Dynamic QoS Adaptation using COPS and Network Monitoring Feedback

Toufik Ahmed[1,2], Ahmed Mehaoua[1] and Raouf Boutaba[2]

[1] University of Versailles, CNRS-PRiSM Lab.
45 av. des Etats-Unis, 78000, Versailles, France
{tad, mea}@prism.uvsq.fr
[2] University of Waterloo, Dept. of Computer Science
200 University Avenue West, Waterloo,
Ont. N2L 3G1, Canada
{tad, rboutaba}@bbcr.uwaterloo.ca

Abstract. This paper presents an approach to handle out of profile traffic using Common Open Policy Service and network monitoring feedback. The proposed approach is based on monitoring and reporting information sent by bandwidth monitors installed on each node of a Diffserv Domain. A monitor interacts with Policy Decision Point. This later, depending on the network state, pushes policy decision rules to the Policy Enforcement Point in order to accept, remark or drop out-of-profile traffic dynamically. This allows a dynamic reallocation and management of network resources based on current network state and applications QoS requirements. An implementation and a performance evaluation of the dynamic QoS adaptation framework using a Java COPS and a Linux-based network testbed are also presented.

Keywords: IP Diffserv, QoS Adaptation, COPS, Network Monitoring.

1 Introduction

Recent works on IP Quality of Service (**QoS**) Management led to the development and standardization of enhanced protocols and services. The IETF has defined the Policy-based Network Management (**PBNM**) architecture to configure network services. Currently most efforts are focused on Differentiated Services (Diffserv) in the Internet. The goal of the policy-based network management is to enable network control and management on a high abstraction level by defining configuration rules called policies. Policies specify how a network node must be configured in vendor-independent, interoperable and scalable manner.

Diffserv architecture defines, at a lower level, four types of data-path elements: traffic classifiers, actions elements, meters and queuing elements [1]. Combining these elements into higher-level blocks creates a Traffic Condition Block (**TCB**), which can be managed by policy-based network management tools. The configuration of Diffserv TCB using PBNM involves the use of administratively prescribed rules that specify actions in response to defined criteria. All the information needed to perform this task such as profiles, user information, network configuration data, and IP infrastructure data such as network addresses and name server information are

stored in a policy repository. These configurations do not change frequently because they are not associated with specific application or traffic but with the network management. The more difficult part in the configuration is to have the traffic entering the network appropriately marked (audio, video and other data). Since, the user is signed up for the service, edge devices could be configured to mark user's traffic with the appropriate PHB. With a known IP address and/or IP port number, the administrator can specify a policy that refers to user application IP address and marks traffic coming from that address appropriately.

In the actual configuration, when the user signs up for a particular service, he/she must specify his/her traffic profile and the action that must be taken when the traffic exceed this predefined profile (out of profile traffic). Generally, the out of profile traffic is dropped or marked as best effort traffic. This model is static and does neither respond to application needs nor favor an optimal utilization of network resources.

In this paper a Diffserv QoS management is explored with the objective to overcome the limitations of the static model. In addition to the techniques described above (i.e., PBNM, TCB, etc.), network monitors are used to make the system reactive by making automatic and real-time decisions concerning out of profile traffic. An architectural model allowing the configuration and dynamic management of the Diffserv domain will be presented, experimented and evaluated.

The reminder of this paper is as follows. In section 2, we compare static and dynamic policy decision approaches and present our proposal, which is based on dynamic QoS adaptation through network resource monitoring and feedback signaling. Section 3 is devoted to the implementation of the proposed QoS network management framework. Performance evaluation and results analysis are discussed in Section 4. Finally, we conclude in Section 5.

2 Dynamic QoS Adaptation

2.1 Static Policy Decision

In the Diffserv architecture, a particular traffic receives a predefined treatment based on predefined policies. This treatment is interpreted as a particular PHB [2], [3]. This task is done by the TC (Traffic Control) function, which assigns the correct DSCP [4] for the client's traffic according to it SLA (Service Level Agreement). Recall that each client defines it requirements and these are translated into SLAs. The allocation of resources (QoS) still static and can lead to bandwidth wasting and starving clients.

