A Novel Reverse Frame Selection Scheme for Video Streaming over VBR Channels

Dayong Tao Jianfei Cai
School of Computer Engineering
Nanyang Technological University, Singapore 639798
Email: {taod0001, asjfcai}@ntu.edu.sg

Abstract— **In this paper, we propose a Reverse Frame Selection (RFS) scheme based on dynamic programming to solve for the problem of video streaming over VBR channels. In particular, we first consider forward frame selection (FFS) for video streaming over CBR channels. We propose to discard non-optimal states in FFS to reduce the computational cost of dynamic programming. Then we find that the problem can also be solved by RFS with one additional benefit of finding all the optimal results for different preloads in one round. Furthermore, we extend RFS for video streaming over VBR channels where we do not know when and how the channel is going to change in the future. The major advantage of our proposed scheme is that we only need to run RFS several times, and the obtained results can be applied to any type of VBR channels with bandwidth changes occurring at any time. Preliminary results show the good performance of our proposed scheme.**

I. Introduction

Bandwidth smoothing techniques are commonly used to optimize the resources required for delivering variable bit-rate (VBR) encoded video without loss of data [1]–[3]. However, under both constrained bandwidth and limited buffer size condition, lossy smoothing has to be adopted. In [4], Ng and Song suggested to delete frames when the transmission exceeds the rate limit. In [5], Zhang *et al.* proposed a selective frame discard algorithm to minimize the number of frames that must be discarded in order to meet the bandwidth and buffer size constraints. In [6], Zhou and Liou proposed a nonlinear frame sampling strategy to maximize the delivery of the video's accumulated semantic scores. These approaches usually assume either constant bit-rate (CBR) channel or the channel behavior is known a priori. In practice, the time-varying channel rate, especially in the wireless communications, is hardly predictable. In [7], Feng and Liu proposed two solutions for video streaming over VBR channels: 1) pre-compute a smoothing plan and adaptively drop frames under bad channel conditions, and 2) run the smoothing algorithm online under the new bandwidth condition for the rest of the frames. The second approach requires online computation of smoothing plans which may introduce delays and the situation becomes even worse when there are many concurrent connections. In [8], Gan *et al.* proposed a more robust dual-plan bandwidth smoothing scheme for layer-encoded video streaming. Upon bandwidth renegotiation failure, the scheme adaptively discards the enhancement layer data to maintain the original frame rate.

Besides the problem for VBR channels, most existing lossy smoothing algorithms do not consider packet loss caused by network congestion or physical-layer bit corruption. Recently, we have seen extensive studies on rate-distortion optimized (RDO) video streaming over lossy channels [9]–[11]. The most representative work is the one in [9], where Chou *et al.* proposed a framework for streaming packetized media over a lossy packet network in a RDO way. The proposed framework is able to minimize the end-to-end distortion under a rate constraint by choosing the right packets to transmit at a given transmission opportunity. Although the framework is very comprehensive and theoretically sound, it requires an accurate network delay model, which is very difficult to obtain for a network such as the Internet. In addition, the proposed optimal packet scheduling in [9] is very complex, which might limit its implementation in practice.

In this paper, we assume the packet loss problem can be well handled by the error control techniques deployed in the transportation layer and the link layer. We only focus on optimal lossy smoothing based on the a priori motion information in the video. We adopt the Pixel Change Map (PCM) mechanism [12] to compute the amount of motion in each frame. Our goal is to select a set of frames out of the video that can maximize the accumulated motion values while being guaranteed transmittable and playable under both bandwidth and buffer size constraints. The reason for adopting the heuristic PCM mechanism is for simplicity. In fact, any frame or content classification scheme can be used in our proposed system. The PCM mechanism by no means is the only or the best way to measure content importance.

The rest of the paper is organized as follows. In section II, we analyze the problem of video delivery over CBR channels and propose to discard non-optimal states in the forward frame selection (FFS) scheme. In section III, we present our reverse frame selection (RFS) scheme, which has one additional benefit of finding all the optimal results for different preloads in one round. In section IV, RFS is further extended to solve the problem of video streaming over VBR channels. We only need to run RFS k times, where k is the number of channel bandwidth samples, and the obtained results can be applied to any type of VBR channels with bandwidth changes occurring at any time. In section V, we evaluate the performance of RFS under different VBR channel conditions. Finally, we summarize the paper in section VI.

II. Forward Frame Selection for CBR channels

Fig. 1 shows a commonly used discrete-time model at frame level for client buffer management. Each discrete-time point along the horizontal direction is identified by the moment when the frame is being fetched out for decoding, and each buffer fullness level at a frame is called a state indicated by an arrow endpoint. The length of each downward arrow represents the frame size. The slope of the slanted lines determines the amount of data that can be transmitted in one frame time-slot period and is given by $bandwidth/framerate$.

