PRE-PRINT VERSION

A scheduling Algorithm for Interactive Video Streaming in UMTS Networks

R. Laraspata, D. Striccoli, P. Camarda
DEE–Politecnico di Bari
v. Orabona, 4 – 70125, Bari, Italy
Email:{r.laraspata, d.striccoli, p.camarda}@poliba.it;

***Abstract*—The importance of a Variable Bit Rate (VBR) video transmission on UMTS networks is increasing in time. The bursty nature of VBR traffic complicates the design of efficient mechanisms for video retrieval, transport, and provisioning to achieve a high bandwidth utilization and reduce the negative effects of bandwidth fluctuations in wireless channels. To this aim, several scheduling algorithms can be successfully implemented. They regulate data transmission to reduce the rate variability peculiar of VBR streams. At client side, scheduled data are temporarily stored in the client buffer before being decoded on the terminal. In this work, a novel scheduling algorithm, the Scheduling Algorithm for Interactive Video (SAIV) is presented and analyzed. It is an algorithm thought for VBR stream transmission in UMTS networks that takes into account the user interactivity. Scheduling is performed "online", over relatively small video segments to reduce delays. SAIV dynamically varies the sampling frequency of the Real Time Control Protocol (RTCP) feedbacks that carry information on the client buffer status. The sampling frequency is modulated according to the difference between the calculated buffer fill level at server side and the real buffer fill level at client side. The latter is exploited to reschedule data with the updated information. Numerical results testify the SAIV effectiveness compared to the classical SLWIN online algorithm already known by literature, in some simulation scenarios of real interest.**

Keywords: Adaptive Scheduling, UMTS, VBR video, RTCP feedback, User interactivity.

I. Introduction

For several years mobile and fixed networks, as Internet, were not communicating each other. Nowadays, services to end users provide a diversified and personalized range of applications to anyone, anywhere, anytime. Users can combine telecommunications, information technology and entertainment services that various operators offer. The main aim of the Universal Mobile Telecommunication System (UMTS) is to combine the most important trend of telecommunications market and allow customers to access efficiently to a wide range of data and applications. In this way, the standard meets the growing needs of mobility, flexibility and opportunity of choice. UMTS standard [1] provides to mobile users the same several multimedia applications, typically used in wired networks. UMTS network supports both pre-existing services and offers quite new revolutionary services including broadband Internet access. It implements interactive and multimedia services in addition to voice, text, picture and audio/video contents that the Global System for Mobile Communications (GSM) standard already provided. All of these services were independent applications until now. Traditionally telecommunication environments have been integrated with vertical business and technology segmentation. In 3G communication system a horizontally seamless layer service network integrates the Internet transport Protocol (IP) into a mobile service environment, making new opportunities for IP-based mobile applications [2]. The 3G system provides many different services with different Quality of Service (QoS) guarantees. UMTS service classes require end-to-end QoS support. For this reason the Third Generation Partnership Project (3GPP) labels four main QoS classes to data bearer. One of this is the streaming class [3]. Thanks to the increasing transmission capacities of wireless communication networks, video streaming becomes an interesting and feasible application. End users can click on a link using web browser on their mobile devices and play the selected video clip. The client can start playing the video few seconds after the first part of the multimedia contents have received and before the download from the streaming server is completed [4]. Both the highly fluctuating conditions of wireless links and the limited amount of buffering on mobile terminals strongly influence the correct delivery of audio and video contents to the terminals. Despite the highly variable bandwidth conditions of wireless channel and the relatively high data bit rates, UMTS systems should ensure continuous and lossless data delivery. 3GPP standardized the mobile packet-switched streaming service, commonly referred to as the 3G-PSS standard [5]. Figure 1 shows a simplified UMTS architecture for packet-switched operations [6][7].