Some algorithms such as *Time Sliding Window Three Colour Marker* (TSWTCM) [5] and a *Two Rate Three Color Marker* (TRTCM) [6] can be used to mark IP packets treated by the edge router with a Diffserv PHB. These algorithms meter the traffic stream and marks packets based on measured throughput.

To receive a particular treatment, the user must specify it profile **TSpec** (Traffic Specification). TSpec specifies the temporal properties of a traffic stream selected by a classifier. It provides rules for determining whether a particular packet is in profile or out of profile. The Meter uses a Token Bucket to control user traffic. The following is a non-exhaustive list of potential profile parameters:

1. *Peak rate p* in bits per sec (bps)

2. *Token bucket rate r* (bps),
3. *Bucket depth b* (bytes),

An *Excess Treatment* parameter describes how the service provider will process excess traffic, i.e. out of profile traffic. The process takes place after Traffic Conformance Testing. Excess traffic may be dropped, shaped and/or remarked. Depending on the particular treatment, more parameters may be required, e.g. the DSCP value in case of re-marking or the shapers buffer size for shaping. All these actions are decided once the network element is configured and are not changed over the time. Fig. 1 gives an example of how out of profile traffic is treated using static configuration. In this Figure, user sends traffic not conforming to his Traffic Specification. Edge router control this traffic by a token bucket. Non-conforming traffic will be dropped always.

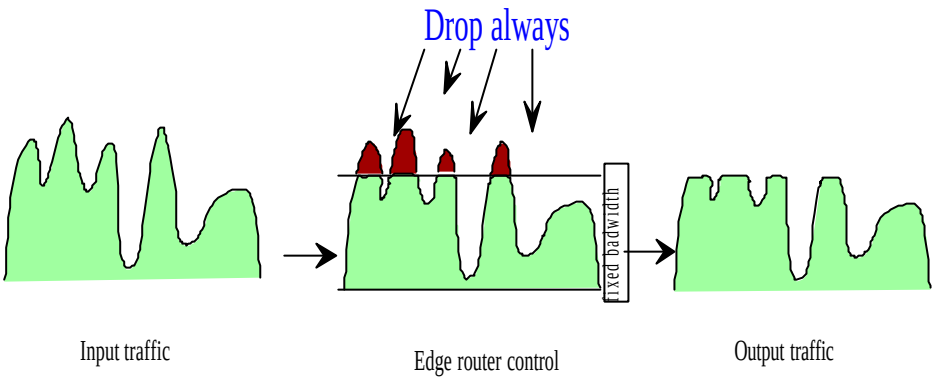


Fig. 1. Static Policy Decision

For this reason, there is a great need to control dynamically the action taken by the network element for more flexible resource allocation. For this, different conditioning actions may be performed on the in profile packets and out of profile packets or different accounting actions may be triggered dynamically according to current network state. Clearly, a more flexible resource allocation can be achieved by controlling dynamically network elements behavior.

2.2 Dynamic Policy Decision

In the static approach out of profile traffic is simply dropped, remarked or assigned a new profile. This decision is static and is taken once for all, i.e. when the network element is configured.

For example the *Policing Rule = drop out-of profile* packets can be applied to all the packets which are out of profile regardless of whether the network is capable or not to transmit this packet.

shows where we can dynamically decide what actions must be applied to out of profile packets. In contrast to the static approach, these actions vary according to the network state (network link load, traffic behavior, etc.).

2.3 Automatic Diffserv Domain Configuration

When a network element is started, its local PEP requests the PDP for all policies concerning Diffserv traffic marking using COPS (Common Open Policy Service) [7], [8], [9]. The policies sent by the PDP to the PEP, may concern entire router QoS configuration or a portion of it, as an updating of a Diffserv marking filter. The PDP may proactively provision the PEP reacting to external events generated by some monitors such as a bandwidth monitor.

