

## QoSpy an approach for QoS monitoring in DiffServ Networks.

Ulrich Hofmann

Alessandro Anzaloni

Ricardo de Farias Santos.

[anzaloni@ele.ita.br](mailto:anzaloni@ele.ita.br)

Instituto Tecnológico de Aeronáutica  
São José dos Campos-SP-Brazil

### Abstract.

This work proposes the development and evaluation of QoSpy, a SNMP-based monitoring application that uses information from networks elements (e.g. routers) to assess QoS. The idea is to perform intermediate monitoring to provide means of detecting and pinpointing QoS degradations into network. To validate this idea, a prototype was deployed and tested with a experimental testbed.

### 1- Introduction.

In the last few years, the Internet has turned from a research into a business network, with services that vary from simple online shopping to video conferences. But the Internet was not designed to put up with the actual business needs. Although voice and video applications can be used over the Internet, they can not guarantee their quality. To provide guarantees, there must be some kind of traffic differentiation in order to treat applications with diverse levels of priority. Quality of Service (QoS) aims at providing mechanisms that allow traffic associated with mission critical applications, like Voice over IP and Video Conference, to have a higher

priority over others. Some models that provide differentiation between services were already developed. Integrated Services(IntServ)[1] and Differentiated Services (DiffServ)[2] are the most important of them. The DiffServ model was considered in our work since it is more scalable and less resource consuming if compared with the IntServ model. DiffServ allocates resources on a aggregate basis. It associates the flow in classes and applies different QoS requirements to each class. The packets classification is based on the Diffserv Code Point (DSCP) field in the IP header. Nevertheless, it is not enough to commit resources, since QoS degradation will probably happen because of failures in networks elements or a not efficient QoS configuration. Therefore, QoS monitoring must be also employed in order to assess if the contracted QoS is being delivered to the customer. QoS monitoring subject has been studied in the last few years. Essentially, it is categorized as end-to-end and distributed QoS monitoring. In the first, the QoS assessment is carried out at the edges, i.e. in the source and destination of a flow. Parameters like

delay, jitter and packet loss are analyzed to detect degradation that may occur in the network. This scheme is able to report QoS degradation, but it cannot spot where it occurs. Distributed QoS monitoring does not analyze end-to-end results but it assesses QoS based on data fetched from intermediate monitors in the network path. Therefore, it is able to both detect QoS degradation and locate where it occurs. This is an outstanding advantage since it allows a manager to promptly solve the cause of degradation.

Although QoS monitoring is an important element, few tools were developed so far. Most of them are based on monitoring stations that are inserted into the network in strategic points to measure traffic (reference [3] is an example of such solution). This approach is not scalable and is very expensive since we must use and configure a computer for each monitoring point. Furthermore, we are supposed to specify beforehand the interesting monitoring points in the network, which is a very difficult task, since network congestions points are unforeseeable. Since it is not possible to set aside one monitor for each segment, we are not able to identify all degradations that can happen. Also, these tools usually execute per-flow monitoring instead of flow aggregates monitoring, which makes them not scalable and not suitable to DiffServ monitoring.

This paper presents the development and evaluation of a distributed QoS DiffServ monitoring tool, QoSpy, based on the Simple Management Protocol (SNMP)[4] to get data directly from the network elements (e.g routers), eliminating the need of installing and configuring network monitors into the network. The information is accessed with the help of MIBs (Management Information Base). The work has manifold objectives:

- Distributed monitoring of the network elements in charge of implementing QoS. For example, it should be possible to monitor more than one router simultaneously.
- Online visualization of retrieval data as well as storage of this data for future analysis.
- It should be able to provide data related to each DiffServ Traffic Class configured in the network element.
- It should be cross-platform.

In order to achieve the above objectives the paper is organized as follows :

- Section 2 describes the QoSpy implementation showing the architecture and its main components as well as the description of the developed application.
- Section 3 addresses the QoSpy evaluation showing the use for monitoring QoS parameters. Two

experimental testbeds were used for the evaluation. One at ITA (Instituto Tecnológico de Aeronáutica in São José dos Campos-Brazil) and the other at ANC (Advance Network Center of Salzburg Research in Austria).

- In the last section the final considerations about this work are presented, as well as ideas for future research in this area.

