Quality Adaptive end-to-end packet Scheduling to Avoid Playout Interruptions in Internet Video Streaming Systems

Rossella Fortuna, Luigi Alfredo Grieco, *Member, IEEE,* Gennaro Boggia, *Senior Member, IEEE,* and Pietro Camarda

Abstract

In Internet multimedia streaming, the quality of the delivered media can be adapted to the Quality of Service provided by the underlying network, thanks to encoding algorithms. These allow a fine grained enhancement of a low quality base layer at streaming time. The main objective that should be satisfied in such systems is to avoid the starvation of the decoding process and consequent playout interruptions. In this work, we tackle the problem using a control theoretic approach. In particular, we design and implement the novel end-to-end Quality Adaptive Scheduler for properly distributing the network available bandwidth among base and enhancement layers. The developed solution can be adopted in many contexts given that it has been designed without assumptions on the delivered media nor on the protocol stack. Anyway, to test its effectiveness, we have casted it in a H.264/AVC SVC based video streaming architecture for unicast Internet applications. The performance of the scheduler has been experimentally evaluated in both a controlled testbed and several "wild" Internet scenarios, including also UMTS and satellite radio links. Results have clearly demonstrated that our Quality Adaptive Scheduler is able to significantly improve the performance of the video streaming system in all operative conditions.

Index Terms

Bandwidth allocation, videostreaming, layered video, SVC, H.264.

I. Introduction

Multimedia streaming services are nowadays broadly deployed in different scenarios of social life, thanks to technological drivers such as the ever increasing availability of bandwidth, the evolution of advanced quality adaptive

Manuscript received XXX XX, 2009; revised XXXX XX, 2009. This work was supported by the projects PS 121, DIPIS, COST-TMA. A preliminary version of this work appeared as G. Boggia, P. Camarda, R. Fortuna, L. A. Grieco, *A Scheduling Strategy to Avoid Playout Interruptions in Video Streaming Systems,* Proc. of Int. Conf. on Computer Commun. and Networks, ICCCN (2009).

Authors are with the "Dip. di Elettrotecnica ed Elettronica (DEE)" at "Politecnico di Bari". V. Orabona 4, 70125, Bari - Italy

E-mail: {r.fortuna; a.grieco; g.boggia; camarda}@poliba.it

encoding systems, and the wide diffusion of mobile networking devices with high computational and storage capabilities [1]-[5].

Adaptation is the enabling key of such services [6]. In fact, unpredictable packet losses, delays, and bandwidth fluctuations due to the intrinsic behavior of packet switching networks can be compensated by adapting the quality of the delivered media [7]-[9]. In these systems, the media is usually encoded in a base layer stream, representing a low quality version of the content, and one or several enhancement layers that can be progressively added or striped with fine-grained resolution, according to the Quality of Service (QoS) provided by the network [4], [5]. The main objective that should be satisfied in these systems is to avoid the starvation of the decoding process and the consequent playout interruptions [10]. In this work, we tackle this problem using a control theoretic approach. In particular, we design and implement a new algorithm, which will be referred to as Quality Adaptive Scheduler (QAS). It runs at the application layer, properly distributing the network available bandwidth among base and enhancement layers. With QAS, the client feed-backs to the server the amount of buffered base-layer, $Q(t)$, that has not jet been played out. By exploiting these feedbacks, the server regulates the base layer transmission rate, $R_b(t)$, in order to keep $Q(t)$ at a constant target layer Q_0, thus reducing the risk of playout interruptions. The remaining quota of available bandwidth, if present, is used to transmit the enhancement layer. In this way, when the available bandwidth is scarce, QAS is able to lower the risk of playout interruptions at cost of a reduced quality of the delivered media.

For sake of generality, QAS has been designed without making assumptions on the delivered contents nor on the protocol stack. Its effectiveness has been tested casting it in a H.264/AVC SVC (Advanced Video Coding, Scalable Video Coding [4]) based video streaming architecture for unicast Internet applications. This choice is motivated by the fact that the H.264/AVC standard [11] holds a leading position due to its very high compression efficiency. As example, for the same video quality, H264 coding shows an average bit-saving of 30% with respect to the classical MPEG-2 compression standard [12]. The Joint Video Team (JVT) working group is now developing the scalable extension of H.264 [4] (i.e., the H.264/AVC SVC) to add the flexibility of a scalable encoding strategy to the efficiency of H.264. This scalable coding also provides a great flexibility in partitioning the video stream [13]-[15]. In fact, as in the not-scalable H.264 version, the stream is fragmented in elementary Network Abstraction Layer (NAL) units that semantically differs from each other, so that portions of a stream containing different information types can be easily identified and treated, e.g., timing informations, parameter set, enhancement layer data or base layer data [14], [15]. To this aim, the Fine Granularity Scalability (FGS) is used, which allows the server to truncate enhancement layer at any point [4]. Using FGS, the server can finely regulate the enhancement layer transmission rate for a full exploitation of available bandwidth. The client/server architecture, that have been implemented, employes TCP [16] protocol at transport layer. Even if UDP is generally preferred for transmitting multimedia flows [17], this choice is not unusual, as testified by many analogous examples, either in literature [18], [19], [20], or in commercial applications (for instance the well-known *Youtube*, as asserted in [21]). One of the reasons motivating TCP choice is that many networks filtrate UDP flows allowing only TCP connections, as reported in RFC 4571 [22]. Furthermore, implementing QAS on top of TCP represents a challenging objective. In

fact, since TCP uses a window-based congestion control [16], QAS has not an explicit estimation of the available bandwidth, and, as a consequence, NAL scheduing has to take into account cross-layer information coming from the transport layer.

The rest of the paper is organized as follows: in Sec. II, related works are summarized. In Sec. III, the proposed video streaming architecture is described. In Sec. IV, QAS is designed. In Sec. V, experimental results are presented. Finally, the last section draws the conclusions and forecasts future research.

II. Related Works

Research on quality adaptive video streaming systems has been very active in the last years. Herein, we summarize the main contributions that we consider relevant for our discussion, which is mainly focused on unicast video streaming applications in the Internet.

In [10], authors present a theoretical discussion on the optimal repartition of the bandwidth among base and enhancement layers. The results of the study can be only applied to scenarios having a negligible Round Trip Time (RTT). Furthermore, no validation in realistic settings has been reported.

Many works on quality adaptive video streaming, such as [7] and [23], consider the problem of minimizing quality variations, in order to maximize perceived video quality. They only focus on enhancement layers delivery, under the assumption that base layer rate is smaller than the available bandwidth. This assumption could be unsafe because Internet bandwidth is variable and can even snap to zero during transients. Moreover, assuming that base layer rate is smaller than the available bandwidth, implicitly drives to encode the video using a very small base rate and a very high enhancement rate. The subtle effect of this choice consists in a reduction of the compression efficiency, which increases proportionally to the ratio between base rate and enhancement rate [24].

With our approach, assumptions on the available bandwidth can be relaxed: QAS is able to avoid playout interruptions if the average available bandwidth is greater than the base rate, thus compensating transient bandwidth reductions.

In [8], an algorithm for the joint design of encoding rate control and congestion control has been proposed. At transport layer, a modification of the TCP Friendly Rate Control (TFRC) algorithm [25] has been designed, ensuring TCP friendliness in the long term. Constraints on both encoding rate and sending rate of the transport layer are derived using a Virtual Buffer approach [26]. The performance of the algorithm has been tested using ns-2 simulations in a four hop topology with short and long lived TCP connections as well as multimedia traffic sources.

