A Bandwidth Dependent Smoothing Algorithm for Interactive Video Streaming in UMTS Systems

Pietro Camarda
Politecnico di Bari
Via E. Orabona, 4
70125 – BARI (ITALY)
+39 080 5963642
camarda@poliba.it

Domenico Striccoli
Politecnico di Bari
Via E. Orabona, 4
70125 – BARI (ITALY)
+39 080 5963301
d.striccoli@poliba.it

Mariella Ragno
Politecnico di Bari
Via E. Orabona, 4
70125 – BARI (ITALY)
+39 080 5963301
mariellaragno@yahoo.it

ABSTRACT

In the recent past, a growing number of services, such VBR video transmission, have been implemented in UMTS cellular systems. In such a context, to reduce the high bit rate variability of VBR streams, several smoothing techniques, performed at server side, have been developed. They regularize the bit rate of transmitted data maintaining, at the same time, a constant video quality at receiving side.
In this paper, a novel smoothing algorithm, the Buffer Dependent Smoothing Algorithm (BDSA), has been developed and analyzed. It schedules VBR video data, taking into account both the feedback information on the real residual free buffer size coming from the client terminal, and the available bandwidth information. Numerical results show the BDSA effectiveness, in terms of losses, if compared with the classical smoothing algorithms known by literature.

Keywords

Video Streaming, UMTS, Smoothing, Interactivity, Available Bandwidth.

1. INTRODUCTION

In about ten years, cellular systems brought a significant revolution in the telecommunication world, developing a growing number of services for an ever growing number of users. Packetized data transmission, peculiar characteristic of Internet, is widely adopted in the Universal Mobile Telecommunication System (UMTS) standard, actually developed by the 3rd Generation Partnership Project (3GPP) organization [10]. UMTS is mainly thought for mobile multimedia content transmission (pictures, video streaming, video conference, TV programs, etc.) with relatively high Quality of Service degrees. One of the most important issues of UMTS systems is to transmit multimedia streams over UMTS networks, guaranteeing at the same time a high Quality of Service over wireless networks with highly variable conditions (variable bit rates, network delays, jitters, handover, etc.) [8]. Thus, the need to regularize the stream bit rate at transmission side, to avoid channel congestions, guaranteeing continuous and lossless playback at receiving side is particularly felt [4].

In Figure 1 a model of streaming architecture for UMTS is represented, considering the specifications illustrated in [8].

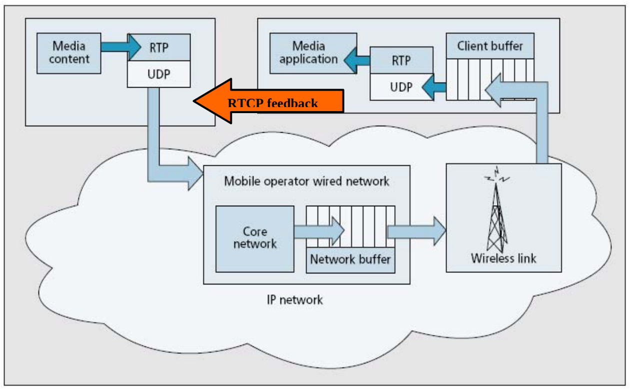

Figure 1. UMTS architecture for video streaming.

Multimedia streams are transmitted to the final users through the UMTS core network, utilizing the Real-time Transport Protocol (RTP) [20] over the User Datagram Protocol (UDP), particularly suitable for the transmission of real-time applications. As shown in Figure 1, the Real Time Control Protocol (RTCP) [1][17], implemented at transport layer, is utilized to send feedback information about the video stream transmission "status" to the server. This feedback channel is useful for two main reasons. Firstly, it enhances the user interactivity with the streaming server, allowing operations like the fast forward, rewind or pause on the video stream. Secondly, it periodically carries on several information on the main client terminal characteristics. This operation is performed through a RTCP packet called Next Application Data Unit (NADU). As specified in detail in [1], the fields of a NADU packet are: Playout Delay (PD) that indicates the time interval (in ms) between the actual Applications Data Unit (ADU) and the following, Next Sequence Number (NSN), representing the sequence number of the next packet to be codified, and the Free Buffer Space (FBS) that indicates the free space in the receiver buffer, expressed in multiples of 64 bytes. By the NADU examination, a large number of information can be derived that provide an important feedback on the dynamic "evolution" of the transmission. This can be fruitfully exploited

MOBIMEDIA 2007, August 27-29, Nafpaktos, Greece
Copyright © 2007 ICST 978-963-06-2670-5
DOI 10.4108/ICST.MOBIMEDIA2007.1691

to improve the quality of the provided service at server side.