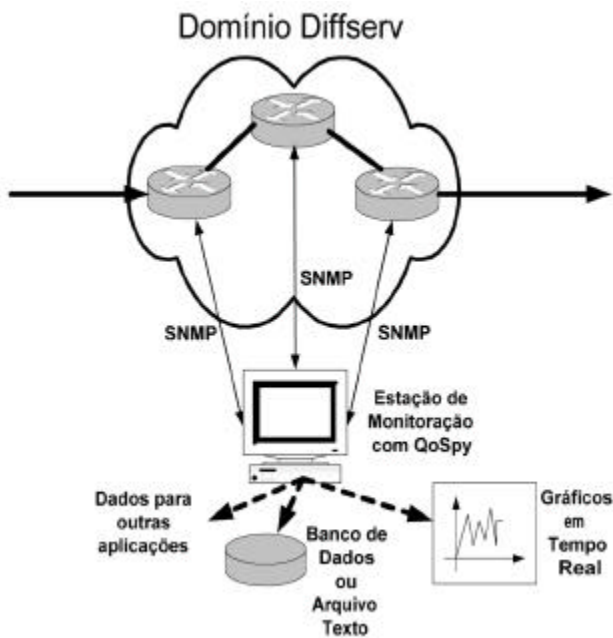


Figure 1 shows the simplicity of the implementation architecture used. To build the monitoring application, AdventNet SNMP API was used. The AdventNet product was chosen because it showed to be easier to learn, its cross-platforms features, and eases the creation of GUIs. The idea is to use the information provided by routers by SNMP to offer statistics about traffic classes and the QoS objects used to implement it. The monitoring tool can run in any station of the network and access data from any router that implements CISCO-CLASS-BASED-QOS-MIB[5]. This MIB is private and provides read access to QoS configuration and statistics information for Cisco platforms. Configuration information available through this MIB concerns the Traffic Classes, Policies, Match statements and other QoS objects used to implement the desired PHB(Per Hop Behavior). Statistics available through this MIB include summary counters per traffic class before and after any configured QoS policies are enforced. In addition, detailed feature-specific statistics are available.

## 2- QoSpy Implementation.

This section begins presenting the architecture and the main components(the MIB and the SNMP API). Afterwards, is given the application description.

### 2.1- The Architecture.

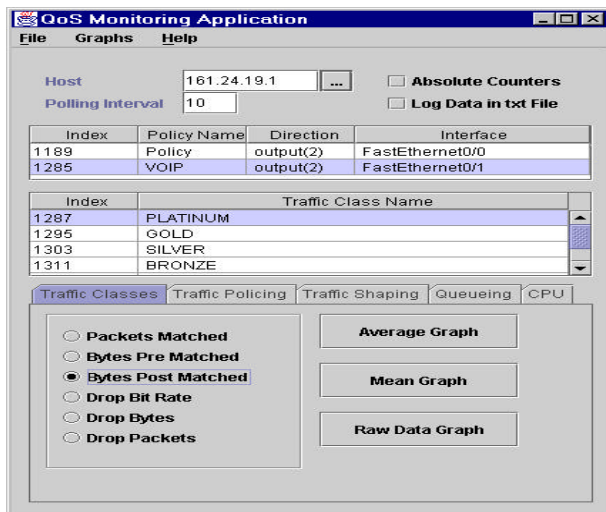
Figure 2 shows how this MIB is structured. It is made up by 20 tables that are used to characterize QoS objects. These objects are classified in five groups:

- Match Statement: the match criteria used to classify packets

- Class Map: a DiffServ traffic class containing one or more match statements to classify packets into different categories and to permit diverse PHBs.
- Feature Action: the QoS objects used to implement the desired PHBs. They include police, traffic-shaping, queueing, random detect and packet marking. After packets are classified, based on the match statement, these action are applied to each traffic class.
- Policy Map: a user-defined policy that associates each QoS action to user-defined traffic class ( Class Map).
- Service Policy: identifies the association of a policy map with an interface.

## 2.2-Application description.

The application is composed by one main window, which offer direct access to the monitoring functions, and a Graph Window that is used to plot the retrieved information from the router. The available data is arranged in different tabs depending on QoS elements to be monitored. Figure 3 shows the main window.



