, because the several elements can be located on different network devices. For instance, a policy consumer can be located within the management system whereas another consumer can be implemented within a router that supports PBNM.

Figure 1 also shows the scope of Network Executive and which elements of the QAME architecture it is responsible for. Network Executive has access to the policies database, where Policy Consumers get information about which Targets they should deploy the policies on. Network Executive is also responsible for presenting operation maintenance information to the manager through the QAME's integrated Web interface. The next section presents more details about how Network Executive operates within this context.

3. Network Executive Architecture

Figure 2 presents the complete architecture of Network Executive.

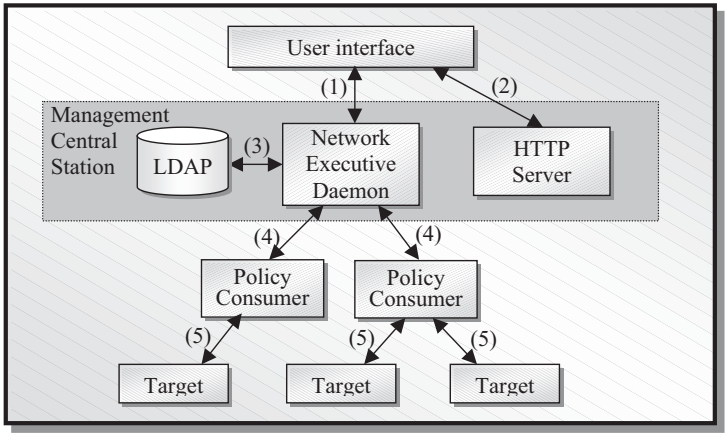

Figure 2 – Architecture of Network Executive

Basically, the system is divided into four main elements:

- the *user interface*, through which the network manager interacts with the system;
- the *management central station*, which implements the logic of the functionalities requested by the manager;
- the *policy consumers*, which translate the defined policies into specific actions for each target; and
- the *targets*, which are programmed according to the policies defined for the network where they are found.

The *Network Executive Daemon*, the LDAP server and the HTTP server are located on the management central station, while the policy consumers are found on several network parts, preferably close to the targets in order to minimize network management traffic.

The main part of the system is the *Network Executive Daemon*. It is responsible for managing user-defined policies by storing and consulting the LDAP server and calling on the policy consumers in charge of the many targets involved in a QoS policy.

Network Executive Daemon was developed in C. It uses API functions included in the OpenLDAP package [13] to access the directory service (figure 2, arrow 3) and it communicates with the Web interface (figure 2, arrow 1) and the policy consumers (figure 2, arrow 4) through a proprietary protocol that uses sockets. The LDAP directory service used to store the policies is OpenLDAP, also available through that same package. This server implements version 1 of the LDAP protocol.

The system's web interface is implemented in Java 1.1 as Applets so that any web browser can use the system. The JDK version used is 1.1 because it corresponds to the Java Virtual Machine found on most web browsers, even though newer JDK versions, such as 1.4.0, are available. Likewise, the option for Java 1.1 aimed at increasing the number of potential users, since it is enough to have an updated web browser. The classes that make up the graphic interface are stored on any HTTP server. For the specific implementation of this system we have used the HTTP Apache [14] server for Linux.

Along with the *Network Executive Daemon*, other important elements of the system are the *policy consumers* presented in figure 2. They handle the implementation of user-defined policies. A given policy consumer can have one or more *targets* under its responsibility. The only restriction is that these targets should implement the same type of QoS solution (i.e. DiffServ, IntServ, ATM, etc.). This is so because there are different policy consumers for different QoS solutions. Therefore, an ATM switch will be under the responsibility of an ATM policy consumer while a DiffServ router will be managed by a DiffServ policy consumer. The option for implementing different policy consumers for different QoS solutions aims at making the system more flexible for future expansions, such as the development of a new policy consumer for new QoS technology. The only restriction while developing new Policy Consumers is that the set of messages they can receive from the *Network Executive Daemon* must be the same to guaranteed compatibility with others Policy Consumers and the whole system.

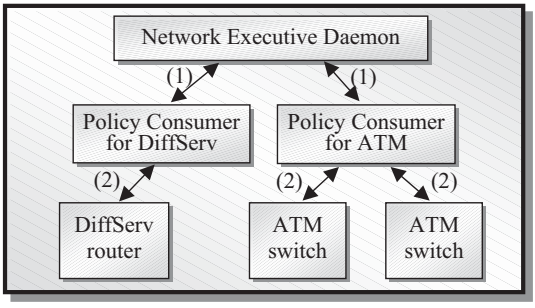

Figure 3 – Different Policy Consumers for Different QoS Solution

The policy consumers receive policies from the *Network Executive Daemon* through a proprietary socket protocol (figure 2, arrow 4 and figure 3, arrow 1) similar to COPS [15].