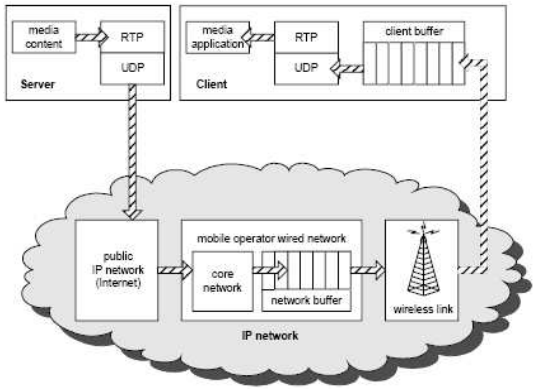

Fig. 1. Packet-switched architecture in UMTS network

The 3G-PSS standard integrates video, audio, images and formatted text into mobile multimedia applications at the same time. It defines both the protocols and the codecs for streaming video. The streaming server sends Real-time Transport Protocol (RTP) packets to the client through IP network using User Datagram Protocol (UDP) [8]. IP network is composed by other three subnets: a public network (e.g. Internet), the core network of the mobile operator and the wireless link that often acts as a bottleneck. The streaming server is either placed in a public network like the Internet or in the operator's network directly. A public network is present only if the server is not physically located within the mobile operator's network. RTP/UDP are fully compliant with real time transmission of multimedia applications. UDP is a protocol at transport layer of the ISO-OSI stack while RTP is a upper-layer session protocol [8]. RTP packets are encapsulated in UDP datagram. The sent packets are temporarily buffered at the Radio Network Controller (RNC). At receiving side the client stores packets in its buffer and passes them to the specific media application. Two major problems regarding buffer occupancy can appear, that must be solved to achieve a satisfactory QoS for wireless streaming. The first one is the client buffer underflow at the client side. This results to continuous rebufferings, because there are no more available packets in the client buffer. The application is forced to interrupt playout in order to wait for new data. The second problem is the client buffer overflow. Buffers on mobile terminals often have very limited size. If the transmission rate is higher than the playout rate, the buffer quickly fills. In this case the subsequent packets are dropped. The PSS standard by 3GPP-SA4 working group proposed an extension of RTP protocol: the Real Time Control Protocol (RTCP) to overcome these problems [9]. RTCP was introduced in order to control the transmission rate for wireless multimedia streaming and the status of the streaming client buffer. A further complication in video transmission is the video compression, such as MPEG, that introduces a burstiness that jeopardizes the high quality of compressed video streams and the transmission efficiency. Variable Bit Rate (VBR) traffic complicates the design of efficient real time storage, retrieval, transport, and provisioning mechanism to achieve high resource utilization [10]. VBR data, in combination with bandwidth fluctuations of the wireless channel, could easily bring to a higher error probability at transmission side [11]. For all these reasons, it is very important to reduce the bit rate variability of VBR streams at the same time guaranteeing a continuous and lossless playback at receiving side. To this aim VBR media streams are scheduled for transmission at server side [12]. Scheduled data are transmitted ahead of their playback time. The transmission schedule is built in such a way to avoid buffer overflow and underflow at client buffer side for the whole playback duration, achieving a considerable reduction in rate variability [13].

In this paper, we present a novel scheduling algorithm to be implemented in UMTS systems for real-time interactive multimedia applications. This algorithm takes into account the user interactivity, that could cause buffer overflows and underflows. It exploits RTCP feedbacks to avoid these events. The algorithm is based on the idea to dynamically change the RTCP sampling frequency according to user interactivity. In this way the amount of scheduled data is modulated according to the specific user actions.

II. RELATED WORK

In this section we describe scheduling principles and RTCP feedback. RTCP reports are periodically exchanged between the streaming server and the client to improve the QoS of the video streaming. Their main function is to make available at the streaming server a feedback about the bit distribution in the client buffer. Since the transmission control adapts the rate both to playout status and channel conditions, it can solve these problems at best.