A real-time TCP-based video streaming system is proposed in [18]. Due to the real-time nature of the system, a discarding frame algorithm provides the sending of only frames that would arrive in-time at the client, according to an estimated end-to-end delay. Each frame is associated with a transmission deadline, after which it is discarded by the source. To assign a higher priority to base frames, the deadlines of enhancement frames are lowered when the available bandwidth is scarce. This mechanism is particularly suited if temporal scalability is used. It is not useful with FGS, because a mechanism to dynamically truncate FGS frames is not provided. As a consequence, it

can not exploit FGS capability to totally adapt enhancement rate to available bandwidth. Another limit of the study presented in [18], is that system performances have been evaluated only by ns-2 simulations.

To conclude, the main lacks of the aforementioned works are: (i) restrictive or simplistic hypothesis on the underlying network and/or on the allowed codecs; (ii) the absence of a validation through real experiments on Internet.

In the present work, our objective is to design a robust and general mechanism for avoiding base layer buffer underflows, which can heavily degrade the quality of the media due to abrupt playout interruptions. QAS algorithm has been implemented and tested in many networking conditions to demonstrate its general validity. We believe that this proposal can be though as an important building block of a generic streaming architecture. In fact, it has been designed without making assumption on the delivered media or on the protocol stack. Thus, it could be integrated in the future to enforce recent proposals available in literature, such as those previously reported in this section.

III. Streaming Architecture

The H.264/AVC SVC based video streaming system implemented is based on RTP/RTCP protocols (Real Time Protocol/Real Time Control Protocol) [27]. TCP has been used at the transport layer. Working in an Unix Kubuntu-based environment, the default Kubuntu TCP implementation has been used, i.e., CUBIC TCP [28]. Live555 libraries [29] and the JVT Joint Scalable Video Model (JSVM) module [30] have been properly modified to allow the implementation of QAS.

The client is made of three modules: the *network entity*, the *decoder*, and the *viewer* (see Fig. 1). The *network entity* receives RTP packets, reorders NALs in decoding order, and stores them in a decoding buffer. To enable the adaptation of the base layer transmission rate ($R(b)$), every Round Trip Time (RTT), using RTCP-APPlication-defined (RTCP-APP) packets [27], it also sends a feedback message to the server containing the actual base layer buffer level $Q(t)$. If $RTT > 200ms$, the feedback is sent just after 200 ms in order to enforce system responsiveness. The *decoder* decodes NALs in its buffer to obtain video frames. The *viewer* displays video frames at the sequence frame rate.

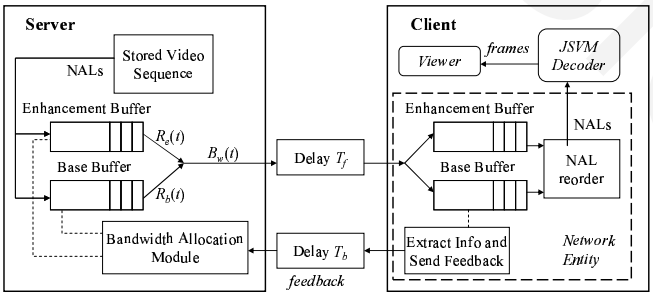


Fig. 1. QoS control Implementation in Client/Server Architecture.

The server, upon receiving a client request, recognizes the file indicated by the client, extracts NALs from the storage file of the requested video-sequence, partitions NALs in enhancement and base ones, enqueues them into

two different transmission buffers, and starts the streaming using the QAS scheduling strategy, that will be described in the next section.

Each H.264 NAL can be fragmented and each fragment is encapsulated in one RTP packet, using a modified version of FU-B (Fragmentation Unit Packet, type B) packetization mode of the RTP protocol [27]. Fragments of a single NAL have to be sent with increasing RTP sequence numbers [15]. The structure of the resulting RTP payload is shown in Fig. 2. With reference to this figure, the *FU-B indicator* informs the client that FU-B encapsulation mode is used; such an encapsulation allows interleaved packetization mode (i.e., it is possible to send NALs out of decoding order). The *Decoding Order Number* field is useful in the Interleaved mode given that it represents the decoding order of the NAL which the encapsulated fragment belongs to. Finally, the *FU header* contains information about the current fragment (if it starts or ends the NAL), and also it indicates the type of the NAL, making possible to distinguish among base and enhancement NALs.

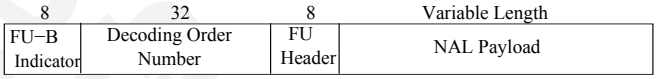


Fig. 2. RTP Payload Structure.

IV. QAS Algorithm

The main role of the QAS strategy is to properly distribute, among base and enhancement layers, the bandwidth available along a given client-server connection. The goal is to avoid or strongly limit playout interruptions at the decoder side. In detail, the receiver periodically feedbacks its base layer buffer level, $Q(t)$, to the sender, which, exploiting a closed-loop control algorithm, throttles the transmission rate, $R_b(t)$, of the base layer. Making this, it keeps the base layer buffer at the desired target level Q_0. If the so computed $R_b(t)$ exceeds the available bandwidth, $B_w(t)$, it is set equal to $B_w(t)$. The bandwidth left available by the base layer, if present, is used for transmitting the enhancement layer at rate $R_e(t)$, that is:

$$R_e(t) = [B_w(t) - R_b(t)]^+ \ , \tag{1}$$

where $[x]^+ = \max(0, x)$.

A. The Basic Control Law

The algorithm used by the server to evaluate $R_b(t)$ has been developed with a control theoretic approach. In particular, the interaction between client and server can be represented using a closed-loop system (see Fig. 3) [31]. The control approach is particularly useful to design the equation for the computation of the base layer sending rate, because it allows to control the level of the remote base decoding buffer, even in the presence of not-negligible network-inserted delays. In such a scheme, we jointly exploit a proportional controller and a Smith predictor to compensate both disturbances and delays introduced by the network. Using a constant gain we can

exploit control theory arguments related to Linear Time-Invariant systems. The base layer buffer has been modeled using an integrator. The rate at which the JSVM decoder extracts base NALs from such a buffer is the signal $d(t)$. It is worth to note that the server cannot predict $d(t)$ in advance, because it does not know exactly neither the playout delay after which the client begins the extraction of base NAL, nor, eventually, the presence of playout interruptions that force to zero the extraction rate $d(t)$. Thus, from a control theoretic perspective, $d(t)$ can be modeled as a disturbance [32]. Note that the base layer buffer is drained at rate $d(t)$ while it is filled at rate $R_b(t - Tf)$, which is the base layer transmission rate, delayed by the forward path from the sender to the receiver. The integrator $1/s$ translates the impact of the receiving base rate $R_b(t - Tf)$ and of the reading rate $d(t)$ into the queue level $Q(t)$. The receiver periodically feedbacks the $Q(t)$ signal to the sender, using the backward path, which inserts a delay T_b.

We take into account delays in the control loop because they are not negligible and can be critical for system stability. These delays on forward (T_f) and backward (T_b) paths are modeled by the functions $e^{-T_f s}$ and $e^{-T_b s}$, respectively. As a further consideration, even if T_f and T_b are considered separately in Fig. 3, the controller just requires to know their sum, i.e., $RTT = T_f + T_b$, which is easier to estimate in real settings.

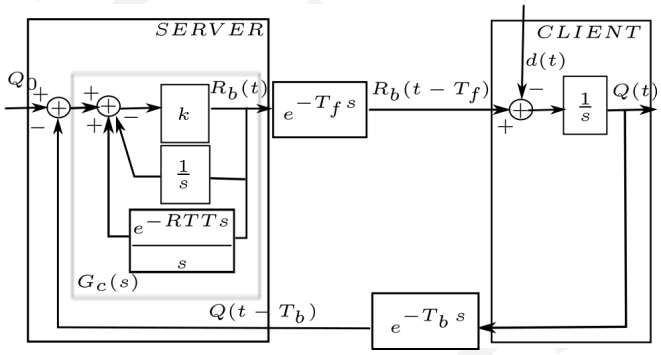


Fig. 3. Closed-loop control based on the Smith predictor.