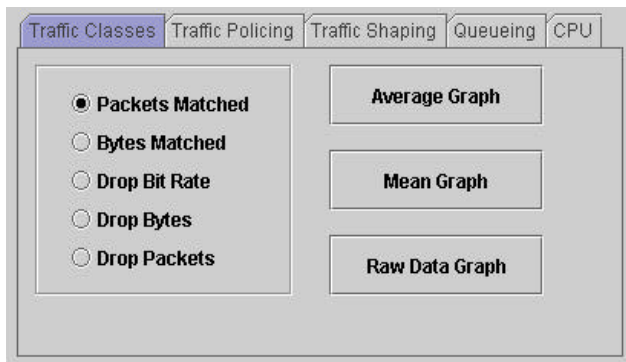
The main window has the router identification (IP Address), the list of QoS policies defined for that router, and the list of available Traffic classes for the chosen QoS policy.

The two check boxes on the right can be used to set some user preferences. If the absolute counters option is checked, the graph will show absolute counters instead of the difference of them. If log data in text file check box is checked, the monitored information will be saved in a text file in the application's main directory for later analysis.

Figure 3 shows a detailed view of the policies list. After choosing the desired policy, the traffic classes list will be updated with the QoS classes used with the policy chosen in polices list. The next step is to select one of the QoS data that can be monitored, which are grouped in five tabs:

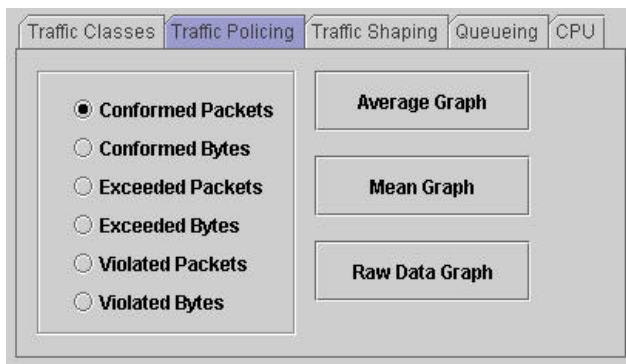
### a- Traffic Classes Statistics.

This data is important to monitor the state of the whole network. It helps, for example, to learn if a priority class is discarding more packets than expected or if the reserved resources are enough for each class. Thus, the configuration can be corrected for a better performance. Figure 4 shows the five different types of information about traffic classes that can be fetched with the monitoring application.



### b- Traffic Policing Statistics.

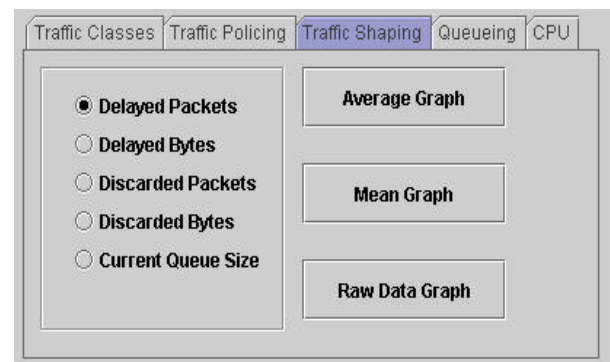
This data is related to the functioning of token buckets used for traffic policing. It is useful to check how much of the customer's traffic is conforming or not with the contract. Figure 5 shows the token bucket informations that can be fetched with the monitoring tool.



### c- Traffic Shaping Statistics.

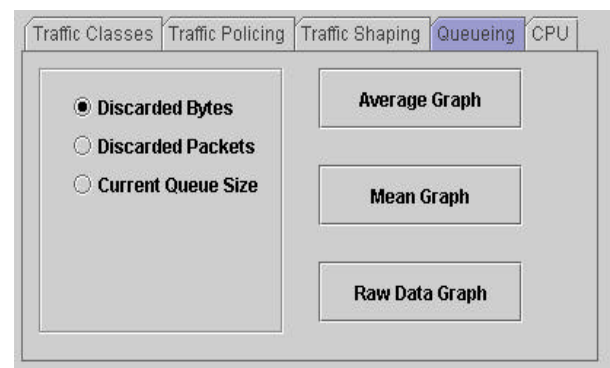
The Traffic Shaping Tab allows access to the token bucket's statistical information used to implement Traffic Shaping for QoS DiffServ Classes it allows to assess if the traffic entering a DiffServ is accord with the specified in the customer's contract. Thus, it is possible to know about packets/bytes delayed or dropped by traffic shaping

as well as the current queue size( see Figure 6).



### d- Queueing Statistics.

These statistics are related with the queues deployed by CBWFQ( Class Based Weighted Fair Queueing)[6] for each QoS traffic class that uses this feature. Figure 7 shows the three kinds of information that can be fetched. The queueing statistics will only be shown for that traffic classes that use CBWFQ.



