An Auditing System for QoS-Enabled Networks*

George Din[1], Akihiro Hayakawa[2], Ina Schieferdecker[1], Peter Deussen[1]

Email: {din, schieferdecker, deussen}@fokus.fhg.de, hayakawaak@nttdata.co.jp

[1]Fraunhofer Research Center for Open Communication Systems (FOKUS)
[2]NTT DATA Corporation

Abstract

The ongoing challenge of supporting high performance communications, real-time critical and secure applications leads to a shift from traditional IP networks towards a more intelligent, end-to-end, service-aware network paradigm. Future networks have to i) be more content or application-aware; ii) provide dynamic features for service creation; iii) observe and enforce network-wide policies; and finally, iv) enable control from the network provider to the end-user. This paper presents the technical aspects of a policy based auditing system for a QoS-enabled network. The goal is to define the architecture of policy-based network QoS control and to develop and implement a prototype to evaluate this architecture.

Keywords: Auditing, QoS, Policy-based Network Management, Online Testing, TTCN3

1. Introduction

The ongoing challenge of supporting high performance communications, real-time critical and secure applications leads to a shift from traditional IP networks towards a more intelligent, end-to-end, service-aware network paradigm. The increasing need for enhanced network services such as virtual private networks (VPN), quality of service (QoS), security, collaboration, and directory technologies underpins demands on the core infrastructure of next generation networks: flexibility, service differentiation, isolation, privacy, and manageability. Future networks have to i) be more content or application-aware; ii) provide dynamic features for service creation; iii) observe and enforce network-wide policies; and finally, iv) enable control from the network provider to the end-user [13] .

This paper presents the technical aspects of a policy based auditing system for a QoS-enabled network. The goal is to define the architecture of policy-based network QoS control and to develop and implement a prototype to evaluate this architecture. Our work deals with two main tasks.

Firstly, we consider a QoS enabled target network. In such a network all resources are managed with the help of network policies, which are deployed dynamically at runtime. Such policies reflect, on one hand the user requirements, whose interaction with our system is done automatically via a web interface, and, on the other hand, the current status of the network and the different functioning requirements imposed by the network or service manager. In this scenario we want to design and develop a policy computation system able to process different kinds of information (user requests, operation policies, network status, authorization policies etc.) and to compute the final end network policies deployable into the target network. Secondly, because network faults as well as policy misconfigurations or policy conflicts may prevent the intended enforcement of policies, we audit the target network in order to validate the policy enforcement.

The validation activities are performed by the auditing system that uses a measurement system to probe, at different levels, the network components. The collected data is evaluated by an on-line test component that employs a number of so-called on-line test cases (OTC), i.e. executable and dynamically deployable validation components.

The idea of a policy based network management system is not new. In [14] some challenges and practical usage scenarios on policy based management of network services are presented. A terminology for Policy Based Management is defined in [16] .

Several concrete architectures for policy-based management of networks are proposed in [3] [7] [14] [17] . One common characteristic of all these approaches is the XML data format, which is adopted by all of them as formal language to describe policies. The architecture for a system to detect faults in distributed systems at application level is outlined in [15] . All these papers discuss in detail different architectures for policy-based management of distributed systems and also consider some possibilities to specify policies.

Apart from that, some papers argue and consider different policy languages. Relevant here can be the Core Policy Model specification of DMTF[5] . A survey on the network policy languages is realized in [6] while several policy languages are described in [14] [21]

To capture information about the network behavior, different measurement systems are used. [1] introduces an

* This work was partially supported by the NTT Data – Fraunhofer Fokus cooperation Project "NG-CDN"

Proceedings of the 23 rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03)
0-7695-1921-0/03 $17.00 © 2003 IEEE

architecture for monitoring and measuring the traffic in IP networks. [18] discusses a monitoring tool able to perform measurements of QoS parameters.