Several types of codecs and multimedia contents are supported in the UMTS PSS standard [1]. In particular, to increase video quality, MPEG-4 and H.264 video compression standards are recommended [16][21]. Nevertheless, the traffic produced by such compressed stream is usually highly variable in time [9]. Furthermore, as stated in [7], the wireless channel fluctuations can easily bring to repeated errors that combine with the Variable Bit Rate (VBR) nature of compressed streams.
To improve wireless signal quality and reduce losses of VBR streams, a first solution could be the bit rate reduction through work-ahead smoothing techniques [19][6]. Smoothing algorithms presented in literature are all based on the reduction of the stream peak rate and bit rate variability by transmitting whenever possible, ahead of playback time, Constant Bit Rate (CBR) pieces of the video stream. The CBR entity varies from piece to piece according to a scheduling algorithm that smoothes the video stream bursty bit rate. The receiving buffer present in the UMTS terminal stores the smoothed data, and the original unsmoothed video stream leaves the buffer for decoding and playing. Obviously, smoothed stream bit rates must be chosen appropriately to avoid receiver buffer overflows and underflows, with the aim to guarantee a continuous playback at the client side without frame losses, as will be more clear in the sequel.

The main purpose of this work is to present a novel smoothing algorithm particularly suitable for on-line interactive applications in UMTS wireless networks. The proposed algorithm takes into account the feedback information on the buffer fill level, coming from the client terminal. It is able to take into account also the available bandwidth information to regularize the scheduled stream bit rate and reduce losses both for available bandwidth lack during transmission, and for buffer overflow/underflow at receiving side.

2. SMOOTHING PRINCIPLES

Let us suppose that a VBR video stream is composed by N video frames, each of them of size d_i bytes $(1 \le i \le N)$, as described in [6]. At the server side, the stream data are scheduled according to the particular smoothing algorithm. At the client side, the smoothed video data enter the buffer and the original unsmoothed video frame sequence leaves it for decoding and playout. Let us now consider the client buffer model in the k^{th} discrete time slot, assumed as the basic time unit. A discrete time slot, or frame time, is supposed to be the time interval in which a video frame is transmitted (1/25 s for PAL). To guarantee a feasible transmission, the cumulative input data to the receiving buffer at k^{th} discrete time, $S(k)$, should arrive quickly enough to avoid buffer underflow. The buffer underflow and overflow curves are respectively:

$$D(k) = \sum_{i=1}^{k} d_i \; ; \; B(k) = b + \sum_{i=1}^{k} d_i = D(k) + b \, . \tag{1}$$

So it has to be:

$$D(k) \le S(k) = \sum_{i=1}^{k} s(i) \le B(k) \tag{2}$$

where $s(i)$ represents the smoothed stream bit rate in the i^{th} discrete time slot. The smoothed stream transmission plan will result in a number of CBR segments, and the correspondent smoothed cumulative transmission plan $S(k)$ is given by a monotonically increasing and piecewise linear path that lies between the $D(k)$ and $B(k)$ curves [6].

Several studies on the impact of available bandwidth information on the smoothed transmission plan can be found in literature. An available bandwidth dependent smoothing algorithm, the Network Constrained Smoothing (NCS), is considered in [2]. It takes into account available bandwidth constraints and schedules the single video stream over a server-side transmission. This simple technique considers future network traffic knowledge to derive available bandwidth. The multimedia data are then divided into equal-sized intervals in which a CBR segment is scheduled.

Another example of bandwidth dependent smoothing algorithm can be found in [14]. In this work, network calculus is exploited to optimize the client buffer size, playback delay and look-ahead delay in such a way to generate a lossless video stream schedule respecting particular traffic envelopes, i.e., curves representing the maximum traffic that can be sent to the network. Smoothing developed in [14] can be applied in combination with other existing smoothing algorithms like the one illustrated in [19] to further minimize other metric, like number of bandwidth changes or rate variability.

A Rate Constrained Bandwidth Smoothing (RCBS) is presented in [5] for interactive video streams delivery. It minimizes the amount of buffering needed by smoothing when a maximum constant rate constraint is given, simply by prefetching video data only when the rate constraint is violated, and leaving the original unsmoothed data unchanged when they maintain under the bandwidth constraint. In this way, if compared with the classical smoothing techniques previously illustrated (CBA, MCBA, MVBA, etc.), the buffering needed for smoothing is greatly reduced and client buffers can store much more data for VCR functionalities (stop, pause, rewind and examine operations).