A. Principles of Scheduling

We consider a compressed VBR video stream that consists of N frames and the generic frame i is d_i bytes long, $1 \leq i \leq N$. The server schedules data transmission according to the specific algorithm. Let

$$D(k) = \sum_{i=1}^{k} d_i \tag{1}$$

the amount of data consumed at the client up to the k_{th} discrete frame time. A frame time is the basic time unit, defined as the time to decode a frame (1/25 s). Similarly, to prevent buffer overflow, the client should not receive more than

$$B(k) = b + \sum_{i=1}^{k} d_i = D(k) + b \tag{2}$$

where b is the client buffer size. The curves (1) and (2) represent respectively the cumulative underflow and overflow curves, both non decreasing in time. To avoid losses a feasible cumulative transmission plan, that we call $S(k)$, must verify the following condition:

$$D(k) \leq S(k) = \sum_{i=1}^{k} s_i \leq B(k) \tag{3}$$

where s_i is the scheduled stream rate in the i_{th} frame time. Eq. (3) holds in general for scheduling algorithms. In the off-line case, scheduling is performed over the entire stream length. For on-line algorithms, the transmission plan is calculated at server side on consecutive temporal observation windows of N frame times partially overlapped in time [10]. The algorithm in [10] computes the transmission schedule with minimum peak rate and bit rate variability, and significantly reduces effective bandwidth of variable-bit-rate video streams. The calculated transmission plan is based on the existing video frames. It is constrained by the limits in (3) and is composed by longest Constant Bit Rate (CBR) segments. As shown in [14] windows overlapping allows a further schedule optimization in this sense. The scheduling

algorithm proposed in this work reduces the peak-rate and rate variability in temporal windows of limited size, at the same time performing an on-the-fly computation of transmission plan. We illustrate an on-line schedule that takes into account the user interactivity. The user actions can change the status of the receiving terminal, in terms of decoded frames and buffer occupancy level. The server reschedules data according to the feedback RTCP information on the terminal status. The RTCP packets give useful information about client buffer occupancy with a constant periodicity. According to the PAL standard, it periodicity is comprised between 1 second (25 frame times) and 5 seconds (125 frame times). Generally this value is established at the beginning of the video transmission. In the next subsection, RTCP main features will be explained in detail.

B. RTCP Feedback

The aim of the transmission control is to achieve an efficient client buffer management through several traffic parameters, , carried by the Receiver Report (RR) in RTCP protocol [14]. RTCP packets are generated at regular time intervals. So, the client traffic parameters are sampled with a constant frequency and sent back to the streaming server with the same constant frequency. We can make use of the several fields in RTCP standard to know the status of client buffer. The Highest Received Sequence Number (HRSN) is the sequence number of the last packet arrived at the client buffer. The Highest Transmitted Sequence Number (HTSN) is the sequence number of the last packet sent by the server. The Oldest Buffered Sequence Number (OBSN) is the sequence number of the next packet to be played out at the time the RR is built. The Playout Delay (PD) is the time interval before the OBSN packet is played out. The NSN (Next Sequence Number) represents the sequence number of the next packet to be decoded [15]. The server can keep track of the HTSN to calculate the amount of sent packets [14]. Therefore, RTCP protocol carries several useful information as the playout delay, the sequence number of received packets and, in consequence, the residual buffer space (expressed in multiple of 64 bytes). Based on these information, the streaming server regulates the bit rate of transmitted data according to terminal status, like buffer size and the user action (pause, fast forward, etc.), that strongly influence the buffer fill level during stream running. The bit rate regulation is performed with the same sampling frequency given by RTCP information. Only in these time instants, in fact, the server knows exactly the real status of the client terminal, and can reschedule data accordingly. Fig. 2 shows how the streaming server uses this information to adjust the transmission rate whenever network congestion or losses at receiving side occur.

The 3GPP standard considers the evolution of RTCP packet, called the Next Application Data Unit (NADU) [6]. The NADU packet carries several information, such as the number of packets reached the client, the number of packets stored in the client buffer and playout by the user. These parameters are illustrated in Fig. 3.