The idea of QoS testing, formulated also within this paper, is formulated in [11] [22] Some concrete QoS tests and auditing scenarios are detailed in [9] [19]

Our work introduces the idea of policy-based QoS management in a network: resource negotiation, QoS reservation, policy computation, enforcement, auditing and removing. The QoS requirements are expressed in the form of policies. Depending on the abstraction level, we deal with different levels of policies:

- *high level policies*: here are reflected high level QoS requirements regarding QoS resources.
- *network level QoS policies*: these policies are enforced into the network. They are derived from the high level policies according to network and device-specific requirements.
- *auditing and measurement policies:* derived directly from the network level QoS policies in order to verify the correctness of QoS provisioning.

This paper discusses only the auditing architecture, which is concerned with the network layer assisting of QoS provisioning. In this respect we introduce our approach for a distributed auditing system. This work is motivated by the demand for enabling IP networks to provide QoS support. Tests for QoS and verification scenarios are already performed, some of them were reminded also in the related work section. What we try to achieve here is an automated system able to continuously audit and test the QoS parameters.

QoS provisioning and the different layers of computing, reservation, enforcement make possible the occurrence of faults, policy violations, malfunctions etc. Without auditing it is very difficult to appreciate exactly where the faults appear in the network and which can be the cause. Elementary QoS tests are able to detect which network elements are not working properly. The auditing system, based on systematical analysis and interpretation of the collected data about the behavior of the network, is able to detect which QoS policies are not correctly enforced. Whenever the QoS requirements, specified within a policy, are not fulfilled we deal with a *QoS policy violation.* The Auditing System has the main functionality to detect *violations.*

The paper is organized as follows: In Section 2, we present a general approach of a policy-based management system that extends the framework of the IETF/DMTF. Section 3 focuses on the architecture of the auditing system component. Section 4 presents a case study that was performed in order to validate the applicability of our approach. A summary is given and conclusions are drawn in Section 5.

2. An Approach Towards QoS-enabled Policy-Controlled Networks

Classical Internetworking Protocol (IP) based networks provide a "best-effort" service, which treats all applications equally. However, IP networks need more guarantee for communication quality in order to provide different service levels such as streaming video, on-line games, or interactive real-time applications. QoS enabled networks provide mechanisms to allow network applications to request needed network resources. Several studies have been made on QoS enabled network using policy-based management [2] , i.e. according to [12] , the combination of rules and services where rules define the criteria for resource access and usage.

The Internet Engineering Task Force (IETF)/ Distributed Management Task Force (DMTF) defines a policy-based management framework consisting of the following basic elements [2] : policy management tool, policy repository, policy decision point, and policy enforcement points.

Policies are defined either by the network administrator or are automatically generated using a policy management tool, based on the definition of service level agreements, user requests, and so on. After a policy has been defined, it is stored in a policy repository. A policy decision point (PDP) interprets the policies stored in the repository and enforces them in the network on policy enforcement points (PEPs).

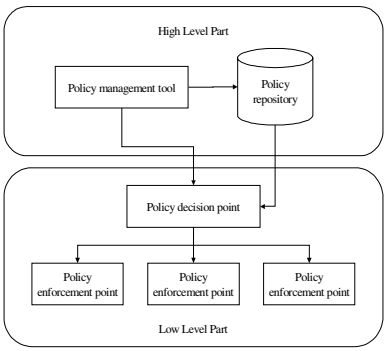

Figure 1. The IETF/DMTF Policy Framework

In contrast to that, our policy framework is divided into the following two parts (compare Figure 1): The High Level Part (HLP) derives and decides upon the policy to be used to realize a requested QoS. It includes a management tool and policy repositories. The Low Level Part (LLP) enforces the selected policies. LLP includes PDP and PEPs.

The purpose of the HLP is to generate the policies to provide the required QoS. There are different approaches to compute the appropriate policy. The Tequila-project, for instance, proposed an approach based on Service Level

Specifications (SLS). An SLS [8] [9] is a definition of the QoS parameters provided to the user. The SLS is derived from the service level agreement (SLA) between user and provider for dynamic service invocations [20] .

This paper focuses on LLP and assumes that the policies to be enforced within a network are given from HLP.