The policy consumers implement policies on the targets (figure 2, arrow 5 and figure 3, arrow 2) by using standard protocols such SNMP. The SNMP functions are available through the NET-SNMP library [16] for Linux.

Finally, it is important to point out that the discovery of new QoS resources and it's *targets*, is not the responsibility of Network Executive. As mentioned in section 2, this function is performed by the *target finder* module of the QAME environment (see figure 1). Newly discovered targets are stored on the *association* database, which Network Executive can access in order to retrieve information about targets on which policies can be applied.

4. Policy consumers

Policy consumers are elements whose main function is to apply policies on the targets under its responsibility [7]. As Network Executive follows the recommendations of the IETF model, the policy consumers in figure 2 have exactly the same functions as the IETF-defined policy consumers.

The architecture of Network Executive implements a set of policy consumers, one for each type of QoS technology support by the system. Hence, an ATM switch will be under the responsibility of an “ATM policy consumer”, whereas a Bandwidth Broker will be managed by a “Bandwidth Broker policy consumer”.

A single policy consumer can apply policies on several different targets as long as they implement the same QoS solution. The option for diverse policy consumer implementations according to the types of technology aims at making the system more expandable. Thus, Network Executive is able to support new QoS technology when a new policy consumer is developed for this technology.

Regardless of the type of policy consumer under analysis, the set of messages it can receive from the *Network Executive Daemon* must be the same so as to keep compatibility of all PBNM system elements.

As in the model proposed by Steves et al. [8], the policy consumer is in charge of checking whether a policy implementation succeeds or fails on a target at the time of application to it. However, some situations require constant monitoring of QoS parameters on some network devices. In such cases, other agents should be able to perform such task, since it is not the responsibility of the policy consumer. The QAME environment has *QoS Monitor* modules that implement these functions [17].

Policy Consumer for Bandwidth Broker

The *Bandwidth Broker* concept that Nichols et al. [18] present is related to an element that is in charge of negotiating a bandwidth reservation with other elements that are able to reserve network resources. Therefore, a *Bandwidth Broker* for DiffServ can allocate network bandwidth by negotiating with a DiffServ router it is aware of.

The biggest advantage of using broker services is that it simplifies the bandwidth reservation procedure in situations that involve several routers. In such cases, it is necessary to supply the broker with the source and target addresses and it will program a bandwidth reservation with every router which the information will flow through. If the manager wishes to do the same without a broker, it will be necessary to configure each router separately until all the path between end nodes is set.

The “policy consumer for bandwidth broker” available on Network Executive is able to apply QoS policies on devices that implement differentiated services [19]. It must be pointed

out that, when broker services are used, the policy target will be the broker and not its subordinate devices.

The policy consumer this section describes uses the brokers implemented by Granville et al. [20]. Thus, a policy the manager defines through the system's Web interface is sent by *Network Executive Daemon* to the policy consumer. Then, the rules must be translated for an access interface used by this type of broker. In this case, the interface is an SNMP MIB for QoS [20]. The bandwidth reservation is made by SNMP SET commands on the *BBServerDiffServTable* defined on this MIB. On table 1, the column on the right presents the objects belonging to the *BBServerDiffServTable*. The column on the left has the parameters used to define policies. Hence, each table line represents a mapping of a policy parameter into a corresponding MIB object implemented by the broker. It is important to notice that this mapping is specific for this type of broker. Other QoS solutions would have other translations and would use Telnet commands, COPS requests, and others.

Table 1 - Policy translation for bandwidth brokers

Policy	Broker
Source IP address	BBServerDiffServSourceAddress Object
Source Port	BBServerDiffServSourcePort Object
Target IP address	BBServerDiffServDestinationAddress Object
Target port	BBServerDiffServDestinationPort Object
Protocol	BBServerDiffServProtocol Object
Class of Service (throughput, delay, jitter and lost_factor)	BBServerDiffServBandwidth Object, BBServerDiffServPrio Object, BBServerDiffServLimitPrio Object

As we see, the relation between policy parameters and the broker table fields is very simple and direct. The most complex procedure is translating the policy-defined Class of Service into the objects BBServerDiffServBandwidth, BBServerDiffServPrio, and BBServerDiffServLimitPrio of the MIB. Through these three objects, it is possible to obtain the QoS parameters the manager describes in the policy.