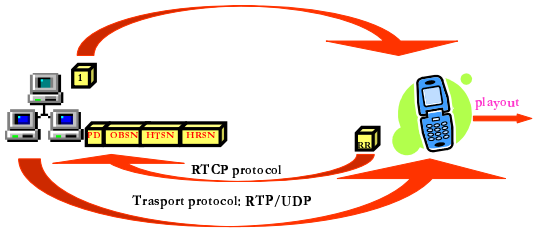

Fig. 2. RTCP protocol

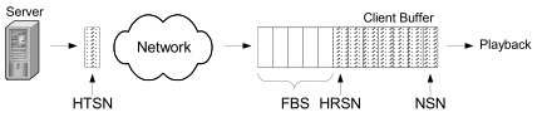

Fig. 3. Parameter of NADU

If a packet has a sequence number less than NSN it has already been decoded. If the streaming server knows the size of each transmitted packets, HRSN and NSN trace the level of the client buffer. The Free Buffer Space (FBS) fields describe the client status. In the generic k_{th} frame time it holds:

$$NSN(k) = HRSN(k) + FBS(k) - b + 1 \qquad (4)$$

Where b is the client buffer size. The quality of the delivered data depends on the variable conditions of the wireless link and the user equipments limited amount of memory. RTCP can be a good solution to enhance the wireless multimedia streaming QoS over IP networks. In the next section we introduce a modulated frequency for RTCP packets, dynamically variable according to the real buffer fill level information carried by RTCP packets. We exploit this information to dynamically vary the frequency of the RTCP packets coming to the server. We illustrate that this method improves the results in the scenarios of practical interest.

III. The Scheduling Algorithm for Interactive Video (SAIV)

In this section we introduce a novel algorithm , the Scheduling Algorithm for Interactive Video (SAIV), thought for real-time interactive multimedia applications. SAIV tries to reduce all problems discussed in the previous sections and considers that the end user performs other actions than playback. It calculates the residual free buffer space and the new schedule according to RTCP information. It takes into account the user interactivity to avoid continuous rebufferings and bad media quality caused by buffer underflows and overflows respectively. To face these problems we propose a dynamic scheduling at the server side. The external user actions like pause or fast forward, change the client buffer fill level and can easily bring to buffer underflows and overflows, increasing the probability of losses and rebufferings. According to the difference between the calculated buffer fill level at server side and the real buffer fill level at client side, RTCP packets sampling frequency is varied. If the user stops playback and server does not quickly react a buffer overflow occurs. On

the other side, fast forward actions can cause underflow. The proposed algorithm increases or decreases the frequency of RTCP packets according to user interaction. We focus on the effects of the user interaction that modify the status of the mobile device. Scheduling is performed over partially overlapped temporal windows. The overlap degree varies dynamically, according to the varying frequency of the feedback information carried by RTCP packets. The variation of the RTCP feedbacks frequency is implemented as follows. Let us suppose that the streaming server sends VBR video to a 3G device. The server reduces the video bit rate variability through the work-ahead scheduling as explained in [12]. The server schedules data in partially overlapping windows of size N frame times.

Now, let us suppose that a NADU packet arrives to the server in the k^{th} frame time. The server calculates through the schedule the expected free buffer level $FBS_s(k)$, as eq. (5),

$$FBS_s(k) = B(k) - S(k) > 0 \tag{5}$$

and then it compares it with the real free buffer space in the k^{th} frame time, coming from the client through RTCP, that we call $FBS_c(k)$.

If the user performs only video playback, surely it holds:

$$FBS_s(k) = FBS_c(k) \tag{6}$$

On the contrary, if the user performs some actions like pause or fast forward, the expected buffer level and the real buffer level will be different, and $FBS_c(k)$ will depend on the specific user action. We suppose only two kinds of events: pause, and 2x and 4x fast forward. In the first case, if the user stops the playback, the data will only enter the buffer but will never leave it. In this case, since the server supposes playback with data also leaving the buffer, it will be $FBS_c(k) < FBS_s(k)$. Vice versa, in the case of Fast Forward action, it will be $FBS_c(k) > FBS_s(k)$. Nevertheless, SAIV performs more. It varies the frequency of the feedback information according to the quantity:

$$|\Delta B| = |FBS_s(k) - FBS_c(k)| \tag{7}$$

In fact, fast forward actions increase the underflow probability, but the server knows the precise sequence number $HRSN(k)$ at each RTCP feedback. So the server can resend scheduled frames starting by $HRSN(k)$, because the buffer is empty. As a consequence, the user will perceive only a delay in frame decoding due to the rebuffering. Frame losses due to buffer underflow will never occur. In the case of buffer overflow, instead, losses will occur because the server continuously sends data in a full buffer and data exceeding the buffer size will be dropped. So $HTSN(k) > HRSN(k)$ and the server is not able to identify and send lost frames again. If the server is quickly able to react through rescheduling, both these events can be efficiently prevented. Each action other than playback performed by the user changes the free client buffer level. SAIV compares it with the free buffer level $FBS_s(k)$ calculated by the server. And according to (7), the larger the difference $FBS_c(k) - FBS_s(k)$, the larger $|\Delta B|$. SAIV dynamically varies the RTCP sampling frequency and consequently the slide length α according to $|\Delta B|$, with $0 \leq |\Delta B| \leq b$. Let us note in fact that each time a RTCP packet arrives to the server, the schedule must be calculated again in the time window. This is equivalent to sliding the time window by the time interval between two consecutive RTCP packets. The RTCP periodicity thus coincides with α. Since it is reasonable that small $|\Delta B|$ increments cause high α decrements, to more quickly modulate the schedule to the user external actions, SAIV implements the following simple hyperbolic relationship between α and ΔB:

$$|\Delta B| = \frac{a_1}{\alpha} + a_2 \tag{8}$$

The parameters a_1 and a_2 are calculated by imposing the bound conditions:

$$\begin{cases} |\Delta B| = 0 \Rightarrow \alpha = \frac{N}{2} \\ |\Delta B| = b \Rightarrow \alpha = 1 \end{cases} \tag{9}$$

The first of (9) means the same fill buffer level at the client and server side. In this case we assume for α the value proposed in [10], that is a good compromise between an efficient reduction of the video burstiness and the computational overhead due to windows overlap. In the second of (9) we increase the sampling frequency at its maximum allowed, that is, a RTCP packet each frame time ($\alpha = 1$). We chose $\alpha = \frac{N}{2}$ as the maximum $slide-length$, because it is the best trade-off between the optimum schedule and the computational overhead of the algorithm, as explained in [10]. The system (9) brings to:

$$\begin{cases} a_1 + a_2 = b \\ \frac{2a_1}{N} + a_2 = 0 \end{cases} \Rightarrow \begin{cases} a_1 = \frac{Nb}{N-2} \\ a_2 = -\frac{2b}{N-2} \end{cases} \tag{10}$$

By (8), replacing a_1 and a_2 and deriving α as a function of ΔB we obtain:

$$\alpha(\Delta B) = \frac{Nb}{(N-2)\,|\Delta B| + 2b} \tag{11}$$

According to eq. (11) the server updates dynamically the frequency of RTCP sampling and rescheduling.

Resuming the SAIV behavior, when a RTCP packet comes in k, the server compares $FBS_c(k)$ and $FBS_s(k)$ and calculates $\Delta B = FBS_s(k) - FBS_c(k)$. Depending on ΔB the server calculates the next RTCP feedback time as:

$$k_1(\Delta B) = k + \alpha(\Delta B) \tag{12}$$

Then the server sends this information back to the client. The server then updates the $NSN(k)$ information by (4), it reschedules video data and sends the first α scheduled frames, until the updated $FBS_c(k_1)$ information comes again from the client. And so forth until the stream end.