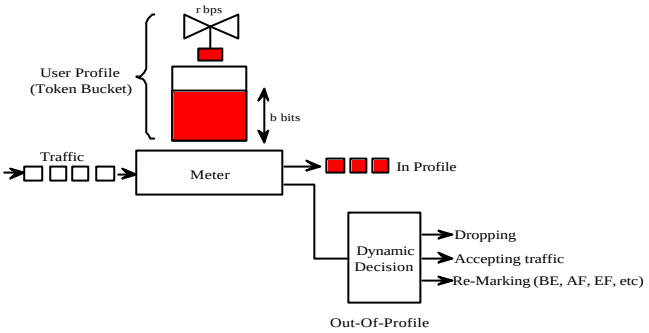


Fig. 2. Dynamic Decision

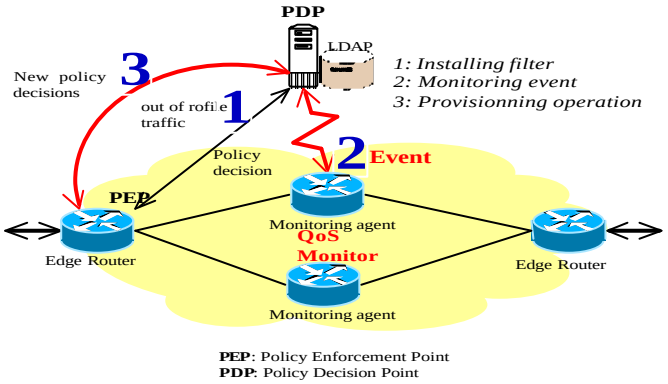


Fig. 3. COPS-PR with Monitoring Event

Fig. 3 shows the steps involved in the configuration of Diffserv domain. These steps are as follow:

– **Step 1**: When the edge router is started, the local PEP requests all policy decisions concerning Diffserv QoS Management (filtering, classes, queuing discipline, and actions for out of profile traffic). All incoming traffics are processed according to the pre-installed rules.

- **Step 2**: When the bandwidth monitor, installed on the core router, detects a significant change in the amount of available bandwidth, it triggers an external event reported to the PDP indicating the current bandwidth availability.
- **Step 3**: The PDP pushes to the edge router (PEP) an update of QoS Management decision.

These steps allow configuring correctly different policies related to the same traffic.

We introduce the following policy rule: let us **On event: If <profile> then <action>.**

A **Profile** is used to determine when a policy rule applies, for instance given pairs of source and destination addresses.

An **Action** is a performed by that the policy enforcement entity to traffic of a given profile. Examples of actions are marking, accepting or rejecting traffic.

Example of policy rules:

- Rule 1: Mark DSCP value EF on all packets with source addresses from 193.51.25.1 to 193.51.25.255 priority 0
- Rule 2: Mark DSCP value AF11 on all packets with destination address 200.200.200.100 priority 1

2.4 Example of Application

Assume, an audio application has subscribed to a particular Diffserv class (an Expedited Forwarding Class). Audio traffic is defined by a particular profile. In this example Diffserv class simply mean that the audio stream will be marked with the appropriate DSCP (EF PHB here). The Administrator of the Diffserv domain configures the environment to support the Gold, Silver, Bronze and other services. Such configuration can be done through a Bandwidth Broker.

Supporting different classes of service in the core network requires putting in place classifiers, which cause the devices to examine the Diffserv mark on the packet and then treat the traffic accordingly. These configurations do not change frequently because they are not associated with specific application or traffic but with the network management. Since, the application is signed up for the service, edge devices are configured to mark application's traffic with the appropriate PHB. Based on the IP address and/or the port number, the administrator can set a policy that marks traffic coming from that address with EF PHB.

In order for customized traffic going to audio application (e.g. feedback traffic, RTCP, client commands) to receive a Diffserv treatment, policy must be deployed to the opposite edge device of a Diffserv domain.

When the audio application starts sending the data, the edge router must ensure; (1) the data sent by the audio server does not exceed what the application has subscribe-to (SLA) and (2) marking conforming traffic (in profile traffic) with the appropriate PHB (EF PHB in our example). In case of receiving out of profile traffic, the edge router requests a decision from the PDP. Since the PDP knows the current network state - because it receives monitoring information from different monitors installed in the network, it decides a new policy rule, for example dropping, marking or accepting out of profile traffic. This decision varies according to current network state.