To compute $R_b(t)$, the server compares $Q(t - T_b)$ against the constant reference signal Q_0. To compensate delays in the control loop, the Smith predictor has been used for designing the regulator $G_c(s)$ [32]. It is a well known technique exploited in time-delay systems, such as the control loop we are considering. It tries to make the compensation by a local reproduction of the plant (i.e., the client side). Its rationale is to predict in advance the behavior of the plant in order to counteract the impairments due to delays inside the control loop. In this way, the system model, in absence of the disturbance $d(t)$, is equivalent to the one shown in Fig.4, free from delays, in accordance with Smith principle.

From the scheme in Fig. 3, we can evaluate the controller transfer function as follows:

$$G_c(s) = \frac{R_b(s)}{Q_0 - Q(s)e^{-T_b s}} = \frac{k}{1 + (k/s)(1 - e^{-RTTs})}, \tag{2}$$

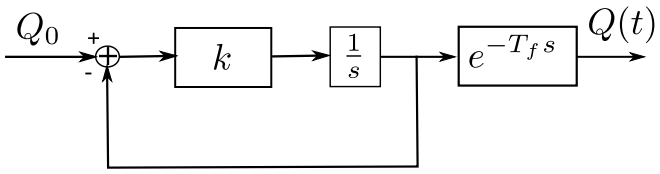


Fig. 4. Equivalent block diagram in absence of the disturbance $d(t)$

which can be translated in the time domain as follows:

$$R_b(t) = k\left[Q_0 - Q(t - T_b) - \int_{t-\text{RTT}}^{t} R_b(\tau)d\tau\right]. \tag{3}$$

The control law (3) has been designed assuming a continuous-time model for the streaming system, but it has to be discretized in order to be useful for our purposes. In particular, a new value for $R_b(t)$ should be computed each time a feedback is received by the sender. Thus, we can say that the control law (3) should be sampled with a period equal to $T_c = \min(200\text{ms}, RTT)$. It is well known (see [33] and [31]) that the discretization of a continuous-time control law can be successfully done if the sampling period T_c is smaller that the slowest time-constant of the closed-loop system. In this case, if we analyze the transfer function between the output variable $Q(t)$ and reference signal Q_0 in absence of disturbance (see Fig.4), we obtain the following result:

$$\frac{Q(s)}{Q_0(s)} = \frac{k}{s+k}e^{-sT_f} \tag{4}$$

which indicates that the time-constant of the system is $\frac{1}{k}$.

Therefore, we have to impose $\frac{1}{k} > T_c$. To be safe, we have set $k = \frac{1}{4T_c}$.

Note that the control loop acts at application layer, regulating the transmission of the base layer at application layer too. Also RTT has to be intended at application layer. RTT used in the control law, in fact, is computed exploiting RTP packets timestamps and RTCP feedbacks. In RTCP feedbacks, the client inserts, together with the instantaneous level of base decoding buffer $Q(t)$, the following informations used by the sender to compute RTT:

- $LastTimestamp$: the timestamp of the last received RTP packet.
- $IntervalTime$: the time interval between the reception of the last received packet and the transmission of the feedback.

The sender will compute RTT as:

$$RTT = NowTimestamp - LastTimestamp - IntervalTime \tag{5}$$

where $NowTimestamp$ is the current timestamp on the sender. Note that clock synchronization of client and sender is not necessary for RTT computation because $NowTimestamp$ and $LastTimestamp$ have been originally issued by the sender while $IntervalTime$ is a difference among time instants. The computed RTT is stamped in all RTP packets sent by the server, using RTP header extension.

Another consideration has to be done about RTT. In the model, we considered RTT as a constant. This is not a realistic hypothesis due to the nature of the Internet, but this simplification has been often done in literature

(see [34], [35], [36]) in order to obtain a mathematically tractable formulation of the control law. Obviously, the validity of the approach designed under this restrictive assumption should be tested with real experiments. This is the reason why, as shown in section V, we have extensively measured the performance of the proposed approach in many real heterogeneous Internet scenarios, also including UMTS and satellite links.

B. The impact of the disturbance

In this subsection, we will analyze the impact of the disturbance $d(t)$ when the control law (3) is used. We can model the disturbance as a sum of a constant value plus a time-varying component, as follows:

$$d(t) = [d_0 + \Delta d(t)] \cdot 1(t). \tag{6}$$

where d_0 is the average value of the base decoding rate in ideal conditions, $\Delta d(t)$ is a a time-varying signal, and $1(t)$ is the unitary step function. The server knows exactly the average ideal base NAL decoding rate d_0, because it is exactly equal to the source base rate, fixed during the encoding process. On the opposite, the signal $\Delta d(t)$ cannot be predicted by the server because it models, besides the source base rate variability, even possible, unpredictable delays in extracting base NALs, due to unpredictable network-dependent playout interruptions.

Given the linearity of the system in Fig. 3, for the superposition principle we can analyze the impact of both components of $d(t)$ singularly. The signal $\Delta d(t)$ will produce an output $q_\Delta(t)$, that we cannot be fully compensated as $\Delta d(t)$ is unknown. On the contrary, as shown below, the d_0 component will produce an undesired effect proportional to its value, that cannot be undertaken, especially in case of high definition sequences. To derive the impact of the constant component of $d(t)$, we will exploit the expression of the Laplace Transform, $Q_d(s)$, of $Q(t)$ obtained for $Q_0 = 0$:

$$Q_d(s) = D(s)G_d(s) = -D(s)\frac{1/s}{1 + (Gc(s)/s)e^{-RTT \cdot s}} \ . \tag{7}$$

As we are neglecting $\Delta d(t)$, we can derive the steady-state value of $Q(t)$ obtained for $d(t) = d_0 \cdot 1(t)$ using the final-value theorem [31], as follows:

$$q_D(\infty) = \lim_{t\to\infty} Q(t) = \lim_{s\to 0} sQ_d(s) = \lim_{s\to 0} s\frac{d}{s}G_d(s) = -d_0(1/k + RTT). \tag{8}$$

C. An Enhanced Control Law

The continuous component of the disturbance introduces at steady state negative offset that increases with d_0 and RTT (see Eq. (8)). As a consequence, with high values of RTT and d_0, the level of base buffer in steady state can dangerously decrease and oscillations due to bandwidth fluctuations could cause underflows. By exploiting the results of the previous sub-section, in order to reject the disturbance, we propose to enhance the control law (3), increasing Q_0 by $d_0(1/k + RTT)$, thus, obtaining:

$$R_b(t) = k\left[Q_0 + d_0(1/k + RTT) - Q(t - T_b) - \int_{t-\mathrm{RTT}}^{t} R_b(\tau)d\tau\right]. \tag{9}$$

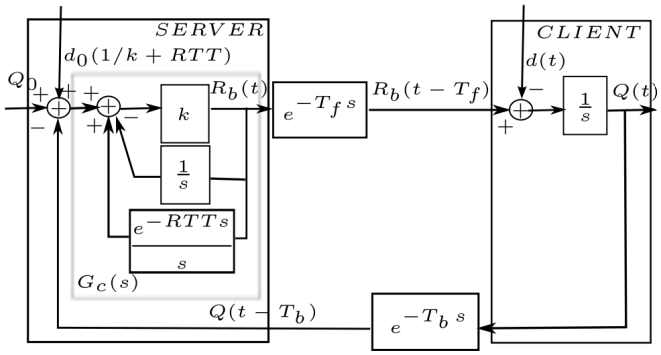


Fig. 5. Enhanced closed-loop control with disturbance rejection

The resulting block diagram is depicted in Fig. 5.

