

RATE-DISTORTION OPTIMIZED DYNAMIC BITSTREAM SWITCHING FOR SCALABLE VIDEO STREAMING

Bo Xie

PacketVideo Corporation
San Diego, CA 92121
xie@packetvideo.com

Wenjun Zeng

Dept. of Computer Science
Univ. of Missouri-Columbia
zengw@missouri.edu

ABSTRACT

Bitstream switching is an effective way to deal with bandwidth variation in transmitting multimedia over time-varying channels such as the Internet or wireless networks. Most of previous work has focused on how to switch to reduce the potential error drift, without necessarily taking into account the impact of the switching on the bandwidth and delay requirements. This paper proposes a rate-distortion optimized dynamic bitstream switching algorithm that *explicitly* takes into account the impact of the switching on both drifting effect and bandwidth requirement. The rate control (or switching decision) of the proposed algorithm *explicitly* takes into account the delay constraint of multimedia applications, using an integrated end-to-end virtual network buffer management approach. Initial results show that it can achieve good performance with well-controlled quality drift and delay, and provides a promising research direction.

1 INTRODUCTION

Bit rate scalability is essential when transmitting video over time-varying channels such as the Internet or the wireless networks. For example, bandwidth can scale by a factor of 2 or more in a typical 2.5G or 3G wireless network. A bitstream that inherently has bit rate scalability is thus of importance. However, a scalable bitstream alone may not provide a large enough scalable bit rate range to address large bandwidth variation without sacrificing the coding efficiency [5]. In addition, current industrial standards, e.g., 3GPP, the Third Generation Partnership Project [4], may not necessarily support scalable codecs. As a result, dynamic bitstream switching among a set of bitstreams encoded at different bit rates may be the only way to deal with bandwidth variation in a standard compliant way. The goals of dynamic bitstream switching are to achieve high utilization of the available bandwidth, and to avoid player rebuffering due to congestion, to produce the best content quality and experience for the end users.

One way to do bitstream switching is to switch only at the I (Intra-coded) frames. This solution is simple, but either incurs long delay in the switching, or sacrifices the coding efficiency significantly, depending on how frequently I frames are encoded. Recently, S/SP/SI frames have been proposed to serve as special bridging frames for switching from one bitstream to another [1][2][3]. These specially designed bridging frames can reduce or completely eliminate the error drifting effect often observed in bitstream switching, but are typically very complex. They typically require generating two intermediate bridging streams between any two bitstreams, each requiring a re-encoding process. Therefore they may not be viable for real-time video communication. The storage overhead for the bridging streams is also of concern. In addition, the bit overhead for transmitting a

S/SP/SI frame can be as large as transmitting an I-frame [2], which may have some negative impact in the bitstream switching scenario where the bandwidth is of the major concern.

The situation becomes much more complicated in the scenario of real-time application-layer multicasting over heterogeneous networks, where multiple bitstreams with different bit rates are generated on the fly, and bitstream switching is required at the server for each individual client. In this case, generating S/SP/SI frames on the fly for each individual client becomes prohibitively costly for the server. Restricting to switching only at I frames has the aforementioned delay and coding efficiency problem. In [7], we proposed to switch from one bitstream to another at potentially any time instance, without generating any intermediate bitstreams. We proposed a source characteristics based fast bitstream switching algorithm that can achieve well-controlled quality drift.

One common problem with all above mentioned solutions is that they only focus on the drifting artifact potentially introduced by the bitstream switching algorithms, without necessarily considering when to switch and what the impact on the bandwidth and delay is. After all, the ultimate goal of dynamic bitstream switching is to address the bandwidth variation issue in streaming applications. A complete solution should address both dynamic rate control and dynamic bitstream switching, with the delay constraint of the multimedia applications in mind.

There have been several prior approaches for dynamic rate control. One is TCP window-based congestion/flow control for *non-real time* traffic. However, the drastic rate change in such algorithm makes it unsuitable for real-time multimedia applications. Several TCP-friendly equation based congestion control techniques [6] have also been introduced, which provide smoother sending rates, at the mean time compete fairly with TCP traffic. However, we feel that there are a number of issues that have not been sufficiently addressed, especially for some streaming applications where wireless networks are involved. First, for wireless streaming, loss is attributed to not only network congestion, but also random bit error in the radio access network. Using TCP rate control and TFRC that are mainly designed for network congestion may not address the problem appropriately. Second, real-time delay constraint for multimedia applications has not been explicitly considered in most of the prior solutions. An algorithm that competes fairly with TCP traffic may not satisfy the delay constraint of multimedia applications. Third, a solution purely based on the statistics observed at the client does not necessarily solve the entire problem. We believe a server based solution that incorporates feedback from the client while *explicitly* considering the delay constraint should do a better job.

