An example of dynamic QoS negotiation

P. Cremonese *, S.Giordano **

*p.cremonese@finsiel.it Finsiel S.p.A via Matteucci 34 Pisa Italy

ph +39 050 968516 fax +39 050 968525

** silvia.giordano@epfl.ch EPFL Lausanne Switzerland

ph +41 21 69356748 fax +41 21 693 6614

Keywords: RSVP, RVBR, QoS

Abstract

The traffic generated by multimedia applications presents a high degree of burstiness that can be hardly described by a static set of traffic parameters. The following paper presents a dynamic QoS negotiation scheme applied to a video streaming application. In applications that uses RSVP, the dynamic and efficient usage of the resources can be reached with the introduction of the renegotiable variable bit rate (RVBR) service, which is based on the renegotiation of the traffic specification. In this paper we describe and discuss the RVBR service and how it applies to resource reservation for Internet traffic with RSVP. For that we propose an architecture design that we evaluate by accomplishing a prototype implementation, whose performance are measured with real MPEG2 video traces. The results we obtained indicate that renegotiation is an efficient mechanism to accommodate traffic fluctuations over the burst time-scale, and that RVBR service can be easily implemented, to this aim, in real applications, using available technology.

1. Introduction

Future applications make use of different technologies as voice, data, and video. These multimedia applications require, in many cases, better service than a best effort service. This service is generally expressed in terms of Quality of Service (QoS), whereas network efficiency depends crucially on the degree of resources sharing inside the network.

To achieve both the applications' QoS requirements and network resources efficiency is extremely important for several reasons, for instance, network dimensioning or traffic charging.

We analyse how to achieve these goals at the source node where a traffic profile is negotiated with the network and the traffic is shaped according to the contract. In many situations, a single traffic profile negotiation can lead to an excessive usage of resources and unacceptable performance. In these cases a straightforward solution is to renegotiate the traffic profile during the lifetime of the connection.

The introduction of the the renegotiable variable bit rate (RVBR) service [1], [2] at application layer is assumed to

simplify and generalise this task. Whenever renegotiation is taking place, the RVBR scheme generates the traffic specification that conforms to the real demand, in order to reallocate the network resources in an optimal way while guaranteeing QoS to the traffic flows. RVBR service uses the knowledge of the past status of the system and the profile of the traffic expected in the near future, which can be either pre-recorded or known by means of exact prediction.

This scheme suits perfectly the dynamics of the traffic generated by multimedia application. Moreover it naturally integrates with the soft state mechanism of Resource ReSerVation Protocol (RSVP) [6], the mechanism for Resource Reservation used in the Internet. RVBR can be used to allow an application using RSVP protocol with Int-Serv traffic specification, not only to specify the traffic for the initial negotiation, but also to to find the optimal *Tspec* for the next renegotiation. In fact, with RSVP as reservation protocol, the reservation has to be periodically refreshed. Therefore the *Tspec* needs to be reissued at each renegotiation time. There is no additional signaling cost in applying a *Tspec* renegotiation at that point, even if there is some computational overhead due to the computation of the new parameters, or to the call admission control, etc. It is important to note here that, contrary to the negotiation of a new connection, with the renegotiation the reservation is never interrupted.

We consider the RSVP with Controlled-Load [9] (CL) service case study, and we describe the implementation design of a video streaming application, which is then implemented in a prototype, whose performance are measured with MPEG2 video traces. We report the results of the conducted trials, which evaluated the RVBR Service for multimedia IP traffic with RSVP.

The rest of the paper is organised as follows. In the next section we give an overview of the analitical model of RVBR and we illustrate how the problem of finding the optimal reallocation parameters has tackled. We also show how RVBR can be used for applications that use RSVP with CL. In Section 3 we introduce an approximation to RVBR in order to be more easily integrated in applications. In Section 4 we propose an architecture design that we evaluate by accomplishing a prototype implementation, whose performance are measured with real MPEG2 video traces. These results, which show the benefits of renegotiation , are presented and discussed in Section 5. Final discussion and future work are given in the conclusion section.