### e- CPU Usage Statistics.

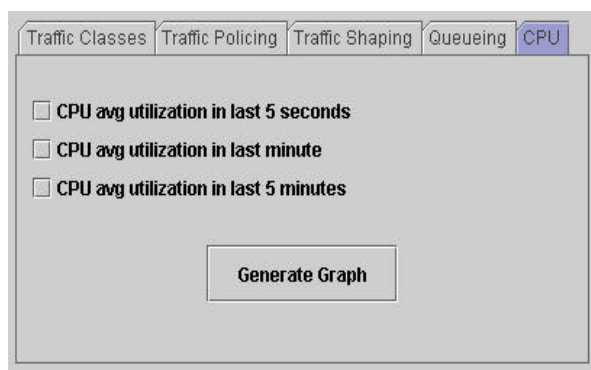
Besides fetching QoS-related statistics, the monitoring tool is also able to get CPU usage statistics from the chosen router. The information is fetched from the routers based on the CISCO-

PROCESS-MIB [5]. Three kinds of information can be obtained :

-CPU average utilization in the last 5 seconds.

-CPU average utilization in the last minute.

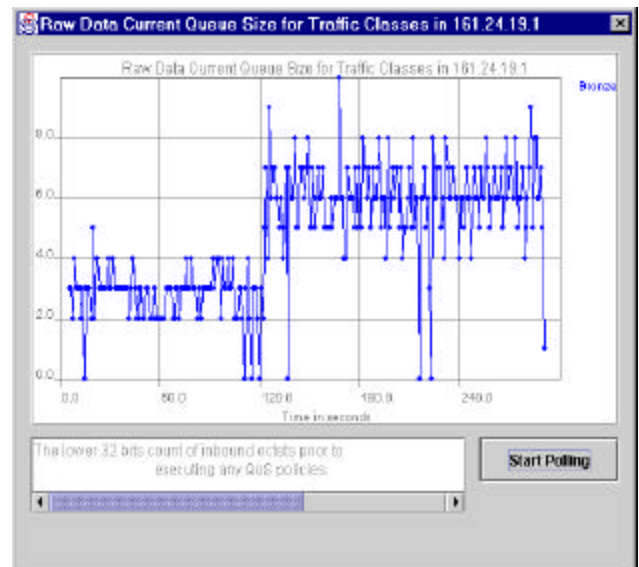
-CPU average utilization in the last 5 minutes. Although this information is not directly QoS-related, it is very important to assess the router working load caused by customer's traffic or even the monitoring traffic. Also , since QoS-related tasks are often computationally expensive and may result in over -utilization of the processor, this information can be used to assess the behavior of the router in implementing the chosen QoS policy. The measurements of the CPU utilization shows that is possible to use a 5 seconds polling interval without overloading the CPU with SNMP requests. Figure 8 shows the informations available from the CPU Usage Tab.



f- The Graph window.

This window is shown in Figure 9 and contains graphs about statistics for DiffServ classes for a specific router.

Three kinds of graph can be obtained: raw data, the mean and the average of the monitored data. The raw data represents the values obtained directly from the router. The mean is obtained calculating the mean value over two consecutive polling intervals. The average is the average of the mean values over the number of polls already done. The graph title provides the IP address of the network element being polled, the QoS Object being monitored and the traffic class.



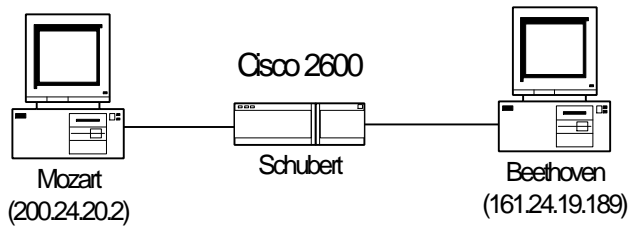
3- QoSpy evaluation.