Low Level Part

The primary objective of LLP is to enforce the policy computed by HLP. However, after the policy enforcement, these policies and therefore the SLAs being supported by the enforced policies can be violated e.g. due to faults in network components. Moreover, additional policies being enforced within the network may violate the policies already enforced. Therefore, within QoS-enabled policy-controlled networks, it is necessary to audit whether the agreement upon QoS according to a SLA is provided or not. Hence, the second objective of the LLP is to *audit* enforced policies.

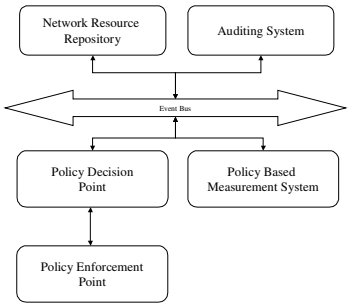

Figure 2. Architecture of LLP

Figure 2 presents the main components of the LLP: Network Resource Repository (NRR), PDP, PEP, Policy-Based Measurement System (MS) and the A*uditing System* (AS). The NRR stores SLSs as generated by the HLP and converts them into various concrete policies, i.e. QoS policies, Measurement Policies and Auditing Policies to configure and control the components of LLP. The PDP is responsible for translating QoS policies based on SLSs into device-specific actions and for enforcing them within the PEPs. The MS component triggers the AS to start an appropriate set of on-line test cases (OTCs) whenever SLS violations occur. The measurements initiated by the NRR to make the provided quality visible are analyzed by the MS. If the MS detects an SLS violation, it creates an SLS violation event and transmits it to AS.

The AS is dedicated to probe network elements in order to identify problems and failures in the current behavior of the network. By using the MS to probe the network, the AS processes the collected data, detects and reports possible policy violations within the network.

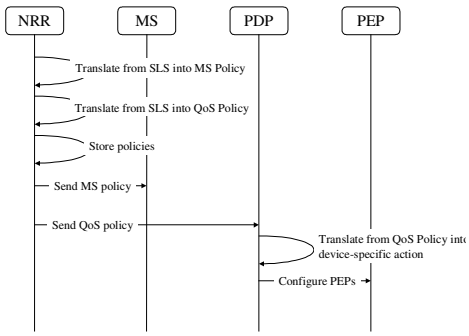

Figure 3. LLP components interaction (1)

Figure 3 shows the various steps used to support QoS also in the case of network malfunctions and failures.The NRR translates SLSs into MS policies. These translations use network topology information (router address table) in order to determine measurement points in the target network.

The NRR translates SLSs into QoS policies. Again, topology information is used as a parameter of this translation process, but this time, to determine whether a particular router enables a certain behavior specified in the SLS.

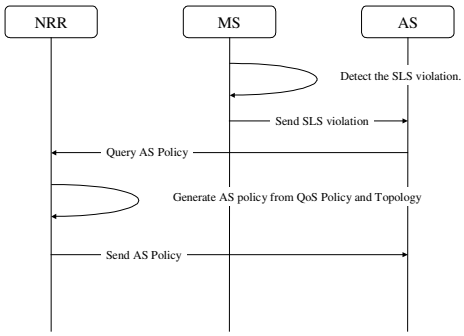

Figure 4. LLP components interaction (2)

The NRR stores QoS policies, MS policy and the SLS, and, in the next steps, sends QoS policies to PDP and MS policies to MS. The PDP translates QoS policies into device-specific actions to be executed by the PEP. Device-specific actions are parameters to configure PEP. Hence, the PDP configures PEP with those actions.

Figure 4 presents the sequence of the auditing procedure. The auditing procedure consists of the following steps:

- When the MS detects a policy violation, the AS requests the NRR for the AS policies corresponding to the violated SLS.
- The NRR computes the Auditing Policy using AS topology information, the QoS Policy and the underlying SLS. The AS topology information consists of meter addresses and a router address table.
- The NRR sends the Auditing Policies to the AS.

The policies for auditing and SLS measurement are computed in the NRR. The computation is basically a derivation from the SLS specification and topology information. Hence, to compute the measurement policies, two items of information are needed: information about the topology (which meters to be used to perform measurements) and (the XML description of) the SLS to find out the values for valid operation. If the values gathered by the AS are not in normal, specified limits, then a violation has occurred and the AS has to be triggered.