2. Resource Renegotiation: RVBR Service

The renegotiable variable bit rate (RVBR) service, can be used by an application to find, at any renegotiation, the parameters for renegotiate the RSVP connection, when the input traffic is known. RVBR service is based on the network calculus [12] definition of the *time varying leaky bucket shapers*; such shapers are defined by a fixed numbers of leaky buckets, whose parameters (rate and bucket size) are changed at specific transition moments. RVBR assumes that the bucket levels are kept unchanged at those transition moments.

RVBR service uses the knowledge of the past status of the system and the profile of the traffic expected in the near future, which can be either pre-recorded or known by means of exact prediction. This scheme suits perfectly the dynamics of the traffic generated by multimedia applications with pre-recorded traffic. Moreover it naturally integrates with the soft state mechanism of RSVP, which allows for renegotiating the resources.

1. Overview of RVBR Service

We first recall the characterisation of the RVBR service in terms of input and output functions as given in [2].

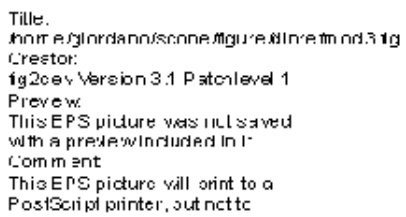

Figure 1 RVBR reference configuration

There is a renegotiable leaky bucket specification (with rate *r* and depth *b*) plus a fixed size buffer *X* drained at maximum at renegotiable peak rate *p*. The elements of a RVBR source, as illustrated in Figure 1**,** are a renegotiable leaky bucket specification (with rate *r* and depth *b*) plus a fixed size buffer *X* drained at maximum at renegotiable peak rate *p*. In [2] the RVBR service is described with two leaky bucket specifications. In the case of RSVP the bucket associated to the peak *p* is the MTU size, hence it is fixed. We further assume it equal to zero to simplify the computation, given that this is not a limitation.

The observation time is divided into intervals, and $I_i = [t_i, t_{i+1}]$ represents the *i-th* interval. Inside each interval the system does not change. The parameters of the RVBR service in I_i are indicated with (p_i, ri, b_i).

The RVBR service is completely defined by:

- the time instants t_i at which the parameters change
- the RVBR parameters (p_i, r_i, b_i), for each interval I_i
- the fixed shaping buffer capacity *X*

A RVBR source cannot send more than the traffic specified by the shaping function s_i, defined as

$$s_i(u) = \min (p_i * u , r_i * u + b_i)$$

Moreover the RVBR service, at the transient times t_i between two adjacent intervals, keeps the level of the buckets and restarts from that level at the next interval. The justification of this choice can be found in [2]. Therefore there is another function, resulting from taking into account the bucket level q(t), which limit the traffic in I_i

$$s^0_i(u) = \min (p_i * u, r_i * t\, u + b_i - q(t_i))$$

If we indicate with the function R(t) the amount of traffic that has entered in the system in time interval [0,t], the resulting output R*(t) is given by [2]

$$R^*(t) = \min (s^0_i(t-t_i) + R^*(t_i), \inf_s (s_i(t-s) + R(s))$$

1. Optimisation of the RVBR parameters

This input-output characterisation of the RVBR service, is further used to solve the problem of finding, at any renegotiation, the optimal s_i to negotiate with the network. This problem is well know to have no trivial solution. For example some input traffic could be specified from a large r_i and a small b_i as well as from a small r_i and a large b_i. In [2] the authors proposes different algorithms that solve the optimisation problem for some specific cost functions, which represent the cost of the traffic specification to the network. Those algorithms are based on the exact knowledge of the input traffic (pre-recorded or know by means of an exact estimation).