In this paper, we propose a *delay-aware* rate-distortion optimized dynamic bitstream switching algorithm. By using an integrated

generic end-to-end virtual network buffer management approach that explicitly takes into account the delay constraint, the proposed solution addresses both dynamic rate control and dynamic bitstream switching in a systematic way. The server makes the dynamic bitstream switching decisions by *proactively* estimating the transmission delay, so as to avoid packet loss and player rebuffering. Simulation results will be shown to illustrate the potential of the proposed algorithm.

2 DELAY-AWARE RATE DISTORTION OPTIMIZED BITSTREAM SWITCHING

In this section, we present an integrated virtual network buffer management approach for dynamic bitstream switching.

A generic virtual network buffer model

Let us assume a *virtual* network buffer located between the server and the client that abstracts the potentially complex network topology, and accounts for the delay and loss of packets, as shown in Fig. 1. This virtual network buffer can be roughly treated as an aggregated buffer of the buffers located at the server and at intermediate network nodes. Denote $B^n(i)$ and $B^d(i)$, respectively, as the virtual network buffer and the decoder buffer occupancies at the time instant i (when the i th frame is streamed). Let $C(i)$ and $R(i)$, respectively, be the channel transmission rate (throughput) observed/estimated *at the receiver* and the source rate (which is assumed to be equivalent to the server's sending rate in this paper) at the time instant i . At any time instance i , data enters one side of the virtual network buffer at a rate of $R(i)$, and leaves the other side of the virtual network buffer at a rate of $C(i)$. Then

$$B^n(i) = \sum_{j=1}^i R(j) - \sum_{j=1}^i C(j) \quad (1)$$

$$B^d(i) = \begin{cases} \sum_{j=1}^i C(j) - \sum_{j=1}^{i-N} R(j) & \text{if } i \geq N \\ \sum_{j=1}^i C(j) & \text{otherwise} \end{cases} \quad (2)$$

where N is the number of frames that are accumulated in the decoder buffer before the decoder starts to decode and playback video. It accounts for the initial playback delay at the receiver, which typically is a few seconds for streaming applications.

Effectively, we can assume that all packet loss is due to two causes, i.e., the packets arrive at the decoder later than its presentation time or never arrive (i.e., have being dropped). The former case may result in decoder buffer underflow where decoder rebuffering is usually invoked. Therefore, an important consideration in bitstream switching is to make sure that the switching will not result in decoder buffer underflow.

Based on Eqs. (1) and (2), it turns out that the maximum allowable level of the virtual network buffer occupancy at the time instant i in order to avoid decoder buffer underflow at the time instant $i+N$ is

$$B^n(i)_{\max} = \sum_{j=i+1}^{i+N} C(j) \quad (3)$$

i.e., the condition for no decoder buffer underflow is

$$B^n(i) \leq \sum_{j=i+1}^{i+N} C(j) \quad (4)$$

Eq. (4) suggests that at any time instance i , the data in the virtual network buffer (i.e., data that has been sent by the server but has

not reached the client) should be depleted within the next N frame time intervals, which corresponds to the initial playback delay of the multimedia applications. For a channel with dynamic bandwidth such as wireless networks, the allowable maximum virtual network buffer fullness is also time-varying. For a constant bit rate channel, the allowable maximum virtual network buffer fullness reduces to a constant.

If, at the time instant i , $B^n(i) > B^n(i)_{\max}$, then the i th frame will not arrive at the receiver within the limit of the initial delay for continuous media playback, and will be assumed to be lost by the decoder. The decoder will display the previous decoded frame (the $(i-1)$ th frame). Otherwise, the i th frame is assumed to be arriving at the decoder on time for decoding and playback.

In practice, to ensure efficient network bandwidth usage, we should have

$$B^n(i) \geq 0 \quad (5)$$

Eqs. (4) and (5) together provide a generic virtual network buffer constraint for a streaming video application. It is interesting to note that some special cases of this generic virtual network buffer model have been used for *encoder* rate control for a constant bit rate channel or an ATM network [8].

Rate-distortion optimized dynamic bitstream switching

We propose to handle the dynamic bitstream switching problem from the virtual network buffer management point of view. For ease of presentation, let us assume there are only two candidate bitstreams to work with. The problem can be formulated as trying to find the optimal switching points which result in overall the least distortion, while respecting the end-to-end virtual network buffer constraints as specified in Eqs. (4) and (5).