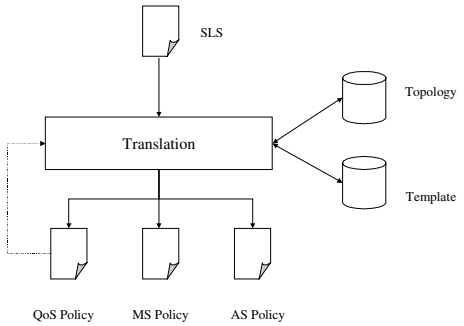

Figure 5. Policy computation

The same parameters are needed for QoS policy computation (see Figure 5) but this time Topology represents information about the routers enabling a certain behavior specified in the SLS.

The computation procedure for auditing policies needs as input parameters: the QoS Policies and the topology description. The latter represents the information about the meters that are employed for the detection of QoS policy violations.

3. Auditing System Architecture

The Auditing System probes the network elements to identify inadvertences in the current behavior of the network. The AS is triggered by the MS whenever violations visible to MS occur. Then, AS makes a lookup in NRR to find out, which policies are enforcing a particular SLA, probes in further detail properties of the violated policy, processes the outcome and gives potential causes of the policy violation. Independently of the MS, the AS performs probing also during the enforcement and removal of policies. This allows identifying policy misconfigurations and policy conflicts.

The AS consists of four components being i) the *system model*, ii) the *online test case* (OTC), iii) the *workflow management system* (WFMS), and iv) the *graphical user interface* (GUI). These components interact using a *communication bus* realized by a low-level communication middleware (compare Figure 6, all figures that exemplifies the architecture of the auditing system are in UML notation).

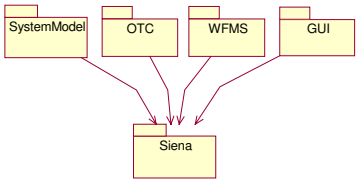

Figure 6. Component View of the Auditing System

The system model (see Figure 7) is a repository of information about the measured data. This component consists of two subcomponents: the database itself and a controller that connects to Siena and interprets addressed notifications. These notifications can be either lookups to ask for different information or updates to add/remove data.

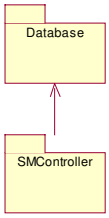

Figure 7. System Model Component

The functionality of WFMS (Figure 8) is enforced by a set of several subcomponents called *plug-ins*. One important plug-in is the ErrorHandler that receives the SLS violation notifications. It looks up the NRR for the policies enforcing the violated SLS. Another plug-in, called Expander, receives the answer from NRR. Its role is to make a list of requests for new OnlineTests corresponding to the policies to be tested. These requests are sent to OTCController.

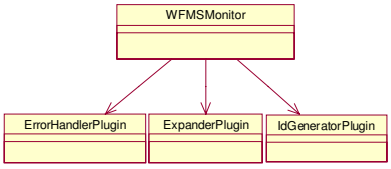

Figure 8. WFMS view

The OTC consists of the OTCController and several OnlineTests. The OTCController creates new OnlineTests whenever a new policy is to be tested. This way we establish a one-to-one mapping rule between policy and tests: each policy corresponds to one – potentially complex – online test. Both, the "create" request, as well as, the "remove" request for Online Test objects is sent by WFMS.

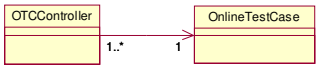

Figure 9. On-Line Test Case Management Component

Last but not least, the GUI (see Figure 10) being a web-based component visualizes data from the system model base. Data that is stored in the SystemModel is transformed to graphical information. The transformations are performed by using JSPs [27] and the data will be requested and received by JavaBeans [28] . The Browser uses the servlets to load any html/jsp pages. The Servlets instantiate the JavaBeans, which are used in two different ways. Firstly, they interact with Siena to look up in the SystemModel base for data and secondly, they are imported by the JSP pages as Java objects to access data received from the SystemModel.

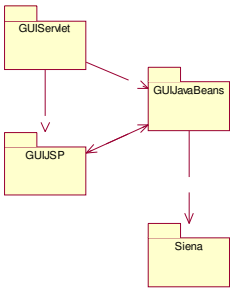

Figure 10. Component View of the GUI

The communication bus, for the above-mentioned components, is Siena [24] . It is started with the setup of the LLP. All other components joining the communication bus have to subscribe to it.