Using the control law (9), a perfect cancellation of the constant disturbance component $d_0 \cdot 1(t)$ at steady state can be fulfilled. In fact, if we define as $G_{d2}(s)$ the new transfer function between the output variable and the disturbance, obtained from the diagram in Fig.5 and setting $Q_0 = 0$, the resulting $q_D(\infty)$ becomes:

$$q_D(\infty) = \lim_{s\to 0} s\frac{d}{s}G_{d2}(s) = \\ = \lim_{s\to 0} s\frac{d_0}{s}\frac{(1/k + RTT)e^{-T_f s}G_c(s) - 1}{s + G_c(s)e^{-RTTs}} = 0.$$

Note that, to apply the control law (9), the sender has to be aware of the average amplitude of the disturbance, d_0.

D. The transmission of NALs

In this subsection, we explain how to schedule the transmission of NALs according to the values of $R_b(t)$ and $R_e(t)$, computed by eqs. (9) and (1), respectively. Assuming that the server sends a base NAL, namely A, of size s_A at the time instant t_A, the next base NAL, namely B, is scheduled at the time instant $t_B = t_A + s_A/R_b(t_A)$, to take into account the actual value of $R_b(t)$ (see Fig. 6).

Livemedia [29] allows delaying NAL transmissions with a microsecond resolution, but timer accuracy depends on the interrupt frequency of used system hosting the server [37]. In our implementation, a 1000 Hz interrupt frequency system has been used. Note that NAL transmissions are intended at application layer, i.e., transmitting a NAL is equivalent to write it on the TCP transmission buffer.

The time interval (t_A, t_B) is used to send enhancement layer, starting from $t_1 = t_A + \varepsilon(t_A)$, where $\varepsilon(t_A)$ is the system delay for writing A in TCP transmission buffer and begin the transmission of the next enhancement NAL, namely D. For simplicity, we neglect the system delay considering $t_1 = t_A$. Just after A transmission, the number of enhancement layer bytes that can be sent is computed as [16]:

$$AllowedBytes(t_A) = \min(awnd(t_A), cwnd(t_A)) - UnackedBytes(t_A) - s_A, \quad (10)$$

where

- $AllowedBytes(t_A)$ is the number of bytes that can be instantaneously sent at time t_A by TCP;
- $cwnd(t_A)$ is the current value in bytes of the congestion window, at the server TCP Linux socket [38];
- $awnd(t_A)$ is the value in bytes of the last advertised window by the client;
- $UnackedBytes$ is the number of bytes still unacked; also this info can be retrieved from Linux TCP socket.
- s_A is the size of the just sent base NAL, A.

After t_A, the server will check regularly the amount of data that can be transmitted, with a time step of $20ms$. At a given check instant $t_k \in (t_A, t_B)$, the number of enhancement layer bytes that can be sent is computed as:

$$AllowedBytes(t_k) = \min(awnd(t_k), cwnd(t_k)) - UnackedBytes(t_k). \tag{11}$$

At each instant t_k, the server can send an Enhancement NAL fragment of size $AllowedBytes(t_k)$, and try to send next fragment at next checking instant. If the fragment ends the NAL without exploiting all $AllowedBytes(t_k)$, straight after the server will send the next Enhancement NAL fragment in another RTP packet as provided by FU packetization mode. In this way, the rate $R_e(t)$ is implicitly estimated from the current congestion window size and from the number of in flight bytes on the half-TCP connection from the server to the client. An explicit estimation of $R_e(t)$ would require to know the instantaneous transmission rate of TCP, which cannot be easily computed given that TCP congestion control is window-based. At every instant t_k, the sender also reschedules the next base NAL B by re-computing t_B using the updated $R_b(t)$ value. If the newly computed t_B precedes the instant t_k, then the base NAL B is instantaneously sent.

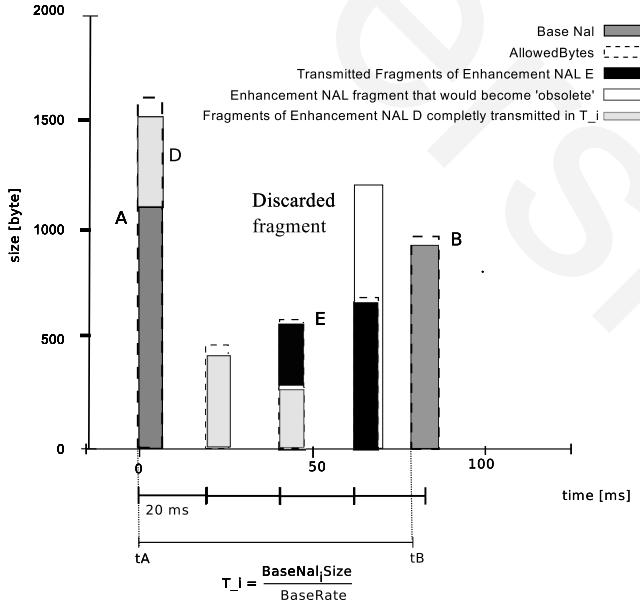


Fig. 6. NAL schedule.

When a base NAL B is sent, the server automatically discards all Enhancement NALs or fragments that precede B in decoding order, and still have not been sent. In this way, the algorithm avoids to waste available bandwidth sending "obsolete" enhancement data, that could be discarded on receiver side. The aforementioned steps are summarized in Algorithm 1

Algorithm 1 QAS steps in $[t_A, t_B)$

Send the NAL A
Compute $t_B = t_A + s_B/R_b(t_A)$
if $t_B > t_A$ **then**
    Transmit a number of enhancement data equals to $Allowed\_Byte(t_A)s$ according to eq. (10)
end if
$t_k = t_A + 20$ [ms]
while $t_k < t_B$ **do**
    Recompute $t_B = t_A + s_B/R_b(t_A)$
    if $t_B \leq t_k$ **then**
        Send the NAL B
    else
        Schedule NAL B at the new computed t_B
        Discard obsolete enhancement NALs preceding A in decoding order
        Transmit a portion of enhancement equals to $Allowed\_Bytes(t_k)$ according to eq. (11)
        $t_k = t_k + 20$ [ms]
    end if
end while

We have to remark that the implemented scheduling strategy presents some approximations. In fact, the values of TCP parameters, used for computing $AllowedBytes$, are retrieved by a *Web100* patched Linux kernel [39] installed on the server host. These values are affected by an error due to the communication delay between the kernel and the user space. This approximation becomes an error in the bandwidth estimation that could lead QAS to send too few or too much enhancement layers. Such an error can be modeled as a further disturbance in the control loop and, as a consequence, its impact is mitigated by the control law.

V. Experimental results

The performance of QAS has been tested in different scenarios, in order to demonstrate its general validity. In particular, we have considered: (i) a wired network in a controlled environment; (ii) an international Internet path; (iii) Internet connections involving either UMTS or satellite links. We have compared QAS with respect to a simple delivery approach, which will be referred to as "No QAS" scheduling, which sends video at full rate without any prioritization of the base layer.