Other interesting works on video smoothing take into account the minimization of bandwidth occupied by smoothed VBR streams. In [3] a smoothing scheme is proposed, based on the $SLWIN(\alpha)$ smoothing scheme proposed in [18]. It dynamically adapts the sliding window size to smooth bursty traffic, minimizing occupied bandwidth and computational cost.

Another approach is suggested in [13], where a Monotonic Decreasing Rate (MDR) scheduler is implemented. It allows only monotonic decreasing rate allocations, reducing bandwidth requirements and greatly simplifying the admission test computational complexity, necessary to establish if a new video stream can be admitted to a system with limited bandwidth resources. A large number of simulations test the algorithm performance.

The dynamic bandwidth allocation issue for RCBR smoothed streams is tackled in [15]. The purpose of this work is twofold. Firstly, a source traffic prediction method is adopted. It is able to predict with sufficient accuracy bandwidth level changes of smoothed video traffic. Secondly, bandwidth prediction is used to decide in advance both channel rate and duration. RCBR algorithm is considered also in [12], where a network testbed is set up to analyze RCBR smoothing performance. RCBR scheme is chosen because it simplifies buffering and scheduling requirements in network switches for VBR streams. RCBR scheme is compared with traditional CBR schemes in this testbed, testifying significant improvements in terms traffic data loss.

In this work, a novel smoothing algorithm for VBR video streams, particularly suitable for UMTS network, is proposed and analyzed. It is called for simplicity Buffer Dependent Smoothing Algorithm (BDSA). It is an “on-line” algorithm, since several UMTS applications require an on-the-fly computation of the schedule, during stream running. The main novelty of this algorithm is that it takes into account the feedback information on the buffer fill level, periodically coming from the client terminal. This kind of approach can be very useful in the context of interactive applications, where the client can perform a variety of actions that cannot be known at server side, but could modify the buffer fill level accordingly. The feedback information on the real buffer fill level coming from the client can be exploited to modify on-the-fly the stream schedule at transmission side, in such a way to avoid bit losses due to buffer overflows and underflows. Regarding this aspect, the periodicity of RTCP control packets carrying the buffer information is generally comprised between 1 second (25 frame times, according to PAL standard) and 5 seconds (125 frame times). This time value is established at the beginning of the video stream session.

Furthermore, the proposed algorithm adapts the scheduled stream bit rate to the available bandwidth. Smoothed bit rate is reduced whenever available bandwidth drastically falls down and increased whenever it raises up again, to guarantee continuous decoding without quality degradation. Let us note that the available bandwidth profile should be known in advance, in the same temporal observation window where scheduled data will be transmitted in the network channel. Nevertheless, there are statistical predictive techniques of bandwidth estimation [11] that could predict the bandwidth behaviour, especially for noisy wireless channels, that is the case of UMTS applications. Since bandwidth prediction is not the purpose of this study, in this work we suppose that bandwidth profile is a priori known. When sufficient bandwidth is available for transmission, BDSA performs the "smoothest" transmission plan, with minimum scheduled rate variability and peak rate, exploiting the same basic principles of the MVBA schedule [19]. Let us note that BDSA can be fruitfully applied both to VBR and CBR video streams. In the latter case, the constant stream bit rate is varied in consequence of the available bandwidth and the free buffer information. Nevertheless, since the best smoothing performance are obtained for VBR streams, in this work we will consider only MPEG compressed VBR flows.

The rest of the paper is structured as follows. In Section 3 the BDSA is presented and analyzed, with particular reference to the off-line and on-line contexts. In Section 4 the BDSA is compared with the existing MVBA algorithm, that does not take into account the buffer feedback nor the available bandwidth information. Finally, in Section 5 some conclusion on the proposed algorithm are provided.

3. THE BDSA PRESENTATION

In this section, the BDSA algorithm is presented and illustrated. It is a server side algorithm that smoothes the VBR video stream taking into account both the buffer fill level information coming from the client terminal, and the available bandwidth profile. As specified in [17], the real free buffer level is periodically provided by RTCP packets to the streaming server as a multiple of 64 bytes. This novel algorithm has been essentially developed for on-line UMTS applications, but its principles can easily be applied to the off-line case, in which all the stream information is a priori known.

Let us suppose a temporal observation window of N video frames, and that the available bandwidth does not influence the schedule. The BDSA aim is to generate a transmission plan that minimizes both the scheduled peak rate and rate variability, always respecting the $D(k)$ and $B(k)$ bounds as defined by (1) and (2). Nevertheless, since the buffer information varies in time, the $B(k)$ curve is modified as follows:

$$B(k) = D(k) + b(k) \tag{3}$$

where $b(k)$ is the free buffer profile, considered as a function of the frame time k.