For bitstream switching between two bitstreams with source rates $R_1(i)$ and $R_2(i)$ ($R_1(i) > R_2(i)$ on average) at time instant i , respectively, the problem can be formulated as follows. Assuming the channel bandwidth profile $C(j)$ is known before hand ($C(j)$ can be estimated as shown later) and that the server is currently streaming Bitstream 1. At the time instant i , the server can choose to stay with Bitstream 1, or switch to Bitstream 2. If it chooses to stay with Bitstream 1, then the i th frame may or may not be lost (i.e., arrive late) at the decoder, depending on the relationship between $B^n(i)$ and $B^n(i)_{\max}$. The distortion at the decoder would be

$$D(i) = \begin{cases} D_1(i), & \text{if } B^n(i) < B^n(i)_{\max} \\ D_1(i) + D_1^{drift}(i), & \text{otherwise} \end{cases} \quad (6)$$

where $D_1(i)$ is the distortion of the i th normal reconstructed frame in Bitstream 1, and $D_1^{drift}(i)$ is the extra distortion caused by drifting when the previous decoded frame is displayed instead.

If on the other hand, the server chooses to switch from Bitstream 1 to Bitstream 2, the distortion at the decoder would be

$$D(i) = \begin{cases} D_2(i) + D_2^{switch}(i), & \text{if } B^n(i) < B^n(i)_{\max} \\ D_2(i) + D_2^{switch}(i) + D_2^{drift}(i), & \text{otherwise} \end{cases} \quad (7)$$

where $D_2(i)$ is the distortion of the i th normal reconstructed frame in Bitstream 2, $D_2^{switch}(i)$ is the extra distortion caused by switching because the reference frames are different, $D_2^{drift}(i)$ is

the extra distortion caused by drifting when the previous decoded frame is displayed for the i th frame due to packet loss/being late.

For a potential switching point i , let $D^{W_i} = \sum_{j \in W_i} D(j)$ be the accumulated distortion over a local common window of time W_i , e.g., a Real-Time Control Protocol (RTCP) report period T_{RTCP} for all potential switching points in that period when real-time transport protocol (RTP) is used. The optimal switching point i^* is the one that minimizes the associated D^{W_i} , under the generic virtual network buffer constraints in Eqs. (4) and (5), i.e.,

$$i^* = \arg \min_i D^{W_i} = \arg \min_i \sum_{j \in W_i} D(j) \quad (8)$$

Typically, a high bit rate bitstream will be streamed as long as the decoder buffer will not be underflow, i.e. Eq. (4) is not violated. In the following, we will use the RTP streaming scenario to further illustrate how the proposed algorithm works. We use I to indicate the time instant a RTCP report is received.

In practice, we can define two virtual network buffer bounds with safety margins to make sure the decoder buffer will not be underflow and that the channel bandwidth is utilized efficiently (we assume the player has sufficiently large decoder buffer size so decoder buffer overflow is less of a problem), i.e.,

$$B_{upperbound} = C(I) * (T_{delay} - M_1) \quad (9)$$

$$B_{lowerbound} = C(I) * M_2$$

where T_{delay} is the initial/start-up delay, M_1 and M_2 are two non-negative safety margins, and $T_{delay} \geq M_1 + M_2$. $C(I)$ can be estimated as the effective throughput observed at the client, which will be conveyed to the server through the client's RTCP report. Without exact knowledge of the future network bandwidth, it is reasonable to assume that the average network bandwidth is $C(I)$ for the next RTCP report period T_{RTCP} . $C(I)$ is used as the channel rate in estimating the virtual network buffer occupancy at each frame instance in the next RTCP report period as well. Better results can be achieved if more sophisticated statistical model can be used to characterize the network bandwidth dynamics.

Upon receiving a RTCP report from the client, the server updates $C(I)$ and the virtual network buffer occupancy:

$$B^n(I) = B^n(I-1) + (B_{sent} - B_{received}) \quad (10)$$

where B_{sent} and $B_{received}$ are the number of bits sent by the server and received by the client, respectively, during the last RTCP report period. Note that $B_{received}$ includes not only packets received by the client at the transport layer, but also *acknowledged/detected* lost packets that the client never receives at the transport layer, so that any known lost packets will be accounted in the calculation of the virtual network buffer occupancy. Both the actual number of bits received by the client and the sequence numbers of the detected lost packets are conveyed to the server through the client's RTCP report. The server then performs the bitstream switching decision based on Eq. (8) subject to the virtual network buffer constraints of Eq. (9). Note the uplink delay is assumed to be zero here. For a more accurate estimation, uplink delay can be estimated and compensated as well.