For the tests, the following settings have been used:

- video test sequence is a modified version of "Canoe" test sequence available at [40]. The original sequence at 30 Hz frame rate with CIF resolution (352x288) and 4:2:0 YUV format has been temporal subsampled at a frame rate of 7.5 Hz. The resulting sequence has been periodically repeated 65 times to obtain a 3575 frames long sequence. Only results for "Canoe" will be shown, but analogous results have been found for other sequences available on line at [40].
- The Group Of Picture (GOP) size parameter in the JSVM codec has been set equal to 4.
- The enhancement layer is made up of a single FGS layer.
- The Quantization Parameter is set equal to 45 for the base layer and 39 for the FGS enhancement layer. With these settings, base video rate is always equals to enhancement video rate. In particular, for "Canoe", full quality video rate is 340 kbps, and base video rate is 170 kbps.
- According to the chosen encoding parameters, a single frame at base layer is made by a shorter than 11 bytes PREFIX NAL (which is a NAL indicating decoding dependencies for enhancement layer prediction and scalability hierarchy [41]) and one variable length base NAL.
- RTP/RTCP over TCP encapsulation is used.
- RTCP feedback messages are sent on regular basis every $T_c = \min(RTT, 200 \text{ ms})$.
- The initial playout delay is equal to 1 s.

During each experiment, the following data have been collected at the client side:

- the sizes of the enhancement and base layer buffers;
- the Peak Signal to Noise Ratio (PSNR) of the decoded video, computed for luma (Y) and chroma (C_b and C_r) components by using the JVT PSNR tool;
- statistics of playout interruptions.

To measure the impact of playout interruptions, the time interval between the visualization of consecutive frames has been considered. In fact, in an ideal case (i.e., no playout interruptions) one would expect a constant inter-frame time. As a consequence, the deviation of the inter-frame time from the expected reference value is a measure of the severity of the playout interruptions. Since we work with video sequence of temporal resolution equals to 7.5 fps, inter-frame time should be close to 133 ms. When buffer underflows occur, the visualization of frames is delayed. In particular, we collect the following measurements:

- Total Delay accumulated at the end of the experiment in the sequence visualization: it is zero if no interruptions occur.
- Number of delayed frames, due to buffer underflows: it is zero in ideal conditions.
- Mean/Maximum/95 percentile playout inter-frame time: in the ideal case all these performance indexes should be equal to 133 ms.

A. Controlled Environment

QAS has been tested in the wired controlled environment pictured in Fig. 7, which models a single bottleneck network, shared by one video stream and concurrent UDP flows. Both end hosts are connected to a Linux router using a 1 Gbps link. On the router interfaces, bandwidth limitations and propagation delays have been applied, using traffic control disciplines available in the Linux operating system.

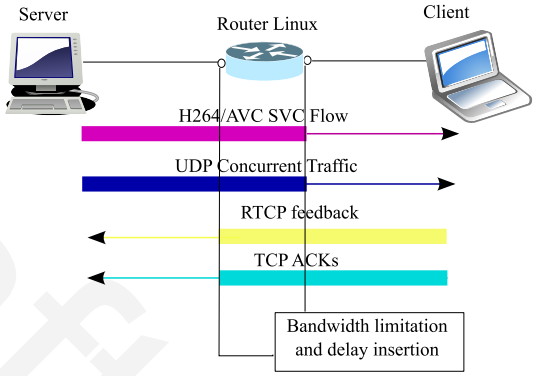


Fig. 7. Testbed design and bandwidth limitation implementation.

The capacity of the links have been set equal to 1 Mbps, whereas propagation delays are set to 50 ms. The bottleneck buffer size is set equal to the bandwidth delay product, i.e., 12.5 kbyte. Each experiment lasts 250 s. To obtain time-varying available bandwidth and queuing delays, an ON-OFF concurrent UDP flow (generated using Iperf [42]) has been inserted. During the ON phase, the UDP flow injects packets at a constant rate of 800 kbps. During the OFF phase it is silent. We have considered three different starting times for the UDP flow, namely, 0 s, 15 s, and 50 s. The stopping time is 200 s. Using these very simple settings, we will show that a small perturbation on the starting time of the UDP flow can dramatically impair the performance of the system when QAS is not used. On the opposite, QAS produces the expected performance in all considered conditions.

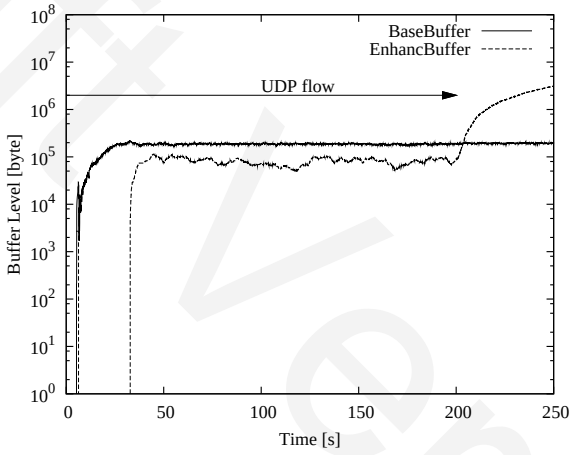
Experimental results, synthetically reported in Tab. I, show the robustness of QAS, which performs in the same way for different network conditions. On the contrary, when QAS is not used, performance can radically change with the experiment settings. From Tab. I is also evident that QAS fulfills its target of avoiding base layer buffer underflows, at the expenses of the video quality. In fact, it has a PSNR slightly smaller than in “No QAS” scenarios.

To provide further insights, Figs. 8 - 10 report the levels of base and enhancement layers stored at the client. From these figures, it turns out that when QAS is not used, the system can produce a lot of playout interruptions. The performance gain due to QAS is relevant (see Figs. 8 and 9) when the UDP flow is turned on at $t = 0$ s or $t = 15$ s. In the other case, i.e., $t = 50$ s (see Fig. 10), QAS does not improve the system performance. The reason is that more later is activated the UDP flow and more data are stored by the client in the buffer; this helps in avoiding underflows (also when QAS is not used).

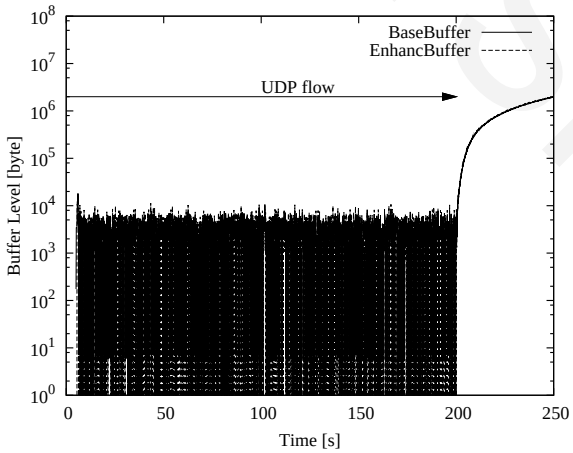
TABLE I

QAS AND "NO QAS" COMPARISON FOR THE THREE EXPERIMENTS IN A CONTROLLED SCENARIO.

		Start UDP at t = 0 s		Start UDP at t = 15 s		Start UDP at t = 50 s	
		QAS	"No QAS"	QAS	"No QAS"	QAS	"No QAS"
Interruptions	Total Delay [s]	0	37.4	0	18.5	0	0
	Num Interr	0	154	0	283	0	0
	Mean [ms]	133	153	133	144	133	133
	Maximum [ms]	133	634	133	660	133	133
	95percentile [ms]	133	303	133	227	133	133
PSNR	Mean [dB]	27.1	29	27.6	29	28.8	29