2.5 QoS Management Algorithm

We have configured 3 rules named Rule1, Rule2 and Rule3 to deal with out of profile traffic. The Policy Server can choose one rule among the several depending on information sent periodically by the monitors. The monitoring information concerns essentially the bandwidth usage of each link in the network. The calculation of shaped value of the bandwidth using *Exponentially Weighted Moving Average* (EWMA) is presented in section 2.6. Below in Fig. 4 our algorithm, this uses the predefined policy rules to make a decision depending on bandwidth usage in the network.

```
Initialization:
 Start Bandwidth Monitor Mi for each Router i to
 calculate the available bandwidth BWi
 Lambda ← 0.2      // Fixed value for historical data
 X ←50% link capacity    // Initial value of EWMA
 Min_th← 40% link capacity
 Max_th← 70% link capacity
Loop:
 BW ← max(BW1, BW2, …,BWi)  //EWMA available bandwidth
 X X←  (1-lambda) * BW + lambda * X
 if  X < Min_th then
  Rule1: Accept out-of-profile traffic
 else if Min_th<=X<Max_th then
  Rule2: Remark out-of-profile traffic with a new DSCP
 else Rule3: Drop out-of-profile Traffic
 End
```

Fig. 4. Example of a simple algorithm using policies.

2.6 Calculating bandwidth usage in the network links

Our algorithm uses a low-pass filter to calculate the bandwidth usage. Bursty traffic can cause a transient congestion. The bandwidth usage is not affected by this transient congestion since we shape this value. The low-pass filter is an exponential weighted moving average.

The EWMA *(Exponentially Weighted Moving Average)* Chart is used when it is desirable to detect out-of-control situations very quickly. It is an Exponential Smoothing technique that employs one exponential smoothing parameter to give more weight to recent observations and less weight to older observations and vice-versa.

When choosing l , it is recommended to use small values (such as 0.2) to detect small shifts and larger values (between 0.2 and 0.4) for larger shifts [10].

Policy decision depends on the EWMA statistic calculated by each network monitor and sent to the Policy Decision Point to be aggregated.

3 Implementation

Our prototype consists of three modules that perform Dynamic QoS adaptation in an administrative domain; these modules are Policy-based network management tool, Network Monitoring System, and Policy System (Policy Decision Point and Policy Enforcement Point). Fig. 5 shows the core components of our implementation.

3.1 Tool Manager

Our policy tool management is a Policy-based Web Bandwidth Broker. It consists essentially of a web interface installed in web application server. The administrator uses the web interface to configure the Diffserv domain and to enter new policy or to edit an old one. A Java Servlet engine is used to store all the information to a repository. We have used an OpenLDAP [11] server running on Linux. Other functions may be provided, such as validation, verification, conflict detection, etc. which are not yet available in our system.

In the top right of Fig. 5, a simple web-based interface of the bandwidth broker is shown. It illustrates the edge router configuration, specially the filter configuration and how setting PHB for the traffic entering the network.

3.2 Network Monitoring Agent

Network Monitoring provides a global network status in terms of resource availability and resource consumption, which is required for the management of the available bandwidth on the network. It is an application that tracks on live resources information in the network. A framework for supporting the traffic engineering of IP-based networks is presented in [12]. Different types monitoring measurements have been identified and are either **passive** or **active**. Passive measurement means that the statistics of the network element are maintained in the form of a Management Information Base (MIB), whereas in active measurement, test packets are injected into the network (like ping test) to gather information. Information collected about these packets are taken as representative of the behavior of the network. Metrics of this data are described in the framework presented in [13].

Our implementation consists of an agent written in Java which collects information on each interface of the router. The collected information consists of a real-time traffic flow measurement in input and output of each interface. This way, the agent augments the functionality of PEP by reporting monitoring information to the PDP in the form of COPS Report State Message. The PDP, when it detects a significant modification in the network state, delivers to the PEP a new policy decision in term of new policy rules. Decision-making is based on the algorithm described in Fig. 4.

3.3 Policy Management System

This system is composed of a PDP and a PEP communicating using COPS protocol. All system components are implemented in Java. The COPS-PR implementation is simplified to exchange policy rule between PDP and PEP.

Simplified COPS-PR implementation is used to exchange policy rules between the PDP and the PEP.

The PEP is associated with the interfaces to which the marking must be applied (edge router). It is notified when the policy changes (or is newly) by a COPS provisioning operation. The PEP receives the policy information and transforms it into a form suitable for the device, e.g. using a Linux Diffserv Traffic Control API. After this, all incoming packets to this device will be marked according to the new marking policy.