The algorithm acts as follows. Supposing to know the stream frame size d_k, the buffer variation profile $b(k)$ in a generic time interval $\left[k_1, k_2\right]$, the maximum bit rate $c_{\max}$ is calculated, as the maximum bit rate without overflowing the client buffer:

$$c_{\max} = \min_{k_1+1 \le k \le k_2} \left\{ \frac{B(k) - (D(k_1) + q)}{k - k_1} \right\} \tag{4a}$$

where q is the initial buffer level in k_1 (in bytes), $B(k)$ is defined by (3) and $D(k_1)$ is the lower bound in k_1.

The latest time instant where $c_{\max}$ is reached over $\left[k_1, k_2\right]$ is:

$$k_B = \max_{k_1+1 \le k \le k_2} \left\{ k \left| \frac{B(k) - (D(k_1) + q)}{k - k_1} = c_{\max} \right. \right\} \tag{4b}$$

Similarly $c_{\min}$ is defined as the minimum bit rate calculated without emptying the buffer:

$$c_{\min} = \max_{k_1+1 \le k \le k_2} \left\{ \frac{D(k) - (D(k_1) + q)}{k - k_1} \right\} \tag{4c}$$

and the latest time instant where $c_{\min}$ is reached over $\left[k_1, k_2\right]$ is:

$$k_D = \max_{k_1+1 \le k \le k_2} \left\{ k \left| \frac{D(k) - (D(k_1) + q)}{k - k_1} = c_{\min} \right. \right\}. \tag{4d}$$

The transmission plan is feasible only if $c_{\max} \ge c_{\min}$ in $\left[k_1, k_2\right]$.

Exploiting this concept, BDSA spans all the temporal window of N video frames. Starting from the first frame time k_1, in each discrete time k, $c_{\max}$, $c_{\min}$, k_B and k_D as defined in (4a)-(4d) are calculated. Besides, the maximum and minimum feasible bit rates in k are calculated, respectively:

$$\begin{cases} c_{\max}^{(k)} = \left\{ \dfrac{B(k) - (D(k_1) + q)}{k - k_1} \right\} \\ c_{\min}^{(k)} = \left\{ \dfrac{D(k) - (D(k_1) + q)}{k - k_1} \right\} \end{cases} \tag{5}$$

If there is $\bar{k}$ such that $c_{\min}^{(\bar{k})} > c_{\max}$, surely a buffer underflow occurs in $\bar{k}$; the scheduled bit rate will thus be $c_{\max}$ in $[k_1, \bar{k} - 1]$, with $k_1 = \bar{k} - 1$. Similarly, if $c_{\max}^{(\bar{k})} > c_{\min}$ a buffer overflow occurs. The scheduled bit rate will be $c_{\min}$ in $[k_1, \bar{k} - 1]$, and $k_1 = \bar{k} - 1$.

This procedure originates the "smoothest" transmission plan among all the feasible transmission plans. It means that it has the minimum rate variability and the minimum peak rate. The analytical proof of this can be found in [19].

The so built transmission plan takes into account the real buffer fill level information taken into account in $b(k)$. Nevertheless, BDSA considers also available bandwidth fluctuations. This further improvement makes the BDSA schedule more robust towards data transmissions over wireless channels, that are more subject to noise and bandwidth drops. To respect bandwidth limitations, BDSA implements a control over the scheduled bit rate $s(k)$, that must satisfy the following three conditions in each discrete time k:

$$S(k) = S(k-1) + s(k) \le B(k) \tag{6a}$$

$$S(k) = S(k-1) + s(k) \ge D(k) \tag{6b}$$

$$s(k) \le w(k) \tag{6c}$$

for each $1 \le k \le N$, and with the initial condition $S(0) = 0$. $w(k)$ is the available bandwidth information at the discrete frame time k. The (6) mean that the BDSA schedule faces available bandwidth drops, at the same time trying to avoid buffer overflows and underflows.

The logic followed by BDSA is based on the following main steps. If no bandwidth constraints are present, BDSA behaves exactly like MVBA, optimizing the transmission plan in terms of peak rate and rate variability. If available bandwidth is lower than needed by MVBA schedule, BDSA reduces the bit rate, adapting it to the available bandwidth profile. The scheduled bits exceeding the available bandwidth profile that would be lost because of bandwidth limitation are put in a "lost_bits" variable and redistributed along the entire transmission plan by increasing the scheduled bit rate $s(k)$ in the k^{th} frame time by Δ bits in the time intervals where available bandwidth allows it, that is, for each k respecting the condition: $s'(k) \le w(k)$.