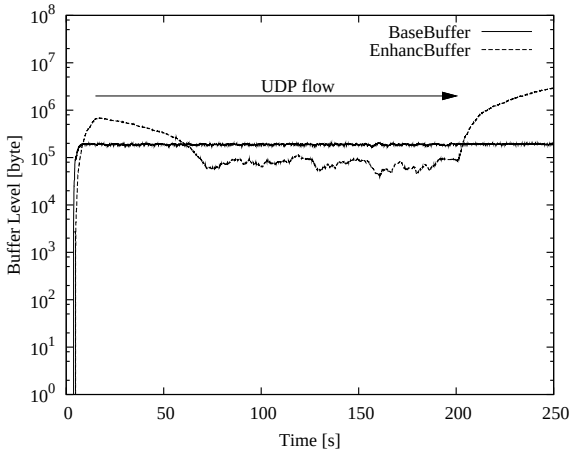


(a) QAS

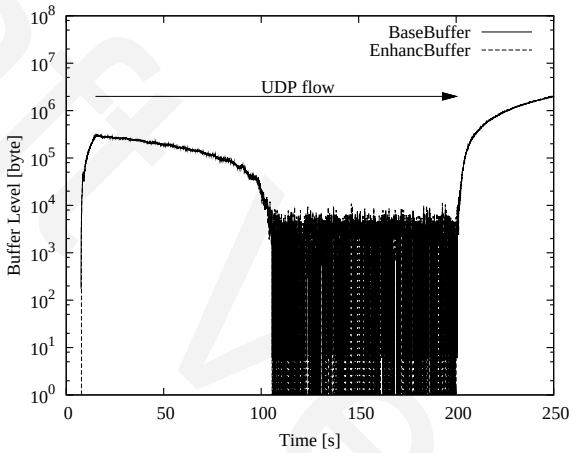


(b) "No QAS"

Fig. 8. Buffer levels in controlled scenario (UDP concurrent flow at t = 0 s).



(a) QAS



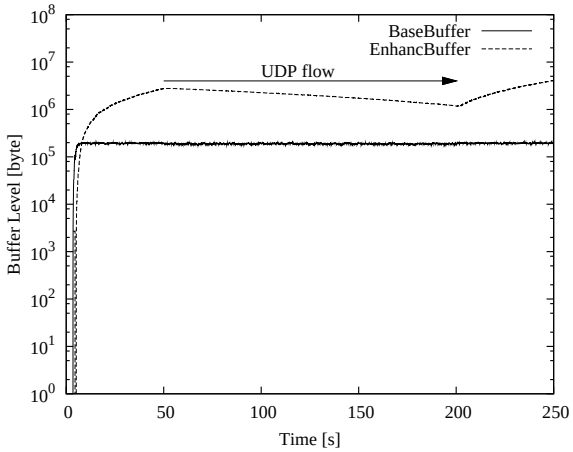
(b) “No QAS”

Fig. 9. Buffer levels in controlled scenario (UDP concurrent flow at t = 15 s).

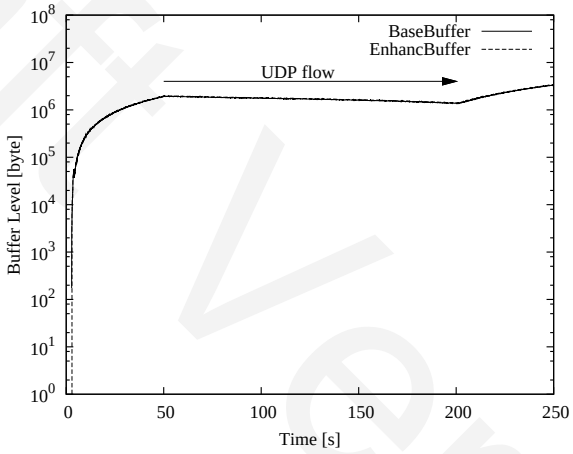
B. Measurements in International Internet Paths

Next experiments are focused on studying the algorithms behavior in real Internet scenarios. The system has been tested over a wide area network on an international path connecting the client, placed at Politecnico di Bari (Italy), to the server, placed in Vienna (Austria). To compare the algorithms, we choose to stream two flows contemporaneously, using for one flow QAS scheduling, and for the other one, “No QAS” algorithm.Due to the high bandwidth variability of the Internet connection, we repeated each experiment 5 times in order to better compare the two algorithms. In Tab. II, we report the average as well as the range of the observed values, obtained over the five realizations of each experiment.

As expected from results obtained in controlled conditions, QAS algorithm is able to avoid playout interruptions in all the conducted experiments. A PSNR reduction of 2 dB is registered in case of QAS usage, whereas “No QAS” presents several interruptions that cause an average total delay equal to 9 s. To better understand algorithm



(a) QAS



(b) "No QAS"

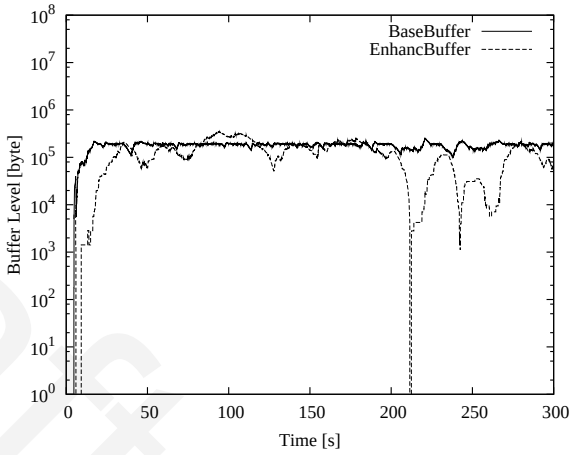
Fig. 10. Buffer levels in controlled scenario (UDP concurrent flow at t = 50 s).

TABLE II

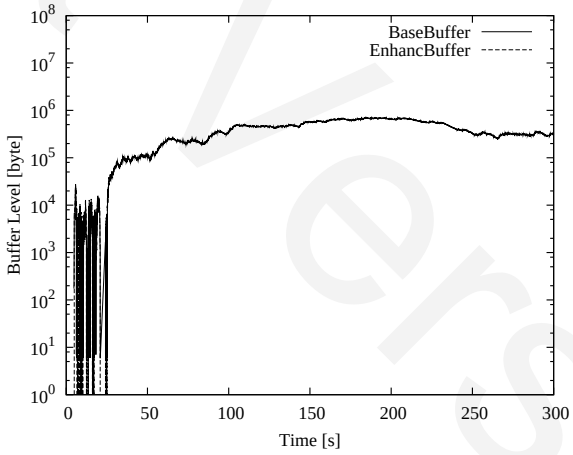
QAS AND "NO QAS" COMPARISON OVER AN INTERNATIONAL INTERNET PATH.

		QAS	"No QAS"
Interruptions	Total Delay [s]	0 [0; 0]	9 [0; 31]
	Num Interr	0 [0; 0]	19 [0; 58]
	Mean [ms]	133 [133; 133]	137 [133; 149]
	Maximum [ms]	133 [133; 133]	1900 [133; 5300]
	95percentile [ms]	133 [133; 133]	133 [133; 133]
PSNR	Mean [dB]	27 [23; 28.7]	29 [29; 29]

behavior, in Figs. 11(a) and 11(b), the buffer levels obtained in one of the 5 realizations have been reported. When QAS is used, the base layer buffer is kept very close to the target value Q_0, while the enhancement one oscillates according to the available bandwidth. On the contrary, when QAS is not used, both buffers are treated in the same way with several playout interruptions.



(a) QAS



(b) "No QAS"

Fig. 11. Buffer levels over an International Internet Path.

C. Measurements using an UMTS link

Several experiments have been done on a heterogeneous Internet path, including an UMTS radio link. The scenario is similar to the one described in the previous section. The only difference is that the last hop for the server in Vienna is a wireless UMTS link (see Fig.12). This scenario is particularly useful to test the robustness of the proposed algorithm in the presence of a wireless link with a narrow bandwidth.