This section evaluate the use of QoSpy for monitoring QoS parameters. Two testbeds were used. Both networks used are experimental. The first is our experimental tesbed in ITA. The second is an experimental testbed at Advance Network Center ( ANC) of Salzburg Research in Austria. The first use case shows that QoSpy can detect

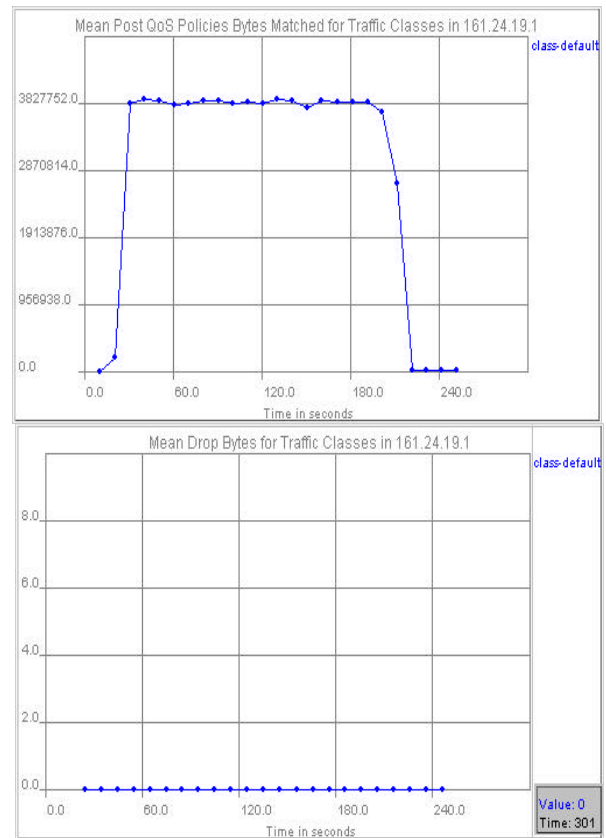
the influence of traffic policing in a router. The second and last use case shows how QoSpy can be used to detect degradation on routers.

### 3.1 Use case 1 : Traffic Policing.

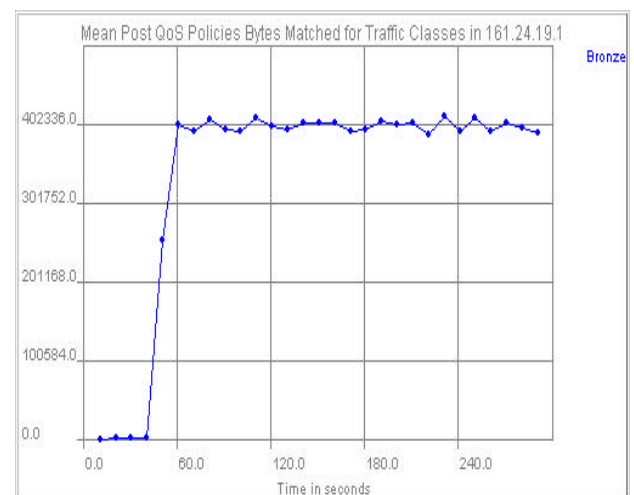
This use case intends to show how QoSpy is used to verify the influence of Traffic Policing on a DiffServ Traffic Class. For this experiment, our testbed was composed by two Pentium III machines connected through a Cisco router series 2600, as shown in figure 10.

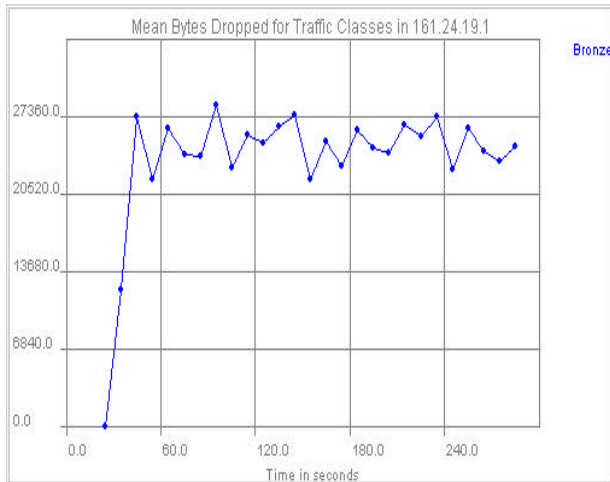


QoSpy was installed on both machines to monitor the router activities. As we could not set the DSCP value of the packets, we directed the flow to the best-effort class. First, the file transfer was done with no object applied to the class. The parameters Drop Packets and Packets Matched to the class were monitored. As can be seen in the screenshots (Figures 11 and 12), QoSpy detected packets for the class but no losses for the traffic, as expected.