To save computation, we can also define two virtual network buffer thresholds $Thred_high$ and $Thred_low$ where

$B_{lowerbound} < Thred_low < Thred_high < B_{upperbound}$. Then when the virtual network buffer occupancy estimated at a frame instance is within these two thresholds, no switching is considered. Once these two thresholds are reached, we start considering switching, including finding the best switching point that minimizes D^{W_i} subject to the buffer constraint of Eq. (9).

In practice, for non-real-time applications such as video on demand, the distortion $D(i)$ (Eqs. (6) and (7)) can be pre-calculated and stored with the bitstream (e.g., in the hint track of MPEG-4 file format) so that they are readily available for on-the-fly bitstream switching decision. For real-time application-layer multicasting over heterogeneous networks, where multiple bitstreams with different bitrates are generated on the fly and dynamic bitstream switching is required for individual clients, the fast intelligent bitstream switching algorithm proposed in [7] can be used to determine the switching point within a window where the delay constraint (Eq. (9)) is about to be violated.

3 EXPERIMENTAL RESULTS

We report some preliminary results here. In this example, we have two bitstreams with bit rates of 128 kbps and 28 kbps, and frame rates of 15 fps and 3 fps, respectively. M_1 and M_2 in Eq. (9) are set to zero in this example. We have an estimated bandwidth profile of the wireless network. In fact, the upper curve in Fig. 2 represents the upper-bounds of the virtual network buffer occupancy, which is defined as a scaled version of the estimated instantaneous network bandwidth (scaled by a constant, which is the initial delay, see Eq. (9)). So the upper curve in Fig. 2 essentially shows the network bandwidth fluctuation. At the beginning of the streaming, the bitstream with a bit rate of 128 kbps is chosen. This bit rate is larger than the instantaneous network bandwidth, so the estimated virtual network buffer occupancy gets accumulated, until the time instance of about 11s where the virtual network buffer occupancy upper bound is about to be reached. The proposed algorithm then chooses to switch to the other bitstream with lower bit rate at a point/frame that is R-D optimized. At about 15s, the network bandwidth increases. The algorithm then decides to switch back to the bitstream with higher rate to avoid virtual network buffer underflow (Eq. (5)).

Fig. 3 shows the *estimated* network delay where it is clear that, at about 11s, the delay bound of 7s is about to be reached where the algorithm decides to switch to a bitstream with lower bitrate so that the estimated delay reduces. The switching is more clearly illustrated in Fig. 4 where the PSNR curves of the two bitstreams involved and the actual streamed stream are shown with two switching points at about 11s and 15 s.

Fig. 4 also shows the PSNR curve of a heuristic client-based approach that switches immediately simply upon a request from the client based on the client buffer fullness, i.e., if the *client* buffer fullness reaches the buffer overflow threshold (set to 90% of the *client* buffer size in our experiment), then immediately switch to the bitstream with higher bit rate; if the *client* buffer fullness comes close to the buffer underflow threshold (set to 10% of the *client* buffer size in our experiment), then immediately switch to the bitstream with lower bit rate. Note that the client buffer size is fixed and chosen as $2 * T_{delay} * (\text{higher bitrate})$, and

the initial client buffer fullness is $T_{delay} * (\min(\text{initial_channel_rate}, \text{initial_bitstream_bitrate}))$. From Fig. 4, we can see this heuristic approach has very slow response to the bandwidth change and even no response in some cases. For example, it never switches back to the higher rate bitstream, although the bandwidth changes back to the higher rate at about 15s. Note that the slow response may potentially result in player rebuffering or inefficient use of the network bandwidth. The problem lies in the fact that switching occurs only when the client buffer is about to overflow or underflow, without a good understanding of the network bandwidth and potential delay. In addition, Figs. 4 and 5 show that the heuristic approach results in significant drifting artifact.

On the other hand, the proposed delay-bounded R-D based approach has a quick response to the bandwidth change. It can achieve a nice bitstream transition without introducing significant annoying visual artifact. It is effectively a *proactive* dynamic rate control scheme that intends to avoid network congestion and packet loss prior to its inception.