The PDP is responsible for decision making and uses for that the network monitoring agents. Our implementation is limited to one domain (there is no inter-domain communication).

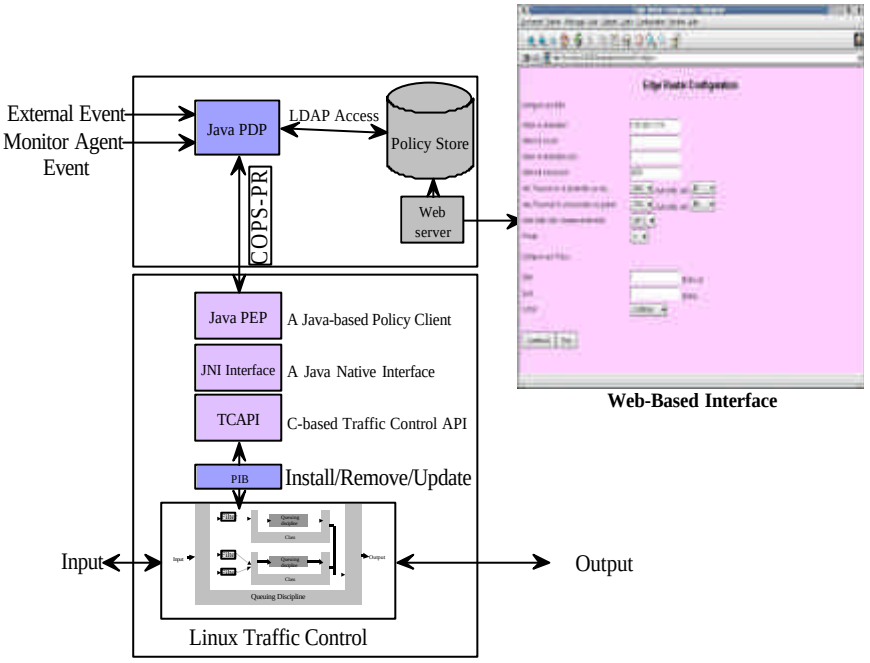


Fig. 5. Our System Implementation using Dynamic Policy-Based Management

4 Performance Evaluation and Results Analysis

The network administrator uses the PBNM tool (BB) to configure the edge and core routers according to a predefined set of policies. Suppose that the administrator's domain can handle EF, AF11 and BE class only. The administrator configures the filters using also PBNM tool. The task of the filter is to mark the traffic entering the network with the appropriate PHB according to user profile. At this step, the administrator chooses how to handle excess of traffic (out of profile traffic) by tuning two control thresholds (Min_th and Max_th).

4.1 Experimental Testbed

Fig. 6 depicts our experiments testbed. User transmits a customized traffic (audio traffic) across a Differentiated Services network. The network is composed of Diffserv capable routers. We use Linux-based IP routers with Diffserv implementation [14], [15]. The testbed is composed of two edge routers connecting by 10 Mb/s Ethernet links.