IV. Experimental Results

In this section we test SAIV with $slide-length$ $1 \leq \alpha \leq (N/2)$ and compare it with the algorithm in [10], that we call SLWIN, with a constant $slide-length$ $\alpha = N/2$. We fix a window length of $N = 600$ frame times for SLWIN. All time units are expressed in frame times. The comparison between SAIV and SLWIN has been obtained by simulating the transmission of 40.000 video frames of four videos, MPEG-4 coded with high quality. Table I illustrates the main statistics of the four considered movies.

TABLE I
Movies Features

Movies	Compression Ratio	Mean Bit Rate (bit/sec)	Peak Bit Rate (bit/sec)	Peak/Mean of Bit Rate
Aladdin	17.46	4.4e+05	3.1e+06	7.07
Jurassic Park	9.92	7.7e+05	3.3e+06	4.37
Silence of Lambs	13.22	5.8e+05	4.4e+06	7.73
The Simpson	5.84	1.3e+06	8.8e+06	6.75

We assume four different types of action performed by the user: pause, play, fast forward 2x and fast forward 4x. The action type and its starting time are randomly chosen by a uniform distribution. We suppose that all actions have a duration randomly chosen by a uniform distribution and ranging between 5 (125 frame-time) and 50 (1250 frame-time) seconds. Thirty simulations of the same video have have been performed, each time with a different random sequence of user actions, calculating losses and then averaging the results. The considered losses occur only for buffer overflow, because when underflow occurs frames are not lost, but the user only displays a jerky movie. Nevertheless, in this case the server can retransmit the frames to be still decoded. As explained above, a different sequence of user actions with different durations, is generated in each simulation. The first experiment shows SAIV and SLWIN performance with a fixed buffer and window sizes. The window size is 600 frames and the buffer size is 1MByte.

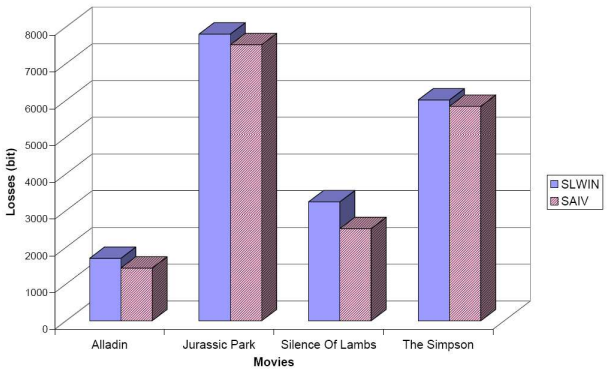

Fig. 4. Histogram of lost frames during the movies transmission

Fig.(4) shows the overall averaged overflow losses for each movie.

It is clearly visible that SLWIN losses are always higher on average than SAIV losses for all considered movies. This demonstrates the SAIV better capacity of reacting more efficiently to external user actions thanks to the dynamic variation of RTCP periodicity.

A. SAIV and SLWIN performance for different window sizes

In Fig. (5) another experiment is illustrated, that shows SAIV and SLWIN performance for different windows sizes. We selected two movies, "Silence Of the Lambs" and "Jurassic Park" both of length 20 minutes. We fixed the client buffer size at 1Mbyte. The window size N varies from 550 frame times (22 seconds) to 1000 frame times (40 seconds) with step 50 frame times (2 seconds). The markers in the figure represent the simulation results averaged over 30 experiments, as previously explained.