As it follows we describe central functions of WFMS and OTC. The WFMS decides on and coordinates the activity in the auditing system via workflow processes.

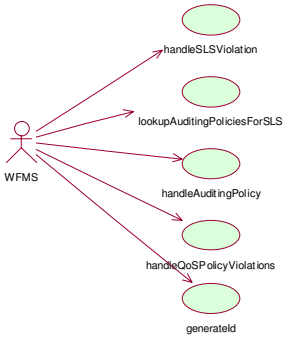

Figure 11. WFMS Use Case Diagram

The OTC functionality deals with a number of tasks: the handling of the auditing policies, the interpretation of the data measured by probing subsystem and the reporting of the violations.

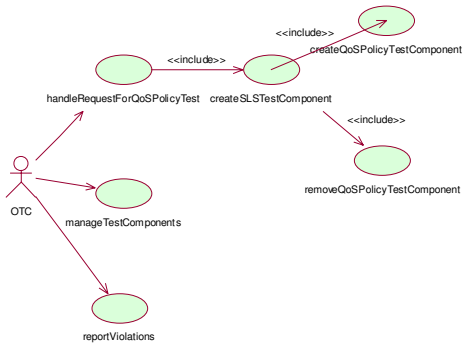

Figure 12. OTC Use Case Diagram

The interaction between the AS components, NRR, and the MS is shown in Figure 13. Whenever the MS detects the violation of a SLS it informs the WFMS. This requests the corresponding list of policies from the network resource repository. The list is used to initiate the online tests belonging to these policies. The online tests themselves may require additional measurements, which are requested from the MS. The collected data is stored in the database and analyzed by the OTC, which reports the cause of the policy violation to the WFMS. This can then decide on the appropriate means to reconfigure or fine-tune the network in order to repair and/or prevent further policy violations.

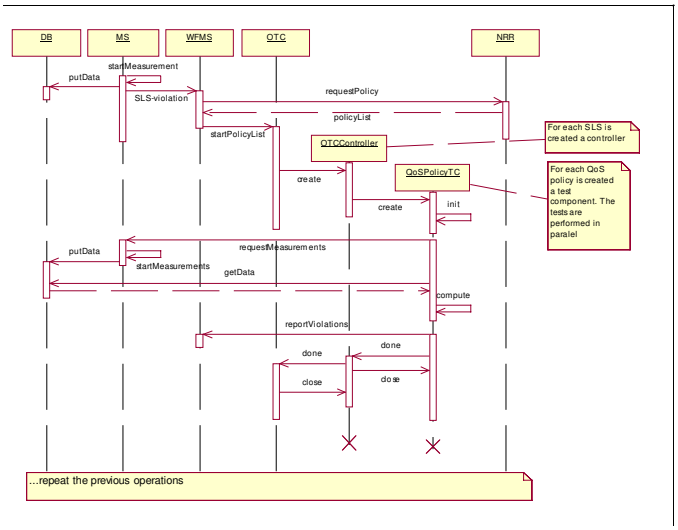

Figure 13. Sequence diagram for the use of AS

Currently, AS provides services to audit bandwidth, delays, packet dropouts for flow related QoS as well as to audit the presence/absence of flows. For example, the bandwidth services audit bandwidth allocated for a certain micro-flow and compare the measured value with the limits. At the invocation of this service, AS starts measurement actions with a rate set by default. The statistical interpretation will be performed on top of predefined formu-

las chosen as auditing strategy. The AS services offer the possibility either to test the existent traffic, in a passive way, or to perform active tests, by auditing specific flows of interest. The network provider sets the flag "type", with value *active* or *passive* when configuring the computation of the auditing policy.

Technology Selection

An auditing system that satisfies the needs of auditing of complex networks must addresses a number of issues:

- **Open and scalable platform**. It must maintain a flexible and modular structure.
- **Platform independent**. Therefore we rely only on java-based technologies.
- **Generic communication interfaces** between components
- Easy **transportation of policies and measurement** results.
- Provide support for **integration on-the-fly of new modules** for auditing, data visualization or storage of interpretation.
- Use of a **test definition language** to describe policy violation scenarios.