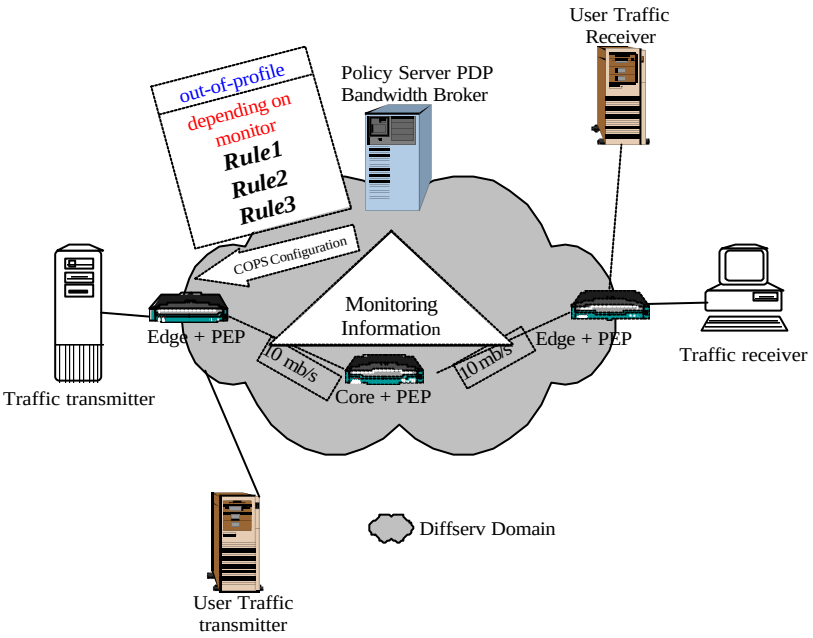


Fig. 6. Experimental Environment

By using our PBNM tool, we allocated 1.5Mbit/s for each AF class (i.e. AF1, 2, 3 and 4), all of which are bounded. We limit the amount of EF traffic to 15% of the bandwidth capacity rate, i.e. 1.5Mbit and we allocated 3.5Mbit for the best effort traffic that are allowed to borrow any available bandwidth. To get distinguish class of service, we used CBQ as our packet scheduler, which is an approach proposed in [16]. For CBQ a single set of mechanisms is proposed to implement link sharing and real-time services. In our implementation, CBQ is used to classify EF, AF, and BE traffic so that each user can get appropriate resources based on packet marking. The scheduler of the Diffserv core router employs GRED queuing discipline to support multiple drop priorities as required for the AF PHB group. One physical GRED queue is composed of multiple **VQs** (Virtual Queues). GRED can operate in ***RIO*** *(*RED with In/Out bit) *mode* [17], with coupled average queue estimates from the virtual queues, or in *standard mode* where each virtual queue has its own independent average queue estimate as required for RED [18]. In our testbed, we used GRED as the queuing

discipline for AF classes, since our marking algorithm takes into account these properties to give different level of QoS.

4.2 Performance Analysis

We loaded the network using *n* IP traffic generator. One traffic generator is composed of a traffic transmitter and a traffic receiver. The traffic transmitter generates a UDP packet of 1024 bytes with IP and UDP headers according to a Poisson distribution with parameter $l = 128$ packet/s that gives 1Mbit/s per traffic generator. In our test, and since our Ethernet links are 10 Mbit/s, we have taken *n=5, n=7 and n=10* in order to load the network differently each time. Each source can be either on or off during exponentially distribution on/off period with an average of $l_{on} = l_{off} = 1s$.

We compare the different network and traffic scenario when activating the algorithm and not activating the algorithm with an IP Diffserv network model.

Policing is performed at the edge of the network for a particular traffic, which is identified by a couple <IP_adr, Port_number>. Policy is determined by the traffic specification **Tspec** (traffic profile). Tspec takes the form of a token bucket (r,b) and the following optional parameters : a peak rate (p), a minimum policed unit (m), and a maximum datagram size (M).

The token bucket and peak rate parameters require that traffic obeys the rule that over all time periods, the amount of data sent cannot exceed M+min[pT, rT+b-M] [19]. M is the maximum datagram size, and T is the length of time period. Datagrams which arrive at an element and cause a violation of the M+min[pT, rT+b-M] bound are considered out of profile (non-conformant) and require a decision from the PDP.

In our experiment, we set the parameters for the token bucket to *r=1Mbit/s* and *b=2K*, for user traffic. This means that this traffic must not exceed 1Mbit/s.

For testing purposes, we transmit an out of profile traffic (not conform to TSpec). This traffic is at a constant bit rate of 1.5 Mbit/s. The token buckets accept only 1Mbit/s, therefore, the 0.5 Mbit/s are considered out of profile. The in profile traffic will be marked with EF PHB whereas the out of profile traffic will be marked either by EF or AF11 or dropped (according to our predefined policy).

Fig. 7 (a) shows the bottleneck link load during the period of the experiment (180s). This load represents the traffic sent from the n traffic generators to the receivers. This measure has been taken from the ingress interface of the core router. During to first 60 seconds there are only n=5 traffic generators that can be either on or off. From time 60s to 120s there are n=7 traffic generators. In the last 60 second (from 120s to 180s) the number of the traffic generators are *n=10*.

The PDP makes the decision according to the smoothing value of the bandwidth usage (i.e., EWMA). This decision is a policy rule sent directly to the edge router of the Diffserv network.