In particular, the algorithm *localOptimum*, which finds the optimal solution when the choice of the network is driven by a linear cost function , was used to perform simulation of Internet traffic that takes the form of IntServ specification with RSVP reservation. In RSVP the sender sends a PATH message with a *Tspec* object which characterises the traffic it is willing to send. If we consider a network that provides a service as specified for the Controlled Load service, the *Tspec* takes the form of a double bucket specification as given by the RVBR service. In fact, with CL service there is a peak rate *p* and a leaky bucket specification with rate *r* and bucket size *b*. This corresponds to the parameter of RVBR service. Additionally, with CL service, there is a minimum policed unit *m* and a maximum packet size *M*, which are assumed to be fixed and thus ignored.

1. RVBR when the traffic is specified by its arrival curve

localOptimum algorithm uses functions that require the knowledge of the exact traffic. In the real case of video stream applications, the module that implements RVBR has to access information related to the network and, therefore, even if the traffic is prerecorded and stored, it is not reasonable to have access to the exact traffic.

To the aim of using the localOptimum algorithm in a real application, we propose an approximation to some functions, which originally work with the exact traffic, in order to work with a smaller and less precise information: the exact traffic R(t) for t in I_i is substituted by upper bound functions. We introduce the function:

$$a_i(u)=\min (p^a_i * u, r^a_i * t\, u + b^a_i)$$

where

$$p^a_i = sup_{t,s} (R(t)-R(s))$$

$$(t-s)$$

$$r^a_i = (R(t_{i+1})-R(t_i))$$

$$(t_{i+1} - t_i)$$

$$b^a_i = S_t [R(t)- r^a_i * t]^+$$

and a second function that takes in account the traffic $q(t_i)$ that is the bucket at the transient period.

$$a^0_i(u)=\min (p^{a0}_i * u, r^a_i * t\, u + b^a_i - q(t_i))$$

where

$$p^{a0}_i = sup_t (R(t)-R(t_i))$$

$$(t- t_i)$$

These functions are arrival curves [12] of R(t), i.e. upper bounds to the traffic R(t). With the introduction of a_i and a^0_i we can approximate the function b_i and the optimal peak rate p_i that in the RVBR are originally computed from the exact traffic R(t).

Therefore, indicating with $w(t_i)$ the backlog in the shaping buffer at time t_i, the function b_i is given by

$$b_i(s)=\max ((a_i(s), a^0 i(s) + w(t_i)+q(t_i)))$$

and the minimum p_i by

$$p_i = \max\left(\sup_s (a_i(s) - X) / s \,,\ \sup_s (a^0_i(s) - X + w(t_i)) / s \right)$$

we can use for our prototype the algorthm for RVBR as defined in [1] and [2]:

Algorithm 1 localOptimum1$(X, \{R(t)\}_{t\in I}, , b_{max}, r_{max}, u, w(t_i), q(t_i), t_{i+1})$

if $b_{max} < \sup_{s\in I}\{\beta_i(s) - r_{max}\cdot s - X\}$ then there is no feasible solution;

else {

$$p_i = \max\left(\sup_{0\le s<(t_{i-1}-t_i} \frac{\alpha_i(s) - X}{s} ; \sup_{0\le s\le(t_{i+1}-t_i} \frac{\alpha^0_i(s) - X + w(t_i)}{s} \right)$$

if $u \le 0$ then {

$x_0 = min(r_{max}, p_i)$;

}

else {

$$x_0 = \sup_{s\in I} \frac{\beta_i(s) - \beta_i(u)}{s-u};$$

$$x_A = \sup_{s\subset I, s>0} \frac{\beta_i(s) - X - b_{max}}{s};$$

$$x_B = \sup_{s\in I, s>0} \frac{\beta_i(s) - X}{s};$$

if $(x_0 \ge \min(x_B, r_{max}, p_i))$ then $x_0 = \min(x_B, r_{max}, p_i)$;

else if $(x_0 \le x_A)$ then $x_0 = x_A$;

}

$r_i = x_0$;

$b_i = \sup_{s\subset I}\{\beta_i(s) - X - s\cdot x_0\}$;