To meet these requirements we chose several adequate technologies, which are presented briefly:

TTCN3 [25] : The Testing and Test Combined Notation (3rd edition) is the standard test specification and implementation language from ETSI. It is a test language to define test procedures for black-box testing of distributed systems. In our case study we used several features of it, like: the ability to schedule and synchronize probe operation and data retrieval across the network, the ability to give stimuli to the network and observe the reactions, and to set verdicts on validated behavior, and finally, the easy management of deployment of probe connectors. We used TTCN3 in the OTC component in order to describe online test cases for QoS validation.

Xindice [26] : The storage of the data is realized by using XML technologies. Xindice is a native XML database, which manages very efficiently XML messages. A generic interface, as front-end of the storage system, helps to create documents, and to query, add or modify data.

Siena [24] : The communication among different enabling components is implemented on top of Siena (Scalable Internet Event Notification Architecture), a standalone Java process that manages the distribution of the events to interested parties. Siena is a generic scalable event-notification service. Conceptually, Siena enables event distribution to transport observed data and to handle large amount of events.

Jahia [23] is an integrated content management and portal solution able to deliver information and applications to the connected users. With the help of Jahia we developed a dashboard application able to visualize different data received from the Siena bus. Our application, plugged to the communication bus, provides a common interface, XML based, for interaction with the other components. The data to be visualized is embedded into XML messages.

Network Meter [4] : Network Meter is a software system for the measurement and accounting of network resource usage information in IP networks. Its architecture presents methods for efficiently measuring IP traffic flows and has a flexible and modular structure. With help of this meter it is possible to compute a variety of traffic metrics for the measured IP flows. Other measurement tasks can be dynamically added and removed from the system. With the help of this extendable metering system we were able to add new measurement tasks for the QoS parameters.

Measurement System. A proprietary measurement manager system, based on java and XML technologies, was also implemented. It is basically the front end to the network meter elements. The MS provides a policy-based management interface for configuration and management issues. The functionalities of the MS are:

- to adapt the XML based policy request to meter commands
- to deliver measured information to the transportation bus.

4. A case study

In this section we describe the prototype implementation of the auditing system described in the former sections. A detailed quantitative evaluation of our approach is subject to future work. Here, we rather describe the technique itself and argue about its usefulness, which is mainly done with a demonstration scenario to show the applicability of our approach.

4.1. Networks DiffServ

Our case study was performed in the framework of a DiffServ capable network. DiffServ is the proposed IETF model for differentiated services in the Internet. It supposes that IP packets are marked with a byte value DS (Differentiated Service), which specifies how the packets should be treated by routers. Providing different service levels implies traffic classification and differentiation, whereas each service has a special quality level. As it was previously presented, these requirements for special QoS parameters are expressed in the form of policies.

The QoS specifications are computed according to different requirements, coming from users, service providers or network provider, and are enforced in the network by PEPs. From the enforcement of a policy until its complete removal, the verification of the supplied QoS requirements is the target of our auditing activities. The auditing

of QoS mechanisms implies different requirements for AS:

- Auditing of the protocols employed by the DiffServ architecture. In this respect, conformance tests of these protocols are required.
- Auditing of the status of the network. It is important to know the status of the network in order to perform computation on the availability of resources.
- Auditing of end-to-end QoS performance. This supposes auditing of particular flows corresponding to a service request. Thus, two approaches have to be considered: SLS auditing, in order to ensure that SLA requirements are fulfilled, and QoS policy auditing to detect whenever some particular QoS policy is violated.

4.2. Demonstration Scenario

We have built a demonstration setup to illustrate the functionality of the system. The architecture of the network we used for our evaluation scenarios is presented in Figure 14.

In this architecture, we used three Linux Routers configured for DiffServ capabilities and interconnected as presented in the figure. The three routers build up a network domain whose edge nodes are R1 and R3. Two other PCs (A and B) are connected to this network, to the edge router R1. As presented, on each link we plugged a meter (PCO1 to PCO5). To the other edge (R3) we connected another PC, which runs a streaming video server.