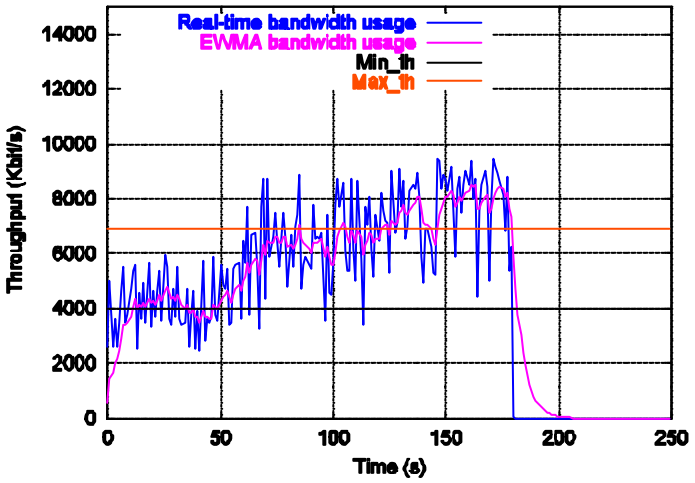
In our experiments, we set the value of Min_th=4Mbit and the value of Max_th=7Mbit. The read time of the bandwidth usage performed by the bandwidth agent is set to 1 second.

The events sent by the PDP are listed below with the corresponding timestamps (see Table 1).

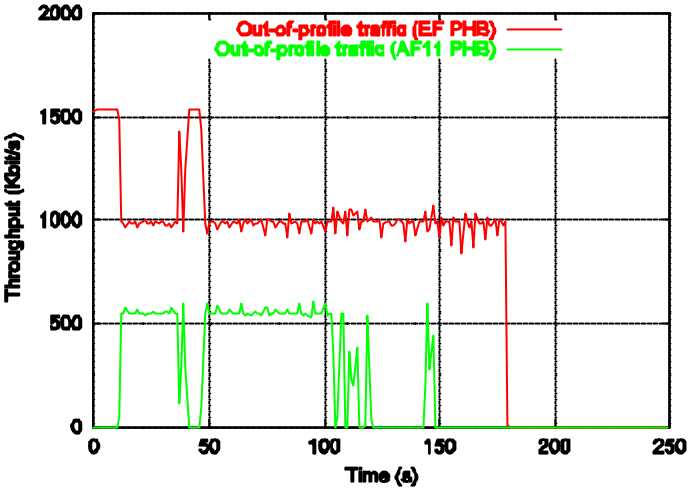
Table 1: List of policies sent by the PDP

TIME (SECOND)	ACTION TAKEN BY THE EDGE ROUTER (POLICY)
0	Rule1: Accept out of profile traffic (EF traffic)
12	Rule2: Remark out of profile traffic with AF11
37	Rule1: Accept out of profile traffic (EF traffic)
38	Rule2: Remark out of profile traffic with AF11
40	Rule1: Accept out of profile traffic (EF traffic)
47	Rule1: Accept out of profile traffic (EF traffic)
103	Rule3: Drop out of profile Traffic
105	Rule2: Remark out of profile traffic with AF11
107	Rule3: Drop out of profile Traffic
109	Rule2: Remark out of profile traffic with AF11
110	Rule3: Drop out of profile Traffic
111	Rule2: Remark out of profile traffic with AF11
112	Rule3: Drop out of profile Traffic
116	Rule2: Remark out of profile traffic with AF11
117	Rule3: Drop out of profile Traffic
141	Rule2: Remark out of profile traffic with AF11
144	Rule3: Drop out of profile Traffic
177	Rule2: Remark out of profile traffic with AF11
179	Rule1: Accept out of profile traffic (EF traffic)

These events show how traffic is subject to a dynamic behavior in the network. This is an interesting function, since it allows an Internet Service Provider making new strategies of traffic engineering easily.



(a)



(b)

Fig. 7. **(a) Bottleneck Link Usage**

(b) Received Audio Traffic with different PHB Color

Fig. 7 (b) shows the received audio traffic with the different PHB colors. In-profile traffic (1Mbits) is always marked as EF whereas out-of-profile traffic (0,5 Mbits) is dynamically accepted as EF, as AF11, or dropped.