}

a_i and a^0_i can be used in a real implementation, because computed with only four parameters: (p^a_i, r^a_i, b^a_i, p^{a0}_i). *These parameters can be easily stored and passed from the application level to the RVBR module*

2. Prototype description

In this section we present the design of the prototype application providing RVBR features via RSVP we implemented under Microsoft NT 4.0 with RSVP by Intel. The prototype realises a client-server application for data-transfer regulated by pre-defined temporisation. The server is composed by the following modules: the Application module, the RSVP Daemon, the Data Pump, the Network Module (IP, UDP, TCP). A graphical representation is given in **Figure 2**

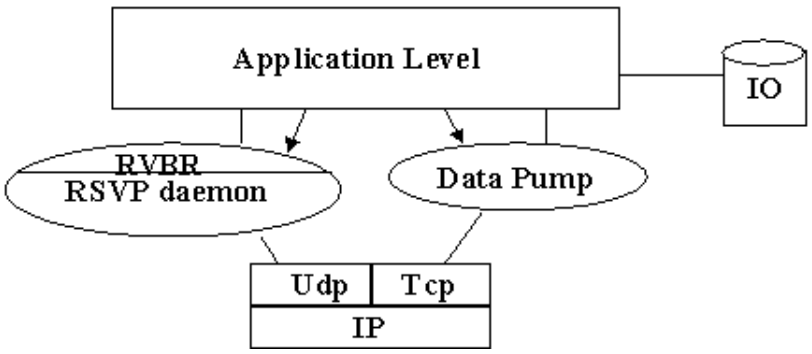

Figure 2 Prototype Architecture: Server

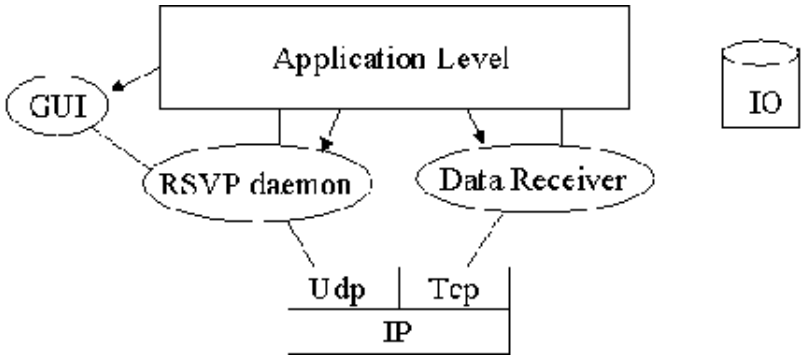

Figure 3 Prototype Architecture: Client

The client has the following module: the Application module, the RSVP Daemon, the Data Receiver, the Network Module (IP, UDP, TCP) and the Graphical User Interface. A graphical representation of the client is given in **Figure 3** .

The behaviour of this prototype is illustrated in **Figure 4**:

Server: The server waits for data request on a predefined and known TCP port. This request contains information about file to be transferred. The server uses this information to access a descriptor that contains the location of the requested file and the related QoS Information. The server computes the new QoS according to mechanism described in Section 4 and asks for reservation to the client sending the *Tspec* packet. Therefore, the following information are provided available to the server: the average rate r^{a}_{i},, *the peak rate* p^{a}_{i} , *the transmission length, the burst size* b^{a}_{i} , *the max slope* p^{a0}_{i} , *the reallocation time ReallocationTime, and the service constraint.*

If the reservation successes, it activates the RSVP daemon with the list of the rest of QoS descriptors, in the other case it activates only the data pump. The RSVP daemon sends a new Tspec according to the local refresh time defined for the soft-state and reallocation time define in the QoS descriptor. It sends a new Tspec every T_{int} *seconds where* $T_{int} = min(ReallocationTime, RefreshTime)$*. The new Tspec contains the new QoS computed by RVBR (if it is needed) or the old one if RefreshTime expires. In the second case the ReallocationTime must be updated.*

When the transmission ends the RSVP daemon closes the RSVP session on the server side.