5. Data scheme

The model proposed by IETF [8] presents a data scheme that must be used to represent system information on the LDAP directory. The recommended schema to store QoS policies is the "*QoS Policy Schema*" [11]. The use of a standard scheme allows PBNM tools of different suppliers to share the same set of policies. Unfortunately, commercial PBNM systems do not implement the data schema defined by IETF [21]. Network Executive implements a data scheme that is partially compatible with the proposed model. This is because the original model is relatively complex and still under development (*internet draft*).

Figure 4 shows the data scheme used by Network Executive. The presented directory tree is enough to store the set of data this PBNM system needs to operate. There are subtrees for information on users (*usersGroup*), policy consumers (*consumersGroup*), targets (*targetsGroup*), associations (*associationsGroup*) and policies (*policiesGroup*). The latter is further divided into a tree to store classes of service (*cosGroup*) and another one for the rules themselves (*policyGroup*). Each subtree has one or more instances of a certain type of object. Next, these objects and their attributes are presented.

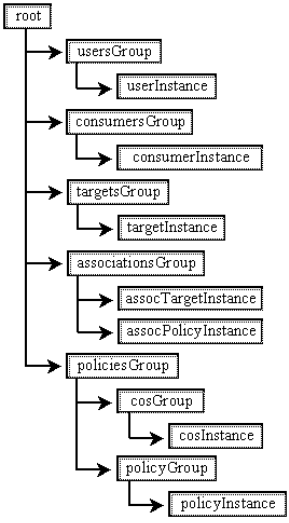

Figure 4 – Network Executive Data Scheme

- *userInstance*: username, name, password, access_level
- *consumerInstance*: ip:port, type, status, description
- *targetInstance*: ip:port, type, status, description
- *assocTargetInstance*: dn-targetInstance, dn_consumerInstance
- *assocPolicyInstance*: dn-targetInstance, dn-policyInstance
- *cosInstance*: name, throughput, delay, jitter, lost_factor
- *policyInstance*: name, source_ip, source_port, target_ip, target_port, protocol, dn_cosInstance

The underlined fields represent the attributes that must be unique, i.e., the key attributes. All attributes beginning "dn" represent fields that refer to other instances. Thus, an instance of *assocTargetInstance* has a reference to an instance of *targetInstance* and another reference to an instance of *consumerInstance*, for example. In this case, an instance of *assocTargetInstance* is relating a *targetInstance* to a *consumerInstance*, that is, a target to the policy consumer in charge of it.

6. Example of use

An example of the system element interaction follows. This example demonstrates the creation and application of a new policy on a given target.

1. The network manager uses a browser to access the *Network Executive Console*. Valid usernames and passwords must be entered on the initial screen.

2. After the user is verified, a new window allows the management of policies, users, policy consumers, targets and associations. Figure 5 presents the window for adding a new policy. The manager must supply source and target addresses and ports, a protocol, and a class of service. The "*save policy*" button starts the processes of policy validation, analysis and storing on the LDAP base.

Figure 5 – Network Executive Console

3. Once a new policy is defined, the manager must inform Network Executive about which target should be matched to the newly created policy. Only the target is informed, and the system should be able to know which policy consumer is in charge of applying the new policy. This association of policy consumers to targets is made when a new target is added to the system.

4. Once the new policy and target are associated, the manager must confirm the policy application process. After a few moments, the user is informed whether the newly defined rules have been successfully installed on the target.

By comparing the PBNM model presented by Stevens et al. [8] and the Network Executive architecture described above, one can observe that the Network Executive project tried to follow the IETF recommendations as closely as possible. This was a concern because most of the existing commercial tools implement proprietary solutions [3]. Therefore, different suppliers offer incompatible solutions. Consequently, two neighboring networks can not negotiate bandwidth reservation, for instance, because different PBNM systems can not communicate. Commercial systems that follow the IETF model, as Network Executive does, will be able to communicate with our tool directly and uniformly.

7. Conclusions and future works

New computer networks that are implemented must support QoS services, since new applications require such services. From a network manager's viewpoint, these new applications are responsible for an increased heterogeneity and complexity of management. The current management solutions do not supply managers with the information needed to implement actions defined by the manager. Policy-based Network Management (PBNM) was created to solve this situation.

The work this paper describes presented Network Executive, a management tool that uses the concept of policies to manage QoS-enabled networks. Its architecture and main features were also described. The use of open protocols, a data schema partially compatible with the IETF model and different policy consumers to different QoS technologies are the most important features of Network Executive. QAME, an integrated solution to manage QoS, has also been shown. Network Executive is part of QAME and interacts with other elements of this environment by sharing common resources such as databases.