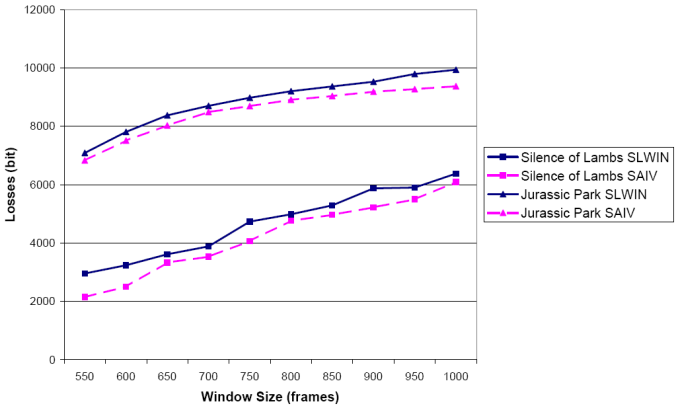

Fig. 5. Lost frames for SAIV and SLWIN with fixed window size

Fig. (5) shows the influence of the window size N on losses for SAIV and SLWIN. Based on the specific user action SAIV dynamically varies the instant of the next RTCP feedback and consequently of the slide-length α. Also in this case, for both movies SAIV performs better than the SLWIN, as shown in Fig. (5). For both SAIV and SLWIN, losses decrease with N decrease since an increased average feedback frequency reduces the loss probability, even if the computational complexity increases too.

B. SAIV and SLWIN performance for different buffer sizes

Another proposed experiment shows SAIV and SLWIN performance for different client buffer sizes. It is illustrated by Fig.(6) that shows the "Jurassic Park" losses and Fig.(7) that shows the "Silence of Lambs" losses.

Losses have been calculated by choosing the same pieces of video stream used in the previous simulation. Type, duration and starting time of the user actions have been randomly extracted by a uniform distribution as during the previous experiments. In this case the window size has been set to $N = 600$ frame times, and the buffer size varies from 64Kbytes to 1Mbytes, with increasing powers of 2.

Once again SAIV performs better than SLWIN. As Fig.(6) and Fig.(7) show, SLWIN losses are higher than the SAIV

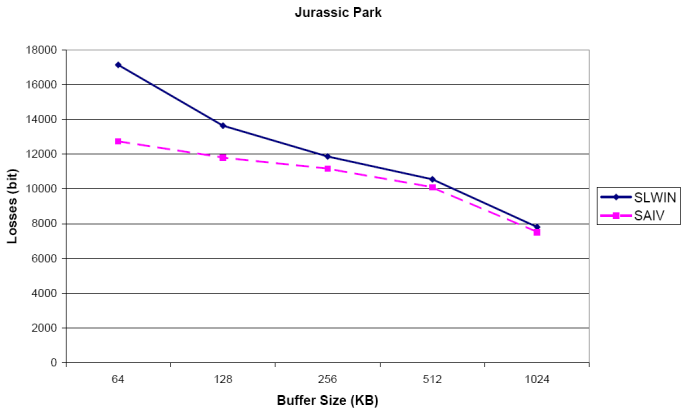

Fig. 6. ”Jurassic Park” lost frames for SAIV and SLWIN with fixed buffer size

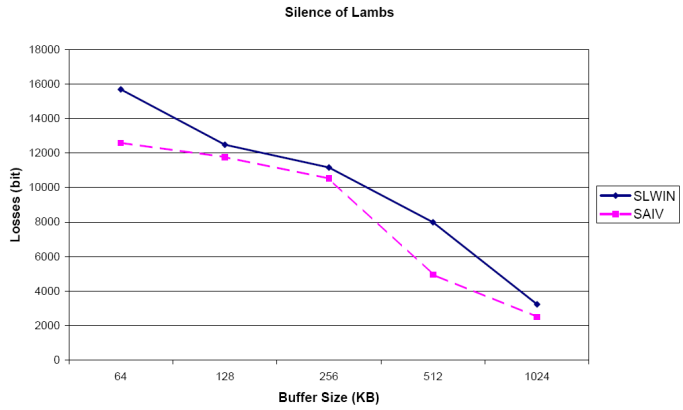

Fig. 7. ”Silence of Lambs” lost frames for SAIV and SLWIN with fixed buffer size

ones. In both cases losses decrease with the increase of the buffer size since larger client buffers reduce the buffer overflow probability. For smaller buffer sizes (64, 128 and 256 kB) losses are high with or without variable slide length. This happens because high quality videos are transmitted, with a relatively high number of bits per frame that are more problematically stored in relatively small client buffers. As a consequence the probability of buffer overflow increases on average. Nevertheless, SAIV losses decrease more quickly with the buffer increase.