This way of operation guarantees the condition (6c), but not (6a) and (6b). BDSA must thus prevent also buffer underflows and overflows. For this reason, BDSA first searches frame times in which buffer overflows or underflows occur. Let us note that, after the bit rate reduction due to the bandwidth limitation, the first critical condition found by BDSA, if any, is surely a buffer underflow. In fact, the MVBA schedule does never present buffer overflows; thus the bit rate decrease due to bandwidth limitations can only bring to buffer underflows. If a buffer underflow condition is found, the scheduled bit rate is increased by adding Δ_{und} bits in all the time period in which $S(k)$ does not respect the (6b), but always verifying the (6c). This amount of bits is taken from the "lost_bits" variable; if they are not sufficient to totally avoid the buffer underflow, the "lost_bits" variable will become null and losses will still occur. Let us note that this procedure of increasing the bit rate to avoid a buffer underflow, can also bring to a buffer overflow in other time intervals of the schedule.

If a buffer overflow occurs, the scheduled bit rate is decreased by reducing the bit rate of Δ_{ov} in a time interval that includes the entire overflow time period. Since this operation is a bit rate decrease, again the "lost_bits" variable increases and its content can be redistributed to compensate bit lacking due to buffer underflows. This suggests that this procedure can be iteratively repeated since the "lost_bits" variable reaches its minimum, or becomes null. Experimented bit losses will be given by the sum of the "lost_bits" variable and the losses occurred for buffer underflows.

The available buffer information is known by the stream server dynamically, in a time interval Δt_b, expressed in frame times, that ranges from 25 (1 second) to 125 (5 seconds). BDSA takes into account this information by modifying the schedule on-the-fly, during the stream running, and taking into account the last available buffer information coming to the server from the client. This is performed as follows. When the new buffer information $a_{real}(k \cdot \Delta t_b)$ arrives to the server at time $k \cdot \Delta t_b$, considered as a multiple of Δt_b, BDSA calculates:

$$a_{sched}(k \cdot \Delta t_b) = B(k \cdot \Delta t_b) - S(k \cdot \Delta t_b) \tag{7}$$

and compares it with $a_{real}(k \cdot \Delta t_b)$. If $a_{real}(k \cdot \Delta t_b) \ne a_{sched}(k \cdot \Delta t_b)$, only the remaining part of the schedule, starting from $k \cdot \Delta t_b$ until its end, is modified, by taking into account $a_{real}(k \cdot \Delta t_b)$. This procedure is repeated each Δt_b seconds, when the new buffer information arrives to the server, until the end of the schedule.

In Figure 2 the implementation of the BDSA taking into account the on-the-fly buffer information is presented, through its pseudo-code.

```
1.  Load stream (N video frames);
2.  Assign $a_{sched}(1) = b$;
3.  Assign $d(i)$, $w(i)$  $\forall 1 \le i \le N$;
4.  $D(i) = \sum_{j=1}^{i} d(j)$;  $B(i) = \min[D(i-1) + a_{sched}(1), D(N)]$;
5.  $S[1:N] = BDSA[D, a_{sched}(1), B]$;
6.  Assign $\Delta t_b$
7.  $k = 1$;
8.  WHILE $k\Delta t_b \le N$
9.      Store $a_{real}(k \cdot \Delta t_b)$ coming from client;
10.     IF $a_{real}(k \cdot \Delta t_b) \ne a_{sched}(k \cdot \Delta t_b)$
11.         $B'(i) = \min[D(i-1) + a_{real}(k\Delta t_b), D(N)]$  $\forall k \cdot \Delta t_b \le i \le N$;
12.         $S[k \cdot \Delta t_b : N] = BDSA[D, a_{real}(k \cdot \Delta t_b), B']$
13.     END
14.     $k = k + 1$;
15.     $a_{sched}(k \cdot \Delta t_b) = B(k \cdot \Delta t_b) - S(k \cdot \Delta t_b)$
16. END
```

Figure 2. Pseudo-code for the BDSA application.