The events shown in the Table represent the time at which the policy is sent from the PDP to the edge router. This later updates traffic policing to reflect this change. For example in the first 60 second, bottleneck link is under load (X <Min_th), so the edge router can accept out of profile traffic as shown in Fig. 7 (b). In the next 60 second (from time 60 to 120 s), load is between Min_th and Max_th, so we accept out of profile but with remarking policy. From time 120 to 180 s, bottleneck link is congestionned, in this case, out of profile traffic is dropped. See Fig. 7 (b) for more details.

5 Conclusion

This paper address the issue of out-of-profile traffic in a Diffserv network. It describes our proposal of using network monitoring feedback and policy decision point. The collected monitoring information is used to manage and to adapt dynamically QoS parameters for user traffic. The example configuration rules described in our testbed clearly demonstrate the advantage of using our proposed resource network management framework. Our system involves a policy-based management system to achieve a more dynamic network behavior in handling user traffic.

Several issues arise when using dynamic control decisions to handle out-of-profile traffic. One problem is the pricing and charging schemes in use: Who pays for the service (out-of-profile traffic), the sender or the receiver ? More work has to be done

in order to define accurately the amount of traffic that excesses the profile in order to establish a payment scheme. Also, time-scale measurement of the PDP response is important and should be evaluated in future work.

References

1. S. Blake, D. Black M. Carlson,E. Davies, Z. Wang, W. Weiss "RFC 2475: An Architecture for Differentiated Services", December 1998.
2. V. Jacobson, K. Nichols, K.Poduri "RFC 2598 An Expedited Forwarding PHB", June 1999.
3. J.Heinanen, , F.Baker , W. Weiss, J. Wroclawsk "RFC 2597 : Assured Forwarding PHB Group", June 1999.
4. K. Nichols, S. Blake, F. Baker, D. Black "RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", December 1998.
5. W. Fang, Seddigh, B. Nandy "RFC2859 - A Time Sliding Window Three Colour Marker (TSWTCM)", June 2000.
6. J. Heinanen, R. Guerin "RFC2698 - A Two Rate Three Color Marker (TRTCM)", September 1999.
7. D. Durham, Ed, J. Boyle, R. Cohen, S. Herzog, R. Rajan,w, A. Sastry "RFC 2748: The COPS (Common Open Policy Service) Protocol", January 2000.
8. S. Herzog, Ed., J. Boyle, R. Cohen, D. Durham, R. Rajan, A. Sastry "RFC 2749: COPS usage for RSVP", January 2000.
9. K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, A. Smith "RFC 3084: COPS Usage for Policy Provisioning (COPS-PR)", March 2001.
10. J. Stuart Hunter. "The Exponentially Weighted Moving Average" J Quality Technology, Vol. 18, No. 4, pp. 203-207, 1986.
11. "OpenLDAP software" available at http://www.openldap.org/
12. W.Lai, B.Christian, R.Tibbs, S.Berghe "Framework for Internet Traffic Engineering Measurement" Internet draft, Work in progress, November 2001.
13. V. Paxson, G. Almes, J. Mahdavi, M. Mathis "RFC2330: Framework for IP Performance Metrics", May 1998.
14. Werner Almesberger "Differentiated Services on Linux" Home Page http://diffserv.sourceforge.net/
15. Werner Almesberger, Jamal Hadi Salim, Alexey Kuznetsov "Differentiated Services on Linux", Work in progress, June 1999.
16. S. Floyd et al. "Link-sharing and Resource Management Models for Packet Networks" IEEE/ACM Transactions on Networking, Vol. 3, No. 4, pp. 365-386, August 1995.
17. David D. Clark and Wenjia Fang "Explicit Allocation of Best Effort Packet Delivery Service" ACM Transactions on Networking, pp. 362-373, August 1998.
18. Sally Floyd and Van Jacobson "Random early detection gateways for congestion avoidance", IEEE/ACM Transactions on Networking. Vol. 1, no. 4, pp. 397-413, August 1993.
19. L. Georgiadis, R. Gu_erin, V. Peris and R. Rajan "Efficient Support of Delay and Rate Guarantees in an Internet" in ACM SIGCOMM, volume 26, number 4, October 1996.