The scenario consists of sending video traffic between the video server and the two PCs across the network domain. Basically, we implemented a two-step approach. Firstly, we illustrated how Policy-Based Management System enables certain QoS configuration according to the desired levels of service. Both clients connect to the server and try to play the same movie. For the user A we allocate 4 Mbps (Premium service) and for user B we consider "best-effort". In the same time we generate also some dummy traffic across the network in order to occupy the 10Mb bandwidth we used to interconnect the computers. It is observable how after the enforcement time the video quality for the client A become considerable better in comparison with the video quality of client B.

Secondly, we simulate a network fault in order to influence the required QoS: the video quality becomes poor. Thus, given QoS policies were violated. We experimented with different types of faults:

- Link down: The link between R1 and R2 was removed. The auditing system reported that the router R2 no longer fulfils the QoS requirements enabled in the ingress node (R3) of the flow VideoServer→A.
- Node down: The ingress node (R3) was shut down. The auditing system detected the fault and considered that the bandwidth of 4 Mbit/s required by the video application was not allocated.
- Wrong marking of DS field. This was simulated by not enforcing the QoS marking policy in the ingress node for the flow VideoServer→A. The auditing system determined that the flow VideoServer→A was treated as "best-effort" class instead of the Premium service class and the marking policy was violated.
- Non-enforcement of a QoS Policy. The policy according to which the bandwidth was allocated for the flow VideoServer→A was not enforced. In this case, some stress traffic was added to the network. This traffic belonged to the same QoS class as the flow VideoServer→A. Hence, both flows were treated in the same way. The auditing system detected that the 4 Mbit/s were not allocated along the link R1→R3.

In all cases, the auditing system was able to detect the faults in the network and, even more, identified also which particular policies were violated. Permanently, the auditing system kept under observation the network elements extracting data events. According to these events, the AS made assumptions concerning certain QoS parameters. At this stage, only the bandwidth QoS parameter was considered. Whenever discrepancies between the audited values and the values within the policies were detected, we had a violation. Currently, we experiment with other QoS parameters: packet loss, packet delay or packet jitter, parameters whose values are specified within the QoS policies.

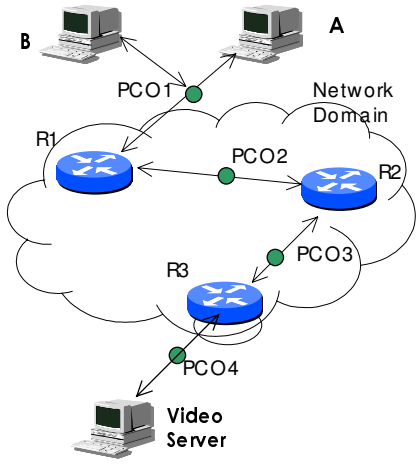

Figure 14. Demonstration Scenario

5. Conclusions

Shifting from classical "best-effort" internetworking to QoS-aware, service differentiating networks, effective and

efficient network management becomes an issue of increasing complexity.

We presented a new approach of policy based network management in QoS networks. For that reason, we refined the IETF/DMTF framework by separating the management component into a high level part and a low level part. The purpose of the high level part is to define and store SLSs, while the low level part is responsible for decision, deployment, and enforcement of policies. We concentrated not only on policy deployment and enforcement, but we also described means to audit the effect of policies in the target network. A dynamic monitoring system was used to gather information on network states and network component behavior in order to validate the enforcement of policies. Validation was done by using a dynamically configurable system of on-line test cases.

We performed a case study to give a proof of practicability for our approach. We implemented several failure scenarios concerning both network faults (link or node loss) and problems in the policy management system (non-enforcement of policies).

The future work will concentrate on defining more complex scenarios able to detect more sophisticated violations or malfunctions. An emerged work will be the research of auditing techniques of QoS enabled networks, which belong to different domains. Also some security issues will be also targeted.

Another interesting idea for future work would be to extend our system with mechanisms able to fix the detected malfunctions. Knowing exactly the problem and the place where the violation occurred, some new techniques will be necessary to adapt the network elements to provide again the requested QoS.