The Data Pump sends data according to the temporisation defined for the special medium. The input file is composed by a list of packet-dimension with the related time-stamp for sending. The Data Pump generates a data-packet according to the required dimension and sends it according to the required time-stamp.

Client: *The client asks for the requested file and waits for the PATH message (containing the Tspec). On this basis it asks for reservation sending the RESV message. If the phase (PATH-RESV) successes, it activates the RSVP daemon, the Data Receiver module and the GUI. In the other case it activates only Data Receiver module.*

- *The GUI module is a window that allows interrogating the RSVP daemon about the resources allocated.*
- *The RSVP daemon waits for change the reservation (Tspec) and sends the new RESV. It tears down the RSVP connection if it receives a PATH_TEAR message.*
- *The Data Receiver builds the new file storing information received from the Data Pump on the server side in a location defined from the user.*

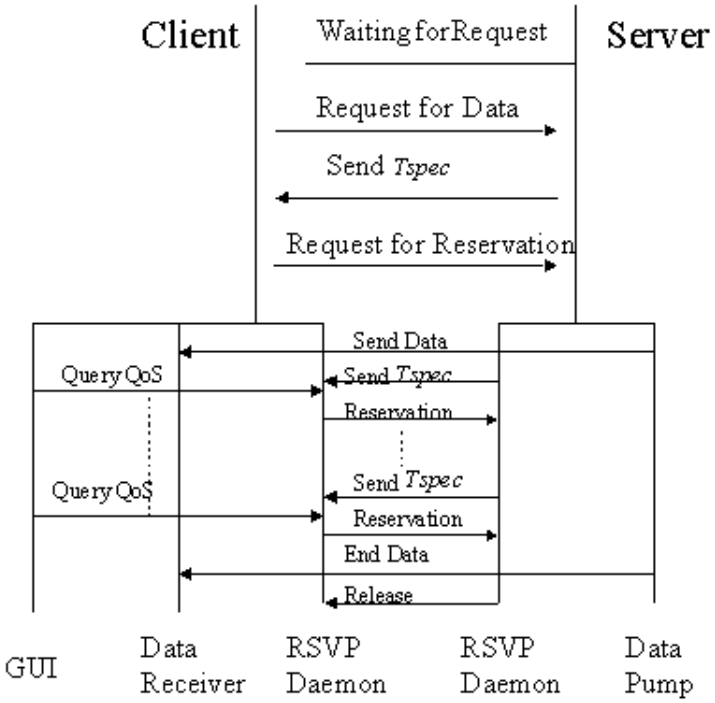

Figure 4 Prototype behaviour

1. Upper bound to the losses for the approximated version of the RVBR Service

In our trials, we use a 4000 frame-long sequence that conforms to the ITU-R 601 format (720*576 at 25 fps). The sequence is composed of several video scenes that differ in terms of spatial and temporal complexities. It has been encoded in an open-loop variable bit rate (OL-VBR) mode, as interlaced video, with a structure of 11 images between each pair of I-pictures and 2 B-pictures between every reference picture. For this purpose, the widely accepted TM5 video encoder [11] has been utilised.

Our trials are aimed at verifying the ability of the RVBR service to provide a better allocation of the resources in a real network and evaluating the overhead (time consuming) introduced by the support of renegotiation.

The trials have been performed between two PCs connected to a shared LAN. The communication between PCs does not go through any router. The trials have been performed varying network parameters and reallocation time: in this context we have identified two network configurations related to a shared Ethernet with medium load and a switched Ethernet (it has been simulated performing the related trials on the unloaded Ethernet). Limitations on *BucketDepth* b_i derive from the buffer capacity of each NIC (256KB).