In this environment, video flows controlled by QAS experience a number of interruptions two order of magnitude smaller with respect to flows not controlled by QAS (see Tab. III). The very huge performance gain due to QAS

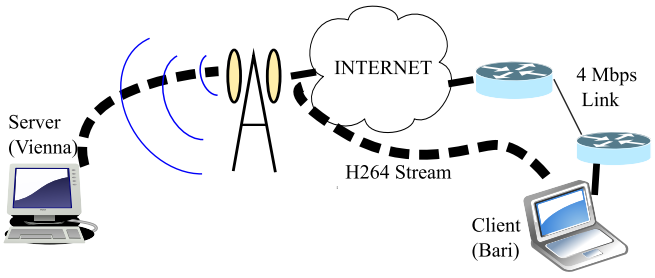


Fig. 12. UMTS testbed.

TABLE III
QAS AND "NO QAS" COMPARISON OVER UMTS CONNECTIONS.

		QAS	"No QAS"
Interruptions	Total Delay [s]	0.3 [0; 1.5]	107 [100; 123]
	Num Interr	1.6 [0; 8]	580 [557; 600]
	Mean [ms]	133 [133; 133]	219 [209; 283]
	Maximum [ms]	233 [133; 636]	1674 [1214; 3385]
	95percentile [ms]	133 [133; 133]	507 [473; 584]
PSNR	Mean [dB]	26.4 [26.3; 26.5]	29 [29; 29]

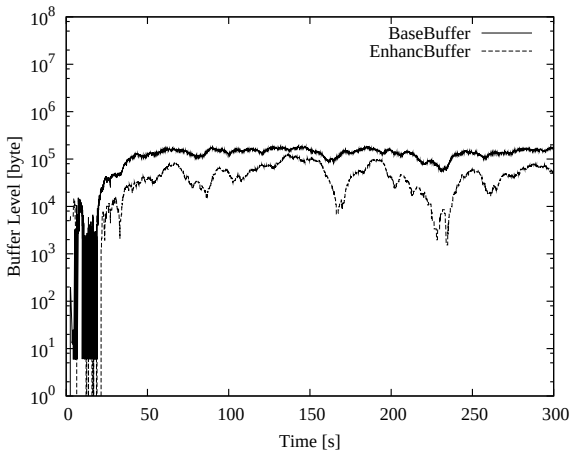
in the presence of a UMTS link clearly highlights the real utility of this algorithm for ubiquitous multimedia applications. As expected in this case, the limitation of the number of interruptions is achieved at the expense of PSNR.

In Fig. 13(a), we plot the buffer levels in the worst experiment, in order to show QAS behavior also in presence of playout interruptions. In particular, in this case interruptions are due to a bandwidth reduction occurred before the target buffer level Q_0 is reached. However, the comparison with "No QAS" behavior, in the same conditions, reveals evident advantages in using QAS (see Fig. 13(b)).

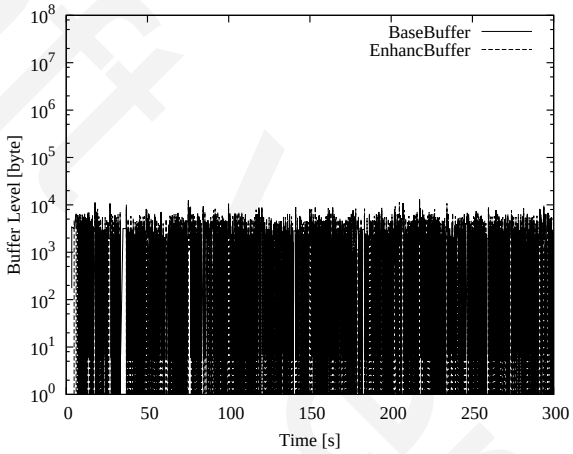
D. Measurements using a Satellite link

The last scenario considers a connection using also a satellite link, as shown in Fig. 14.

The satellite connection is characterized by a large delay and a limited bandwidth, strongly dependent on weather conditions. So that, this scenario is particularly appropriate to analyze QAS performances under strenuous conditions. Due to the great variability of available bandwidth, reported results are characterized by a large variance. In particular, in 3 cases, QAS offers an optimal quality of visualization, avoiding interruptions and offering a PSNR close to the maximum. On the contrary, in the worst case QAS presents a large number of interruptions, reaching a total delay in sequence visualization of 18 seconds. In each experiment, however, QAS algorithm performs strongly better than "No QAS", which presents a very high number of interruptions. Tab. IV summarizes the obtained results. Furthermore, also in this case we report the buffer levels for one of the five realizations to demonstrate the



(a) QAS



(b) "No QAS"

Fig. 13. Buffer levels over Internet paths including a UMTS link.

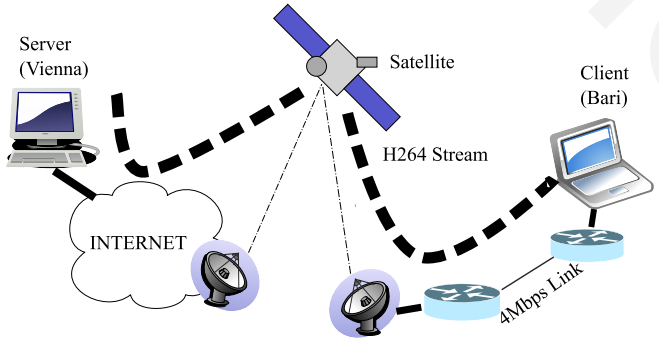


Fig. 14. Satellite testbed.

TABLE IV
QAS AND "NO QAS" COMPARISON OVER SATELLITE CONNECTIONS

		QAS	"No QAS"
Interruptions	Total Delay [s]	4 [0;18]	66 [6; 148]
	Num Interr	22 [0; 96]	195 [29; 408]
	Mean [ms]	135 [0; 142]	194 [135; 280]
	Maximum [ms]	472 [133; 1170]	1626 [946; 2635]
	95percentile [ms]	133 [133; 133]	386 [133; 750]
PSNR	Mean [dB]	27.5 [26; 29]	29 [29, 29]

effectiveness of QAS.

VI. CONCLUSION

The paper has proposed a novel end-to-end scheduling video streaming system, which has been referred to as QAS. The design has been carried out exploiting control theoretic arguments. The main objective of QAS is to avoid playout interruptions, prioritizing the transmission of base layer in respect to the enhancement layers. One of the strengths of the proposed approach is the independence from the underlying network. This has been confirmed by experimental results. They show that QAS is able to strongly limit the number of playout interruptions in several heterogeneous Internet scenarios, also including UMTS and satellite radio links. In this manner, we have highlighted: (i) the robustness of QAS with respect to radio links with narrow bandwidth (UMTS) and/or high RTT (satellite); (ii) the real utility of QAS in mobile ubiquitous environments. Encouraging results presented here will drive us toward further research devoted to the optimization of the idea here presented. In particular, the adoption of enhanced control schemes and rate-distortion oriented design will be considered for an enhanced version of proposed QAS.

REFERENCES

[1] Cisco, "The Exabyte Era," Jan. 2008, white paper.

[2] D. Wu, Y. Hou, W. Zhu, Y.-Q. Zhang, and J. Peha, "Streaming video over the Internet: approaches and directions," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282–300, Mar. 2001.

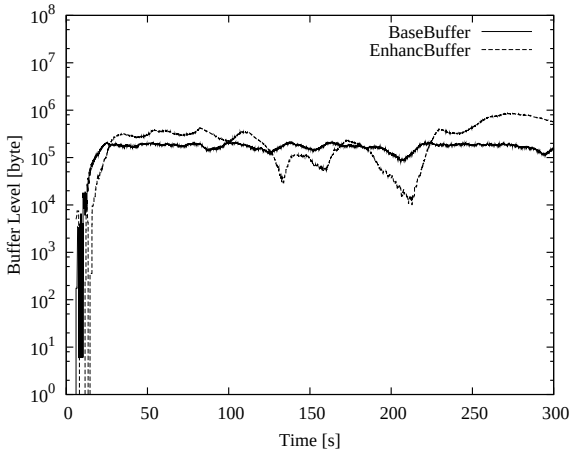
[3] S. F. Chang and A. Vetro, "Content-Aware Adaptive Media Playout Controls for Wireless Video Streaming," *Proc. of the IEEE*, vol. 93, no. 1, pp. 148–158, Jan. 2005.