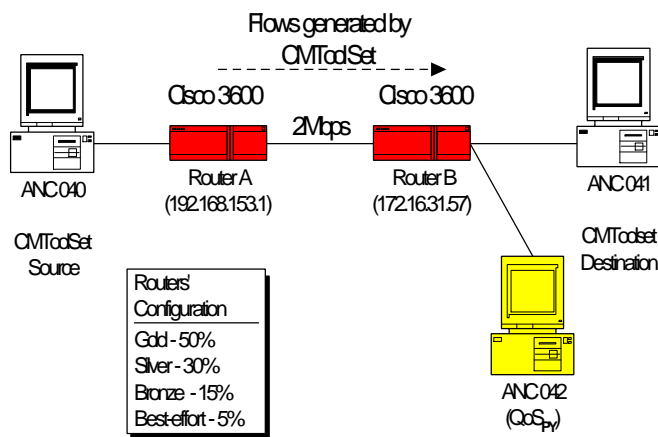
After, a Traffic Policing mechanism was applied to the class and the test was repeated. Screenshots in Figure 12 and 13 shows that QoSpy detected drop packet action and that less packets were admitted by the router by this time.





### 3.2 Use Case 2 : Detecting Service Degradation.

The last use case was performed at Salzburg Research testbed, which can be seen in Figure 14.



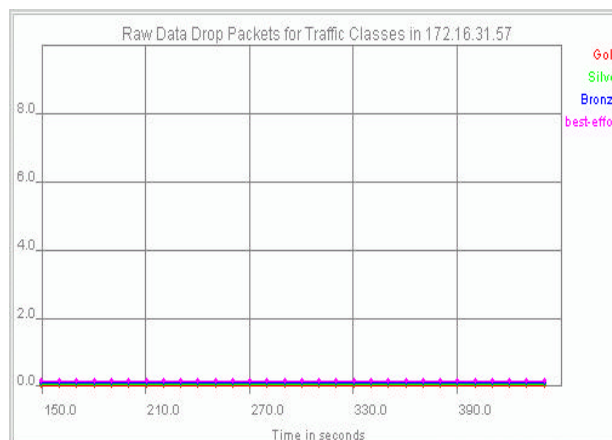
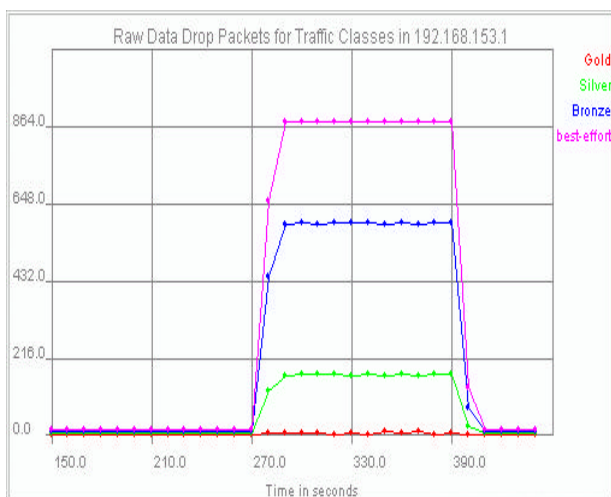
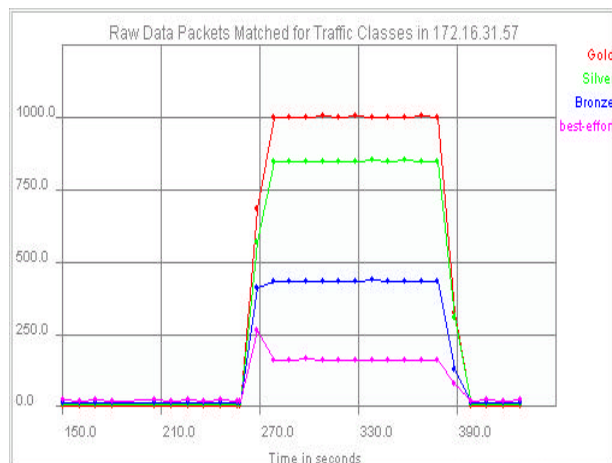
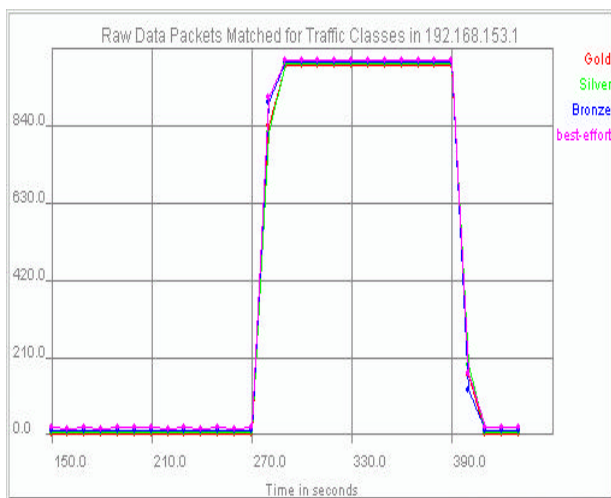
It is composed by two Cisco routers series 3600 connected by a serial link of 2Mbps. Each router has computers connected to it by means of Fast Ethernet interfaces. These computers are used as traffic sources and destinations. A traffic generator named, CMTToolset, was used to generate traffic for this test. This tool