The following figures show a comparison of allocated resources varying the reallocation time. The graphic in **Figure 5** is related to *BucketRate* r_i and *PeakRate* p_i allocation with reallocation time respectively 30 seconds, 60 seconds and without reallocation in the case of a shared Ethernet. In the following we will indicate as *short reallocation* the one at 30 seconds, *long reallocation* at 60 seconds and *legacy* the one without reallocation. In this case (shared Ethernet) we have limited the bucket capacity to half of the NIC capacity to take into account the problems deriving usually from a legacy Ethernet. The line related to Peak rate is constant in the case of 60 seconds and no reallocation while it is a piece wise in the other case (yellow) and it lies under the others. We can observe that the peak value is pretty constant in the case of long period and it is constant (of course) in the legacy case. The short reallocation requires a narrow bandwidth with a peak value always under the others.

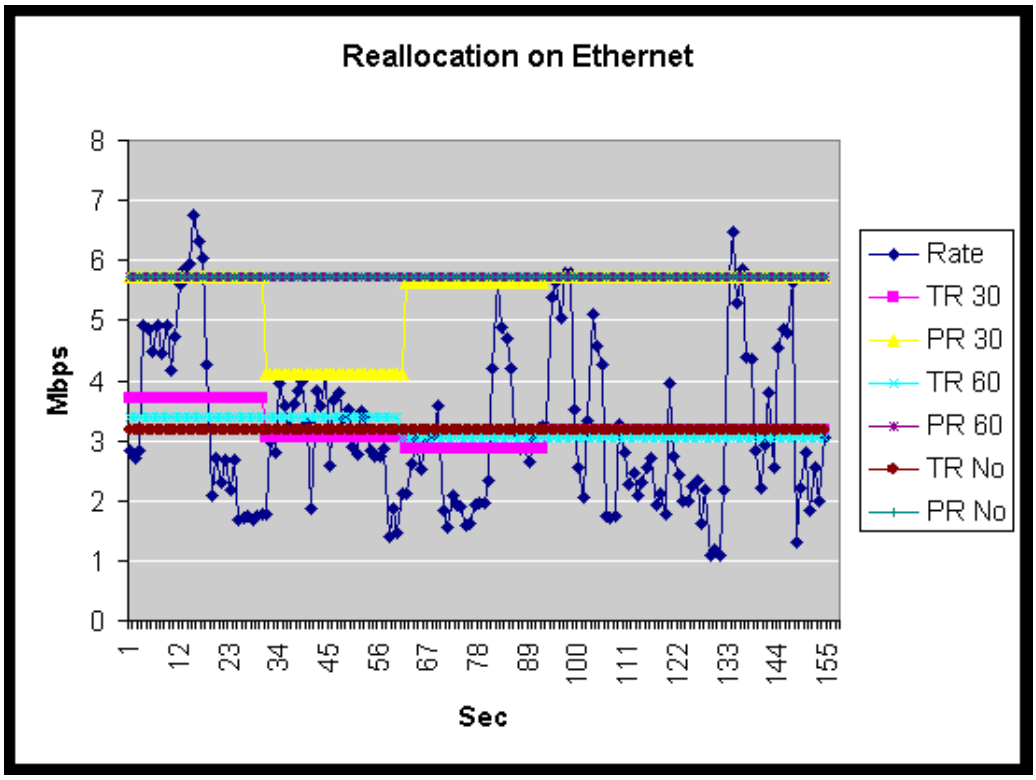

Figure 5: Shared Ethernet. *The BucketRate* r_i is indicated with TR and the *PeakRate* p_i with PR

The graphic in **Figure 6** is related to a Switched Ethernet. In both *short* and *long reallocation* cases, we can observe a gain deriving from reallocation of resources in terms of bandwidth (deriving from *BucketRate* r_i and *PeakRate* p_i parameters of IS) compared to the *legacy* case. An interesting analysis is related to the time needed for reallocation. This is critical because it could affect the right behaviour of the system. A too slow allocation could loose synchronisation between Control Plan (RSVP) and User Plan (Data Pump). The values measured in our trials are lower than 0.1 second, as are shown in **Figure 7**.