[4] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.

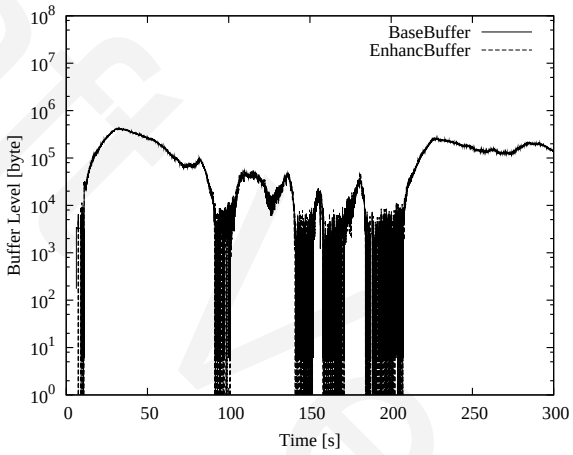
[5] H. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Trans. on Multimedia*, vol. 3, no. 1, pp. 53–68, Mar. 2001.

[6] R. Rejaie, M. Handley, and D. Estrin, "Layered quality adaptation for internet video streaming," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, pp. 2530–2543, Dec. 2000.

[7] M. Dai, D. Loguinov, and H. M. Radha, "Rate-Distortion Analysis and Quality Control in Scalable Internet Streaming," *IEEE Trans. on Multimedia*, vol. 8, no. 6, pp. 1135–1146, Dec. 2006.



(a) QAS



(b) "No QAS"

Fig. 15. Buffer levels over Internet paths including a satellite link.

[8] P. Zhu, W. Zeng, and C. Li, "Joint design of source rate control and qos-aware congestion control for video streaming over the internet," *IEEE Trans. on Multimedia*, vol. 9, no. 2, pp. 366–376, Feb. 2007.

[9] I. Djama, T. Ahmed, A. Nafaa, and R. Boutaba, "Meet in the middle cross-layer adaptation for audiovisual content delivery," *IEEE Trans. on Multimedia*, vol. 10, no. 1, pp. 105–120, Jan. 2008.

[10] D. Saparilla and K. W. Ross, "Optimal streaming of layered video," in *Proc. of IEEE INFOCOM*, Tel-Aviv, Israel, Mar. 2000, pp. 737–746.

[11] T. Wiegand, G. Sullivan, G. Bjntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[12] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An introduction to MPEG-2*. Chapman & Hall, 1997.

[13] H. Choi, J. W. Kang, and J.-G. Kim, "Dynamic and Interoperable Adaptation of SVC for QoS-Enabled Streaming," *IEEE Trans. on Consumer Electronics*, vol. 53, no. 2, May 2007.

[14] S. Wenger, Y.-K. Wang, and T. Schierl, *RTP Payload Format for SVC Video*, IETF Internet-Draft, work in progress, Feb 2008.

[15] S. Wenger, M. Hannuksela, M. Westerlund, D. Singer, and T. Stockhammer.

[16] J. Postel, "Transmission control protocol," IETF RFC 793, Sep. 1981.

[17] P. Papadimitriou, V. Tsaoussidis, and L. Mamatas, "A receiver-centric rate control scheme for layered video streams in the Internet," *Journal of Systems and Software*, vol. 81, no. 12, pp. 2396–2412, Feb. 2008.

[18] E. Gurses, G. B. Akar, and N. Akar, "A simple and effective mechanism for stored video streaming and server-side adaptive frame discard," *Computer Networks:The International Journal of Computer and Telecommunications Networking*, vol. 48, no. 4, pp. 489–501, 2005.

[19] P. Mehra, C. D. Vleeschouwer, and A. Zakhor, "Receiver-Driven Bandwidth Sharing for TCP and its Application to Video Streaming," *IEEE Trans. on Multimedia*, vol. 7, no. 4, pp. 740–752, Aug. 2005.

[20] A. Argyriou, "Using rate-distortion metrics for real-time internet video streaming with TCP," in *Proc. of IEEE Int. Conf. on Multimedia and Expo, ICME*, Toronto, Canada, Jul. 2006.

[21] P. Gill, M.Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in *Proc. of ACM SIGCOMM Internet Measurement Conf., IMC*, San Diego, CA, USA, 2007.

[22] J. Lazzaro, "Framing real-time transport protocol (RTP) and RTP control protocol (RTCP) packets over connection-oriented transport," IETF RFC 4571, Jul. 2006.

[23] T. Kim and M. H. Ammar, "Optimal quality adaptation for scalable encoded video," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 344–356, Feb. 2005.

[24] P. Amon, L. Haoyu, A. Hutter, D. Renzi, and S. S. Battista, "Scalable video coding and transcoding," in *IEEE Int. Conf. on Automation, Quality and Testing, Robotics, AQTR*, vol. 1, Cluj-Napoca, Romania, May 2008, pp. 336–341.

[25] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP friendly rate control (TFRC): Protocol specification," IETF RFC 2448, Jan. 2003.

[26] B. Xie and W. Zeng, "Rate distortion optimized dynamic bitstream switching for scalable video streaming," in *in Proc. IEEE Int. Conf. Multimedia and Expo*, Jun. 2004.

[27] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications*, IETF RFC 3550, Jul. 2003.

[28] I. Rhee and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," in *Third International Workshop on Protocols for Fast Long-Distance Networks*, Feb. 2005.

[29] (2009) Live555. [Online]. Available: http://www.live555.com.

[30] (2009) JSVM. [Online]. Available: http://ftp3.itu.ch/av-arch/jvt-site.

[31] K. J. Astrom and B. Wittenmark, *Computer controlled systems: theory and design*, 3rd ed. Prentice Hall, 1995.

[32] S. Mascolo, "Congestion control in high-speed communication networks using the Smith principle," *Automatica, Special Issue on Control methods for communication networks*, vol. 35, pp. 1921–1935, Dec. 1999.

[33] L. A. Grieco and S. Mascolo, "Adaptive rate control for streaming flows over the internet," *ACM Multimedia Systems Journal*, vol. 9, no. 6, pp. 517–532, Jun. 2004.

[34] V. Misra, W. Gong, and D. Towsley, "A fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proc. of ACM SIGCOM*, Stockholm, Sweden, Aug. 2000.

[35] C. Hollot, V. Misra, D. Towsley, and W. Gong, "A control theoretic analysis of RED," in *Proc. of IEEE INFOCOM*, Anchorage, Alaska, Apr. 2001.

[36] ——, "On designing improved controllers for AQM routers supporting TCP flows," in *Proc. of IEEE INFOCOM*, Anchorage, Alaska, Apr. 2001.

[37] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Addison Wesley, 1994.

[38] J. Crowcroft and I. Phillips, *TCP/IP and Linux protocol implementation: systems code for the Linux Internet*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[39] (2009) Web100. [Online]. Available: http://www.web100.org/

[40] (2009) Available test sequences. [Online]. Available: ftp://vqeg.its.bldrdoc.gov/SDTV/VQEG_PhaseI/TestSequences/ALL_625

[41] Y.-K. Wang, M. M. Hannuksela, S. Pateux, A. Eleftheriadis, and S. Wenger, "System and transport interface of svc," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1149–1163, 2007.

[42] (2009) Iperf. [Online]. Available: http://sourceforge.net/projects/iperf.