V. Conclusions and Future Work

In this paper a new scheduling algorithm, the Scheduling Algorithm for Interactive Video (SAIV), has been proposed. It schedules the transmission of VBR video streams sent over UMTS network. The algorithm takes into account the interactivity of the end user to reduce losses at client side. The user actions influence the transmission plan; the frequency of the RTCP packets that the client sends to the server and the $slide - length$ are varied accordingly. The comparison with SLWIN, the most representative online scheduling algorithm known by literature, testifies the SAIV better performance in UMTS scenarios of practical interest. SAIV performance are influenced by the sequence of the user actions and especially by varying RTCP frequency calculated by the server. Further improvements, in this direction, can be done by testing other methods for the dynamic calculation of the RTCP sampling frequency. Other on-line scheduling algorithms can also be adopted, that can more quickly react to the varying client buffer conditions and further reduces frame losses. The real status of the UMTS core network, together with its intermediate buffers, could also be modeled to analyze the network behavior towards losses.

Acknowledgment

This work was funded by the Projects PS-121, PS-092, PS-025 (Apulia Region, Italy).

References

[1] H. Holma and A. Toskala, *WCDMA for UMTS : Radio Access for Third Generation Mobile Communications*, J. Wiley, Ed., 2004.

[2] I.Elsen, F.Hartung, U.Horne, M.Kampann, and L.Peters, “Streaming technology in 3G communication systems,” *IEEE Computer*, September 2001.

[3] 3GPP-TS23.107, *3GPP Technical Specification Group Services and System Aspects; Quality of Service (QoS) concept and architecture, (Release 9)*, December 2009.

[4] A.Lo, G.Heijenk, and I.Niemegeers, “Performance evaluation of MPEG-4 video streaming over UMTS networks using an integrated tool environment,” *Society for Modeling Simulation Int l*, 2005.

[5] 3GPP-TS26.234, *3GPP; Technical Specification Group Services and System Aspects; Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs,(Release 9)*, December 2009.

[6] M. Kampmann and C. Plum, “Stream switching for 3GPP PSS compliant adaptive wireless video streaming,” *IEEE Consumer Communications and Networking Conference (CCNC), vol.2, pp.954-958*, January 2006.

[7] H.Kaaranen, S. Naghian, L.Laitinen, A. Ahtianen, and V. Niemi, *UMTS Networks: Architecture, Mobility and Services*, J. Wiley and Sons, Eds., 2001.

[8] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A transport protocol for real-time applications,” Internet draft standard RFC 3550, July 2003.

[9] Ott, S. Wenger, N. Sato, C. Burmeister, and J.Rey, “Extended RTP profile for real-time transport control protocol (RTCP)-based feedback (RTP/AVPF),” Internet proposed standard RFC 4585, July 2004.

[10] S. Sen, J. Dey, J. Kurose, and D. Towsley, “Online smoothing of variable-bit-rate streaming video,” *IEEE Transactions on Multimedia*, vol. 2, no. 1, pp. 37–48, March 2000.

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

[12] J. Salehi, Z.-L. Zhang, J. Kurose, and D. Towsley, “Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing,” *IEEE/ACM Transactions On Networking*, vol. 6, no. 4, pp. 397–410, August 1998.

[13] P. Camarda, D. Striccoli, and R. Laraspata, “A bandwidth dependent window-based smoothing algorithm for wireless video streaming in UMTS networks,” *Packet Video Workshop (PV 2007) Lausanne, Switzerland*, November 2007.

[14] N. Baldo, U. Horn, M. Kampmann, and F. Hartung, “RTCP feedback based transmission rate control for 3G wireless multimedia streaming,” *15th IEEE International Symposium on Personal,Indoor and Mobile Radio Communications (PIMRC 2004), vol.3, pp.1817-1821*, September 2004.

[15] I. Johansson, M. Westerlund, and Ericsson, “Support for reduced-size real-time transport control protocol (RTCP): Opportunities and consequences,” Internet proposed standard RFC 5506, April 2009.