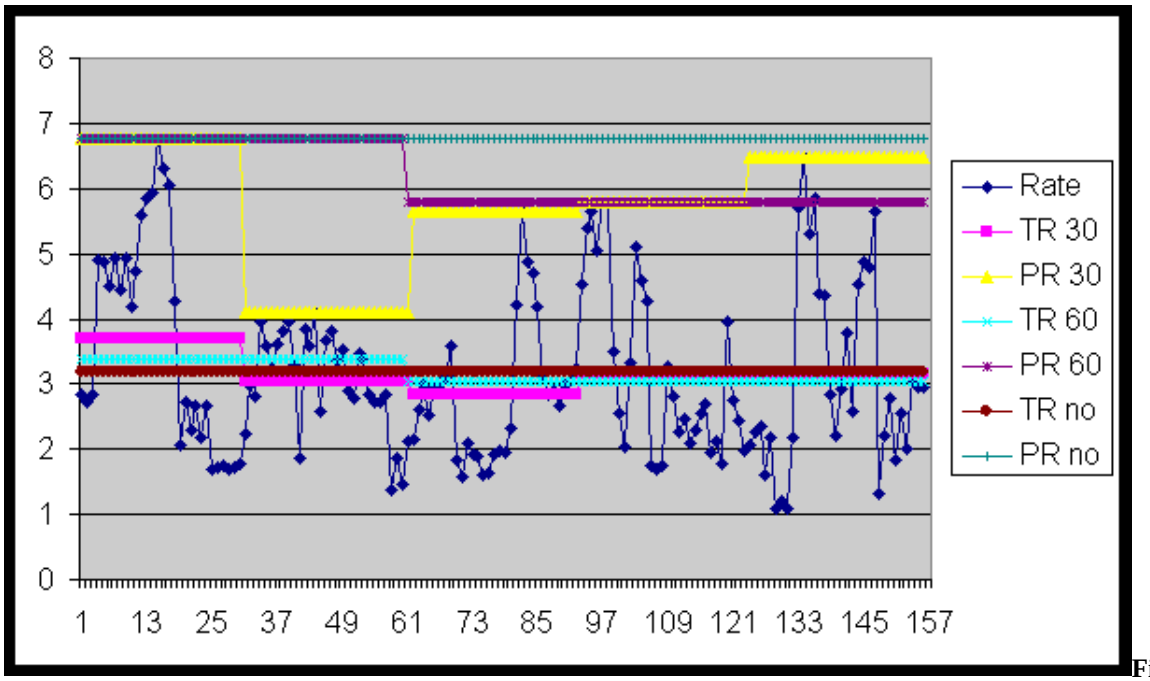

Figure 6: Switched Ethernet. *The BucketRate* r_i is indicated with TR and the *PeakRate* p_i with PR

Figure 7 shows each observed time needed for the reallocation related to all performed experiments. The variation depends on the network load; the average value is 62 msec.

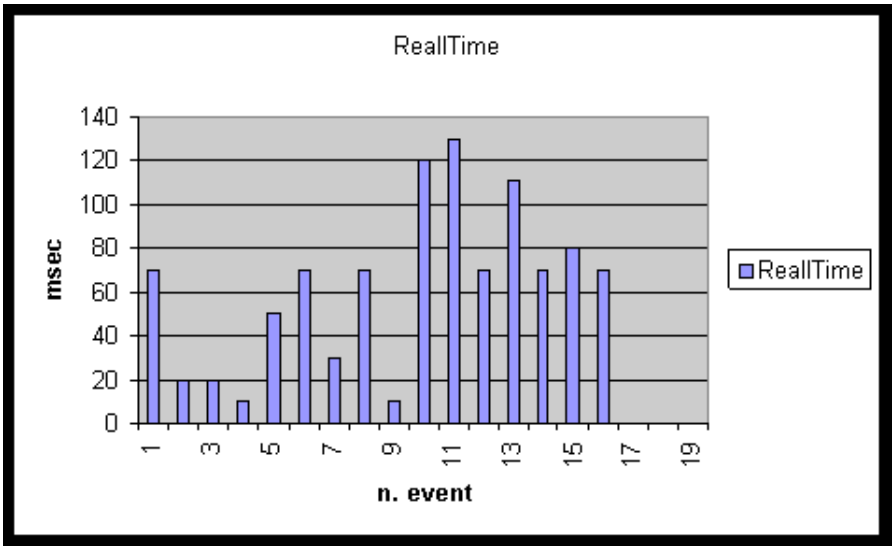

Figure 7: observed time needed for the reallocation

The *BucketDepth* analysis is not significant because it is constrained by the buffer of the NIC.