Lines from 1 to 4 define the main parameters: video stream frames, initial buffer level, available bandwidth and upper and lower limits. Line 5 calculates the entire BDSA schedule with the initial buffer level b. Supposing the real available buffer level $a_{real}(k \cdot \Delta t_b)$ coming at the server each Δt_b seconds (line 6), it is compared with $a_{sched}(k \cdot \Delta t_b)$ derived by the schedule through (7). If the two values are different (line 10), the schedule is calculated in line 12, starting from $k \cdot \Delta t_b$ and exploiting the modified values of the buffer capacity $a_{real}(k \cdot \Delta t_b)$ and consequently of the upper bound B' (line 11). Line 15 calculates the new available buffer capacity after the next Δt_b seconds as derived by the schedule, to compare it with the real available buffer coming during the following iteration. The procedure of lines 9-15 is repeated until the stream end.

4. BDSA PERFORMANCE

In this Section, we test the BDSA effectiveness by comparing it with the MVBA smoothing algorithm already presented in literature, taking into account available bandwidth and buffer size fluctuations together with receiving smoothing buffer sizes information. Experiments are repeated for different VBR video streams. We choose MVBA as meter of comparison, because is the most effective in reducing the scheduled peak rate and the rate variability of a VBR video stream.

Figure 3 represents a first comparison between BDSA and MVBA schedules. Simulation has performed for a piece of 10.000 video frames of the “Star Wars” film, MPEG-4 compressed with high quality. In Figure 3 the cumulative schedule has been represented for both algorithms, together with the buffer overflow (upper bound) and underflow (lower bound) curves, in a time interval of 3.000 frame times.

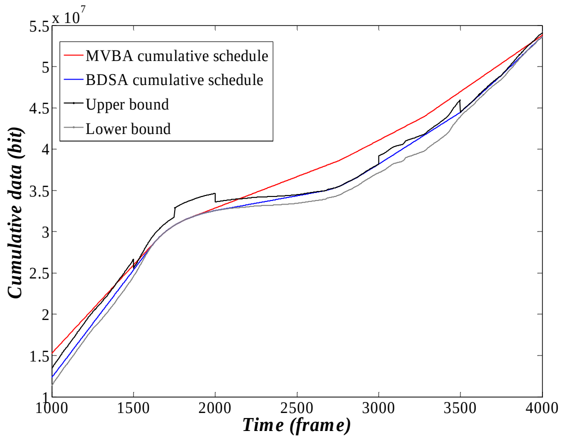

Figure 3. A zoom of BDSA and MVBA cumulative transmission plans.

BDSA schedule has been obtained by randomly varying the buffer capacity; the effect of this variation is testified by the abrupt changes of the buffer overflow profile in Figure 3. The MVBA schedule is calculated supposing a constant buffer capacity of 512 kB, which corresponds to the maximum buffer level simulated for MVBA. In this experiment, no available bandwidth limitation has been supposed, so that it does not influence the BDSA transmission schedule. As can be clearly observed by Figure 3, MVBA schedule repeatedly crosses the upper bound curve, thus provoking a buffer overflow at receiving side. BDSA schedule remains always confined between the upper and lower bounds; no losses occur. From this considerations it can be understood that MVBA performs always worse than BDSA since the former does not take into account the buffer information coming to the server from the client terminal.

Now let us analyze BDSA and MVBA performance in presence of available bandwidth limitations. This situation can happen very often in the context of cellular UMTS networks, where data transmission occur in the open space and are subject to noise and interferences. Figure 4 represents this comparison between BDSA and MVBA. The available buffer randomly varies between 128 kB and 512 kB, starting with this last value. To further stress the system, a bandwidth drop has been simulated at the beginning of the transmission of a piece of “The silence of the lambs” video stream, MPEG-4 coded with high quality. The observation window has length 10.000 video frames. The drop occurs during the first 2.000 video frames. In this way, BDSA can not perform an effective work-ahead schedule to avoid in advance bit losses for bandwidth limitation; it is forced to redistribute bits that would be lost only after the drop.

As can be seen from Figure 4, BDSA follows the bandwidth profile, and compensates the lower scheduled bit rate, if compared with MVBA, by increasing it immediately when available bandwidth rises again. The difference between the two schedules is clearly visible until the 3.000th frame time; after that, BDSA and MVBA schedules coincide, according with the BDSA aim to maintain the MVBA main features. Losses experimented by MVBA are much more than BDSA ones. In fact, MVBA losses occur both at transmission side, due to the initial bandwidth drop, and at receiving side, due to buffer underflow. This underflow occurs because the amount of bits arriving at receiver are less than scheduled by MVBA, because of the hard bandwidth limitation. BDSA will instead experiment only losses for buffer underflow, due to its ability to take into account the bandwidth limitation.