Future research involves the implementation of support to the COPS protocol in order to carry policies between Network Executive and the system's policy consumers and between policy consumers and targets. Other goals for further investigation are the implementation of the complete IETF data schema, development of new policy consumers and the use of natural language in the Web interface to make policy defining easier for the network manager.

Bibliography

[1] QoS Forum: **Introduction to QoS Policies**. White Paper. Available at: <http://www.qosforum.com>. As accessed in July 2001.

[2] Saunders, S.: **The Policy Makers**. Data Communications, May 1999.

[3] Conover, J.: **Policy-Based Network Management**. Network Computing, Nov. 1999.

[4] HP. **PolicyXpert**. Available at: <http://www.managementsoftware.hp.com/products/policyexpert/index.asp>. As accessed in July 2001.

[5] Extreme Networks. **ExtremeWare Enterprise Manager**. Available at: <http://www.extremenetworks.com/products/datasheets/entmngr.asp>. As accessed in July 2001.

[6] Cisco. **QoS Policy Manager**. Available at: <http://www.cisco.com/warp/public/cc/pd/wr2k/qoppmn/index.shtml>. As accessed in July 2001.

[7] Mahon, H.; Bernet, Y.; Herzog, S.; Schnizlein, J.: **Requirements for a Policy Management System**. IETF, Nov. 2000. (Internet draft <draft-ietf-policy-req-02.txt> work in progress). Available at: <http://www.ietf.org>. As accessed on July, 2001.

[8] Stevens, M.; Weiss, W.; Mahon, H.; Moore, B.; Strassner, J.; Waters, G.; Westerinen, A.; Wheeler, J.: **Policy framework.** (Internet draft <draft-ietf-policy-framework-00.txt> work in progress). Available at: <http://www.ietf.org>. As accessed in November 2000.

[9] Coelho, G.: **Network Executive: Uma Ferramenta para Gerenciamento Baseado em Políticas**. Trabalho de Diplomação. [Network Executive: A Policy-Based Network Management Tool. Graduation paper] Universidade Federal do Rio Grande do Sul – Instituto de Informática, 2000.

[10] Coelho, G.; Granville, L.; Almeida, M.; Tarouco, L.: **Network Executive: Uma Ferramenta para Gerência de Redes Baseada em Políticas**. In: Proc. XIX Simpósio Brasileiro de Redes de Computadores. Florianópolis, 2001.

[11] Snir, Y.; Ramberg, Y.; Strassner, J.; Cohen, R.: **QoS Policy Schema**. (Internet draft <draft-ietf-policy-qos-schema-01.txt> work in progress). Available at: <http://www.ietf.org>. As accessed in November 2000.

[12] Granville, L.; Tarouco, L.: **QAME - An Environment to Support QoS Management Related Tasks on IP Networks**. In: Proc. IEEE International Conference on Telecommunications. Bucharest - Romania, 2001.

[13] **OpenLDAP**. Available at: <http://www.openldap.org>. As accessed in July 2001.

[14] **Apache HTTP Server**. Available at: <http://www.apache.org>. As accessed in July 2001.

[15] Boyle, J.; Cohen, R.; Durham, D.; Herzog, S.; Rajan, R.; Sastry, A.: **The COPS (Common Open Policy Service) Protocol**. IETF, 2000. (Request for Comments 2748). Available at: <http://www.ietf.org>. As accessed on July, 2001.

[16] **NET-SNMP**. Available at: <http://net-snmp.sourceforge.net>. As accessed in July 2001.

[17] Ribeiro, M.: **Implementação de uma arquitetura para monitoração de QoS em redes IP**. Trabalho de Diplomação. [Implementation of a QoS monitoring architecture on IP networks. Graduation paper] Universidade Federal do Rio Grande do Sul – Instituto de Informática, 2000.

[18] Nichols, K.; Jacobson, V.; Zhang, L.: **A Two-bit Differentiated Services Architecture for the Internet**. IETF, 1999. (Request for Comments 2638). Available at: <http://www.ietf.org>. As accessed on July, 2001.

[19] Blake, S.; Black, D.; Carlson, M.; Davies, E.; Wang, Z.; Weiss, W.: **An architecture for differentiated service**. IETF, 1998. (Request for Comments 2475). Available at: <http://www.ietf.org>. As accessed on July, 2001.

[20] Granville, L.; Uzun, R.; Tarouco, L.; Almeida, J.: **Managing Differentiated Services QoS in End Systems using SNMP**. In: Proc. IEEE Workshop on IP-oriented Operations & Management - IPOM 2000. Krakow - Poland, 2000.

[21] Saunders, S.: **Directory Disintegration**. Data Communications (page 50), May 1999.