2. Conclusion

In this work we considered generic multimedia applications with pre-recorded traffic and addressed the problem of supporting the QoS requirements for these applications while efficiently allocating the network resources. We analyzed this problem at the source node where the traffic profile is negotiated with the network and the traffic is shaped according to the

contract.

The goal of this work was to investigate the ability of renegotiation. We presented an architecture design based on the RVBR service, which aims to renegotiate the traffic profile.

We implemented a prototype of a video streaming application that uses RSVP reservation protocol with Controlled-Load service integrated with the RVBR service. To this aim we modified some functions used by the RVBR service in order to use and manage a reduced number of traffic information inside our implementation.

We carried this study on a real network with the prototype client and server exchanging RSVP messages containing a *Tspec* renegotiated according to the RVBR service.

The measurement performed on our testbed with real MPEG2 video traces showed the benefits of applying the renegotiation. In fact, our results indicate that renegotiation is an efficient mechanism to allowing to better utilising network resources at the very low price of implementing a service like RVBR.

Some important aspects that were neglected in this first release will be included in next releases. Among them, but not limited to, we consider

- introduction of timing constraints to select the right QoS: it is a parameter of the traffic descriptor at application layer and it should be introduced during the network optimization as a new constraint
- synchronization between User Plan and Control Plan: problems deriving from delay during reallocation (Control Plan) or on the User Plan (e.g. delay for retransmission or window size in the case of TCP, ...) must be considered
- recovery in case of fault: actions to be performed if a reallocation does not successes

Nevertheless, our results indicate that a reasonable renegotiation, i.e. with renegotiation periods of about 30 seconds (default RSVP), the network resources utilization is better and it works using available technology.

1. References

[1] S. Giordano, J.-Y. Le Boudec: **"On a Class of Time Varying Shapers with Application to the Renegotiable Variable Bit Rate Service"** , SSC Technical Report no. 98/035, 1998

[2] S. Giordano, J.-Y. Le Boudec **"The Renegotiable Variable Bit Rate Service"** SSC Technical Report no. 98/038, 1999

[3] S. Giordano, J.-Y. Le Boudec **"QoS based Integration of IP and ATM: Resource Renegotiation"** , SSC Technical Report no. 98/030, in Proceedings of 13th IEEE Computer Communications Workshop 1998

[4] W. Almesberger, S. Giordano, P. Cremonese, H. Flink, J. Loughney, M. Lorang Lorang **"A Framework for the QoS Based Integration of IP and ATM in the DIANA Project"** , SSC Technical Report no. 98/028, 1998

[5] W. G-2 Specification of IP and ATM Technology Integration to Support Quality of Service in Heterogeneous Networks 1998

[6] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin RFC2205: Resource ReSerVation Protocol (RSVP) - IETF 1997

[7] S. Shenker, C. Partridge, R. Guerin RFC2212: Specification of Guaranteed Quality of Service IETF 1997

[8] J. Wroclawski RFC2210: The Use of RSVP with IETF Integrated Services IETF 1997

[9] J. Wroclawski RFC2211: Specification of Controlled-Load Network Element Service IETF 1997

[10] S. Shenker, J. Wroclawski RFC2216: Network Element Service Specification Template IETF 1997

[11] C. Fogg mpeg2encode/mpeg2decode - MPEG Software Simulation Group 1996

[12] J.-Y. Le Boudec Network Calculus, Deterministic Effective Bandwidth, VBR trunks - IEEE Globecom 97 November