4 CONCLUSION

This paper proposes a delay-aware rate-distortion optimized dynamic bitstream switching algorithm that *explicitly* takes into account the impact of the switching on both drifting effect and bandwidth requirement. The switching decision of the proposed algorithm *explicitly* takes into account the delay constraint of multimedia applications, using an integrated end-to-end virtual network buffer management approach. Initial results show that it can achieve good performance with well-controlled quality drift and delay. We believe the proposed framework provides a promising research direction. Work currently under investigation includes considering bitstream switching among multiple bitstreams (including scalable bitstreams), and incorporating other switching mechanisms, e.g., S/SP/SI frames, into the proposed rate-distortion based framework. Streaming proxies, if available, can also be incorporated into the framework, and is expected to provide improved performance.

5 REFERENCES

- [1] N. Faerber and B. Girod, "Robust H.263 compatible video transmission for mobile access to video servers", *Proc. ICIP* 1997.
- [2] M. Karczewicz and R. Kurceren, "A proposal for SP-frames", document VCEG-L-27, ITU-T Video Coding Experts Group Meeting, Eibsee, Germany, 09-12 January 2001
- [3] R. Kurceren and M. Karczewicz, "Improved SP-frame encoding", document VCEG-M-73, ITU-T Video Coding Experts Group Meeting, Austin, TX, 02-04 April 2001
- [4] 3GPP TS 26.234, v 1.5.1 (2001-03), 3rd Generation Partnership Project, TSG-SA, "Transparent end-to-end packet switched streaming service, Protocols and codecs, Release 4"
- [5] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. CSVT*, vol. 11, pp. 301-317, March 2001.
- [6] S. Floyd, et. al, "Equation-based congestion control for unicast applications," *ACM SIGCOMM*, Sept. 2000.
- [7] B. Xie and W. Zeng, "Source characteristics based fast bitstream switching," in *IEEE Inter. Conf. Multimedia & Expo.*, July 2003.
- [8] C. Hsu, A. Ortega, and A. Reibman, "Joint selection of source and channel rate for VBR video transmission under ATM policing constraints," *IEEE Journal on Selective Areas in Communications*, vol. 15, no. 6, pp. 1016-1028, August 1997.

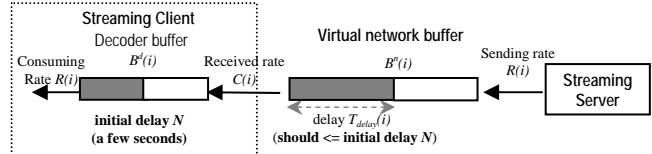


Fig. 1: A generic virtual network buffer model.

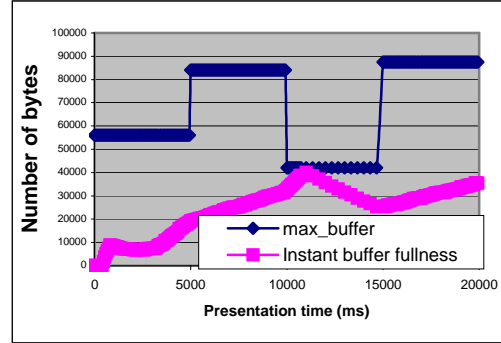


Fig. 2: Virtual network buffer fullness vs upperbound.

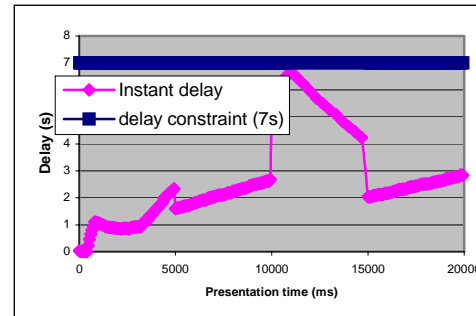


Fig. 3: Estimated network delay.

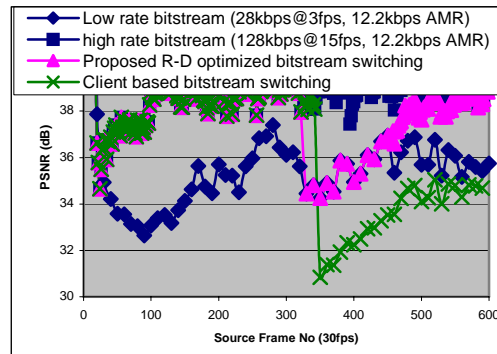


Fig. 4: PSNR curves for different dynamic bitstream switching schemes.



Fig. 5: Reconstructed frames at the same timestamp (at about 13s) after switching. Left: simple switching scheme based on client request; Right: proposed R-D optimized switching scheme.