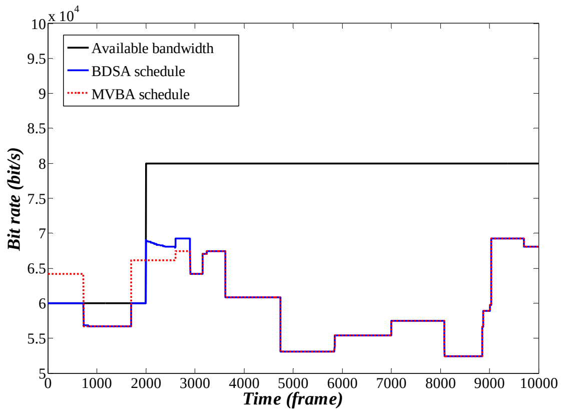

Figure 4. Another example of BDSA and MVBA schedules with an initial bandwidth drop.

Figure 5 represents the distribution of BDSA bit losses during the bandwidth drop simulated in Figure 4. They all occur during the first 1.000 video frames, because of buffer underflows. This is obvious, since the bandwidth drop occurs at the beginning of the schedule and BDSA is not able to prevent frame losses by increasing the scheduled bit rate in advance.

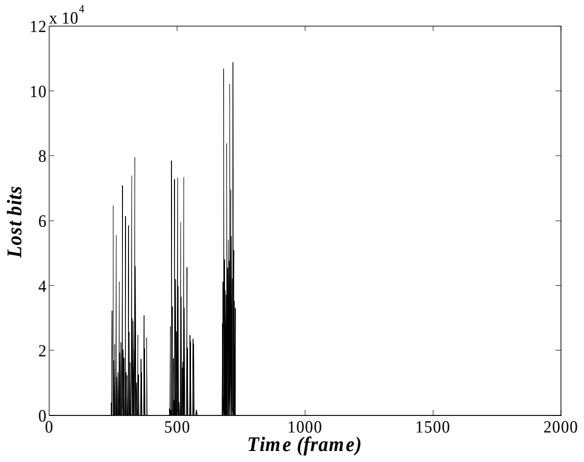

Figure 5. Lost bit distribution for buffer underflow during the bandwidth drop.

Another scenario has been implemented by simulating an available buffer reduction in correspondence of a bandwidth drop in a time window of 10.000 video frames, to simulate the simultaneous lack of bandwidth and buffer resources. Figure 6 shows the resulting schedules. Available buffer profiles lowers from 512 kB to 128 kB twice: between the 2.500th and the 4.000th frame times, and between the 7.000th and 8.000th frame times. The bandwidth profile is illustrated in Figure 6.

The particularity of the result shown in Figure 6 is that BDSA schedule does not follow the bandwidth profile during the last part of the bandwidth drop (between the 4.000th and 4.500th video frames). This happens because the available buffer requirement is more stringent than available bandwidth, forcing the BDSA schedule to further lower its bit rate. Bit rate is increased immediately after the bandwidth drop, when also the available buffer value is of 512 kB. Also in this case, MVBA presents consistent bit losses during the bandwidth drop.

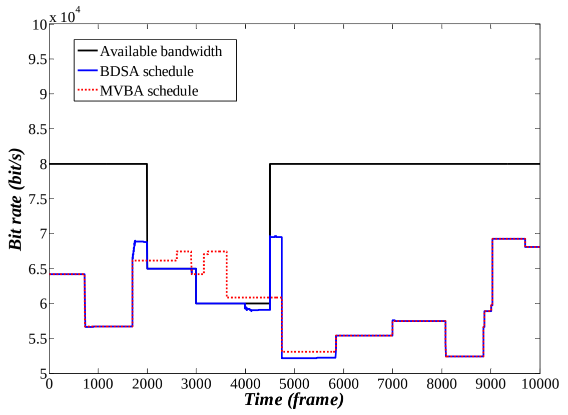

Figure 6. BDSA and MVBA schedules in presence of simultaneous available bandwidth and buffer drops.

5. CONCLUSIONS

In this paper a new smoothing algorithm, BDSA, has been proposed, that regulates the transmitted bit rate of video streams transmitted over UMTS networks taking into account available bandwidth constraints and the feedback information on available buffer periodically coming from the client. Thanks to a simple and efficient bit redistribution along the entire schedule, BDSA is able to minimize frame losses. Its great flexibility of implementation in several situations (off-line or on-line smoothing of single VBR or CBR video stream, or stream aggregates) makes it particularly suitable to be implemented in real-time video streaming in UMTS systems, where critical available bandwidth conditions often occur.