was developed by ANC at Salzburg Research, and has been successfully used for tests that need traffic generation. It can generate many kinds of traffic with different throughput specifications. For this test we used a more complex configuration for the routers. It was composed by four QoS traffic classes: gold, silver, bronze and besteffort. The QoS mechanism used was CBWFQ. In both routers the classes were assigned the following percentage of bandwidth for each class :

- Gold : 50%
- Silver: 30%
- Bronze: 15%
- Best-effort : 5%

CMTToolset was used to generate four

UDP flows (one for each class). Each flow generates 1000-bytesize packets every 10 ms. Each flow is responsible for a throughput of 0,8 Mbps. This sum up 3,2 Mbps, which is more than the link capacity (2Mbps). The flows go from left to right in the testbed scheme. The result is shown in the figures below.



The first figure 15 shows the packets matched to each class. It shows the same curve since all flows are similar. The next figure 16 shows the CBWFQ result. Since the link is overloaded, it has to drop packets in order to match the router configuration. Most of dropping action is for best-effort since it has only 5% of bandwidth available (0,1 Mbps). No drops for Gold class since it has 50% of bandwidth or 1Mbps, which is enough to deal with the incoming traffic (0,8Mbps). The graph for right router packet matches

(Figure 17) shows, of course, more matches for gold since there we had no drops, and few matches for best-effort because many of them were already dropped in the left router. It also shows less packets for the other classes since they had some drops in left router. The packet drops graph (Figure 18) shows no drops, because the left router has done all the necessary job. Therefore, QoSpy showed that it is able to monitor more than one network element simultaneously. Consequently, it allowed us to assess which router was responsible for discarding packets, conforming to one of its objectives in section 1. Additionally, this example shows that QoSpy is able to monitor data for all traffic classes configured for a router.

#### 4- Conclusions.

The goal of this paper was to show the development and evaluation of a QoS monitoring tool for DiffServ-enabled networks, which was named QoSpy. The cross-platform monitoring application requirement was filled thanks to the use of Java as the programming language and AdventNet SNMP library for MIB access. As a result, QoSpy can be used in the most used OS platforms like Solaris, Windows and Linux.

The use cases carried out in section 3 showed that QoSpy is functional and able to report useful information about DiffServ Classes routers. However QoSpy it is not able to measure other

important parameters like delay and jitter. Perform distributed monitoring of delay and delay variation at intermediate points seems to be a challenging object of research. At the moment, the available measuring tools for these parameters are end-to-end monitoring tools. Research activities will be undertaken to implement the needed QoSpy functionalities to perform distributed monitoring of delay and delay variation.

#### References.

- [1] Durham D. , Yavatkar R. , “Inside the Internet’s Resource Reservation Protocol”  
John Wiley&Sons,Inc. , 1999.
- [2] Bernet Y., “Networking Quality of Service And Windows Operating Systems”  
New Riders , 2001.
- [3] Jiang Y. , Tham C-K. , Ko C-C. , “A QoS Distribution Monitoring Scheme for Performance Management of Multimedia Networks” Globecom ’99.
- [4] Stallings W. , “SNMP, SNMPv2, and RMON : The Practical Network Management Standards, ( 2d ed.),  
Reading, MA: Addison-Wesley, 1996.
- [5] CISCO. Cisco-class-based-qos-mib. 2001. Available at : <  
<ftp://ftp.cisco.com/pub/mibs/supportlists>>.
- [6] CISCO. IP Quality of Service. Cisco Press 2001.