6. References

[1] Asgari, S. V. den Berghe, C. Jacquenet, P. Trimintzios, R. Egan, D.Goderis, L. Georgiadis, E. Mykoniati, P. Georgatsos, and D.Griffin, "A Monitoring and Measurement Architecture for Traffic Engineered IP Networks" (2001)

[2] A. Westerinen, et. al., "Terminology for Policy Based Management", IETF RFC 3198, November 2001

[3] L. Capra, W. Emmerich, C. Mascolo, „XMILE: An XML based Approach for Programmable Networks", In Proc. of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII). Schloss Elmau, Germany. May 2001.

[4] C. Schmoll, "Dynamically Configurable Nework Meter for Accounting in IP-based Networks", Diploma Thesis, Technische Universitaet Berlin, Berlin, December 2001

[5] DMTF, "CIM Core Policy Model", May 2000

[6] G. N. Stone, B. Lundy and G. Xie, "Network Policy Languages: A survey and a new approach", *IEEE Network,* vol 15, no 1, pp. 10-21, January 2001.

[7] H. De Meer, W. Emmerich, C. Mascolo, N. Pezzi, M. Rio and L. Zanolin, "Middleware and Management Support for Programmable QoS-Network Architectures", In ACM Proc. Int. Conf. Software Engineering (ICSE02). May 2002.

[8] Internet Draft, "Service Level Specification and Usage Framework", October 2000

[9] Internet Draft, "Service Level Specification semantics and Parameters", November 2000

[10] IXIA White Paper, "VoIP Quality of Service (QoS) Testing"

[11] J. Q. Walker, "Testing and Tuning QoS for Network Policies", 2000

[12] J. Strassner, E. Ellesson, "Terminology for Describing Network Policy and Services," Internet Draft draft-strassner-policy-terms-01.txt, Feb. 1999.

[13] J. Vicente, H. Cartmill, G. Maxson, S. Siegel, R. Fenger, "Managing Enhanced Network Services: A Pragmatic View of Policy-Based Management", Intel Technology Journal Q1, 2000

[14] M. Fisher, P. Mckee, "Policy Based Management of Large Distributed Systems", CaberNet Workshop, 2001

[15] M. J. Katchabaw, H.L. Lutfiyya, A.D. Marschall, and M.A. Bauer, „Policy Driven Fault Management in Distributed Systems",The Seventh International Symposium on Software Reliability Engineering (ISSRE '96)

[16] Network Working Group, "Terminology for Policy-Based Management", November 2001

[17] R. Natarajan, P. McKee, Aditya P. Mathur, "A XML based Poliy-Driven Information Service", In 7th IFIP/IEEE International Symposium on Integrated Network Management, May 2001

[18] R. R. Liao, S. H. Hwang, M. E. Kounavis, "Quality-of-Service Measurements for the MBONE", Advanced Internet Services: Project Report, 1996

[19] R. Stoy, J. Jaehnert, "Test of CISCO's IP QoS Implementation considering Differentiated Services", 1999

[20] Service Level Specification Interest Web Page, avail. at. http://www.ist-tequila.org/sls/

[21] Y. Kanada, "Rule-based Modular Representation of QoS Policies", *10th Networking Architecture Workshop*, pp. 106-113, IEICE, 2000

[22] Z. Cekro, T. Ferrary, "QoS Monitoring: test specification", TF-TANT task force 2000.

[23] Jahia Portal Server 3.0, http://www.jahia.org

[24] Scalable Internet Event Notification Architectures (Siena), http://www.cs.colorado.edu/serl/siena/ http://www.cs.colorado.edu/users/carzanig/siena/

[25] Tree and Tabular Combined Notation, version 3, ITU-T Recommendation Z.140, 2001 avail. at http://www.itu.int/ITU-T/studygroups/com10/languages

[26] The Appache XML Project, avail. at. http://xml.apache.org/xindice/

[27] Java Server Pages Technology, avail at. http://java.sun.com/products/jsp/

[28] JavaBeans Component Architecture avail at. http://java.sun.com/products/javabeans/