6. REFERENCES

[1] 3GPP TS 26.234. Transparent End-To-End Packet-Switched Streaming Service (PSS): Protocols and Codecs (Release 6). http://www.3gpp.org/ftp/Specs/archive/26_series/26.234/26234-630.zip.

[2] Bewi, C., Pereira, R., Merabti, M. Network Constrained Smoothing: Enhanced Multiplexing of MPEG-4 Video. In *Proceedings of the 7th International Symposium on Computers and Communications (ISCC'02)* (Taormina, Italy, July 1-4, 2002), 114-119.

[3] Chang, R.-I, Chen, M.-C., Ho, J.M.. and Ko, M.T. An Effective and Efficient Traffic Smoothing Scheme for Delivery of Online VBR Media Streams. In *Proceedings of the IEEE INFOCOM'99* (New York, USA, Mar. 1999).

[4] Elsen, I., Hartung, F., Horn, U., Kampmann, M., and Peters, L. Streaming Technology in 3G Mobile Communication System. *IEEE Computer* (Sept 2001).

[5] Feng, W.-C. Rate-constrained bandwidth smoothing for the delivery of stored video. In *Proceedings of the SPIE Multimedia Computing and Networking* (San Jose, CA, Feb. 1997).

[6] Feng, W.-C., and Rexford, J. Performance Evaluation of Smoothing Algorithms for Transmitting Prerecorded Variable-Bit-Rate Video. *IEEE Transactions on Multimedia,* 1, 3 (Sept. 1999), 302-313.

[7] Fitzek, F., Seeling, P., and Reisslein, M. Video streaming in wireless internet. *Electrical Engineering and Applied Signal Processing Series* (2004).

[8] Fröjdh, P., Horn, U., and Kampmann, M. Adaptive Streaming within the 3GPP Packet-Switched Streaming Service. *IEEE Network* (Mar./Apr. 2006).

[9] Grossglauser, M., and Bolot, J.-C. On the Relevance of Long-Range Dependence in Network Traffic. *IEEE/ACM Transaction on Networking*, 7, 5 (Oct. 1999), 629-640.

[10] Holma, H., and Toskala, A. WCDMA for UMTS : Radio Access for Third Generation Mobile Communications. John Wiley, 2004, 3rd edition.

[11] Hu, N., and Steenkiste, P. Evaluation Characterization of Available Bandwidth Probing Techniques. *IEEE Journal on Selected Areas in Communications*, 21, 6 (Aug. 2003), 879-894.

[12] Kishore, M., and Liang, Y. An Empirical Study on Renegotiated CBR for VBR Video Services Based on Network Testbed. *IEEE Transactions on Broadcasting*, 52, 3 (Sept. 2006), 362-367.

[13] Lai, H., Lee, J.Y., and Chen, L. A Monotonic Decreasing Rate Scheduler for Variable-Bit-Rate Video Streaming. *IEEE Transactions on Circuits and Systems for Video Technology*, 15, 2 (Feb. 2005), 221-231.

[14] Le Boudec, J.-Y., and Thiran, P. Network Calculus: A Theory of Deterministic Queueing Systems for the Internet. *Book Springer Verlag* (May 2004).

[15] Lee, M. Video Traffic Prediction Based on Source Information and Preventive Channel Rate Decision for RCBR. *IEEE Transactions on Broadcasting*, 52, 2 (Jun. 2006), 173-183.

[16] Mitchell, J.L., Pennebaker, W.B., Fogg, C.E., and LeGall, D.J. MPEG video compression standard. Chapman & Hall, 1996.

[17] Ott, J., Wenger, S., Sato, N., Burmeister, C., and Rey, J. Extended RTP Profile for RTCP-based Feedback (RTP/AVPF). Internet Draft, draft.ieft.avt.rtcp.feedback-08.txt (Jan. 2004).

[18] Rexford, J., Sen, S., Dey, J., Kurose, J.,and Towsley, D. Online Smoothing of Variable-Bit-Rate Streaming Video. *IEEE Transactions on Multimedia*, 2, 1 (Mar. 2000), 37-48.

[19] Salehi, J.D., Zhang, Z.-L., Kurose, J., and Towsley, D. Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements Through Optimal Smoothing. *IEEE/ACM Transactions On Networking*, 6, 4 (Aug. 1998), 397-410.

[20] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. *RTP: A transport protocol for real time applications*. RFC 3550 (Jul. 2003).

[21] Stochkammer, T., and Hannuskela, M.M. H. 264/AVC video for wireless transmission. *IEEE Wireless Communications* (Aug. 2005).