Let b_i^j denote the buffer fullness level at the j-th state at frame i. If state b_q^l is created by state b_p^k, or in other words, b_q^l is directly linked with b_p^k, then we have the following relation:

$$b_q^l = \min\{b_p^k + (q-p)\cdot(bandwidth/framerate),\ B\} - f_q, \quad (1)$$

where B is the buffer size and f_q is the size of frame q. Note that b_q^l becomes a full buffer state ($b_q^l = B - f_q$) if buffer overflow occurs during state transition from b_p^k to b_q^l. In this case, the server has to stay idle for some time or transmit at a reduced rate in order to accommodate b_q^l. From our experience, a full buffer state is rarely on the final optimal frame selection path unless it is at an I frame or the channel rate is very high. Let M_i^j denote the accumulated motion values at b_i^j. For state transition from b_p^k to b_q^l, $M_q^l = M_p^k + m_q$, where m_q is the motion value associated with frame q. If another state b_s^t also leads to state b_q^l, we resolve the collision with:

$$M_q^l = \max\{M_p^k,\ M_s^t\} + m_q \quad (2)$$

In addition, there is a preload at the initial stage just before playback starts. The time required to build up preload is called startup delay and is given by $preload/bandwidth$.

Theoretically, the number of possible states increase exponentially with frame number, which makes dynamic programming computationally prohibitive. However, in this research, we find that the number of states at each frame can be largely reduced by three factors:

1) buffer size because no state can fall outside the given buffer range;
2) inter-frame dependency because P and B frames need their references for proper decoding;
3) non-optimal state such as b_5^2 with $M_5^2 = 0.23$ at P_5 in fig. 1.

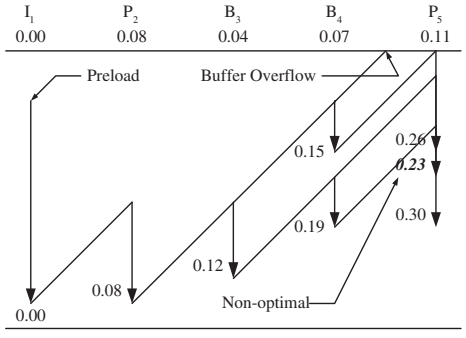

Fig. 1. The discrete-time model for client buffer management.

Lemma 1: For any two optimal states b_i^j and b_i^k at frame i (an optimal state means a state that could be included in the final optimal path), if $b_i^k < b_i^j$, then $M_i^k \geq M_i^j$, and vice versa. In other words, for optimal states, M_i^j increases monotonically as b_i^j decreases.

Due to space limitation, we do not present the proof in this paper. According to *Lemma 1*, we can conclude that for any two states b_i^j and b_i^k, if $b_i^j > b_i^k$ but $M_i^j > M_i^k$, then b_i^k is a non-optimal state and should be discarded. The elimination of non-optimal states can dramatically reduce the computational complexity because it not only reduces the number of states at each individual frame but also prevents those non-optimal states from propagating into subsequent frames.

III. Reverse Frame Selection for CBR Channels

In this section, we present our reverse frame selection (RFS) scheme to solve the problem of video streaming over CBR channels. Unlike the forward frame selection (FFS) scheme, our proposed RFS applies dynamic programming to the video sequence starting from the last frame until the first one. At the first frame, RFS generates all the optimal results for different preloads. Compared with FFS, which needs to run exhaustive searching for different preloads, RFS can find all the optimal paths in one round. In the following, we describe RFS in detail.

A. RFS in Normal Mode

As shown in fig. 2(a), each symbol above the full buffer line represents the frame type (I, P, or B) and the frame number in the video sequence. Frames are arranged from left to right in reverse order. Each real number below the empty buffer line represents the motion value associated with the frame. Obviously, after the last frame is consumed, the buffer should become empty. So the first state at the last frame B_N is created from the empty buffer line, which we refer it as an empty buffer state. The arrows are pointing up because we can only create a state and record its accumulated motion values when the arrow end is within the buffer. In other words, an upward arrow means consumption of data up to the length of the arrow in order to create a state. In contrast, reception of data is reflected by the downward slanted lines between frames. In case of "buffer underflow", such as that from B_N to P_{N-1}, an empty buffer state is created. It means that the amount of data transmitted during the period is more than enough for creating the current state, and the server needs to stay idle for some time or reduce the transmission rate. Because an path may terminate at any frame, if there is no empty buffer state at a frame, we will create one such as that at B_{N-2}.

B. RFS with Buffer Mirroring

Fig. 2(a) is not easy to interpret. We use a simple technique, which we call "buffer mirroring", to make the computation easier and more intuitive. Suppose a mirror is aligned with the empty buffer line in fig. 2(a). Looking from the full buffer line side, we shall see a mirrored buffer model as shown in fig. 2(b). Fig. 2(b) appears just like FFS except that the selection starts from the end of the video sequence until

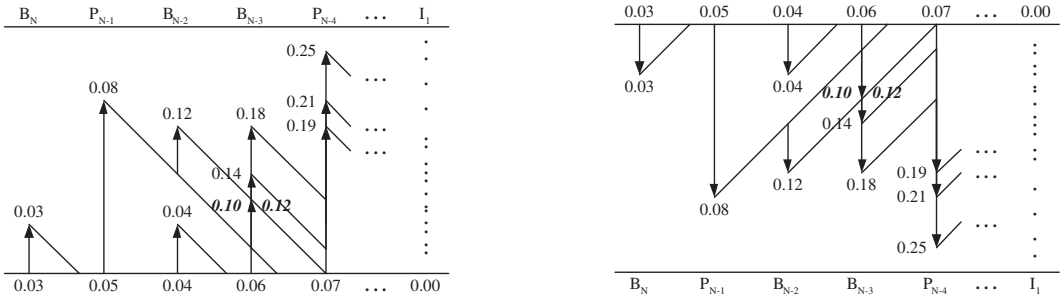

Fig. 2. The RFS scheme (a) in normal mode (b) with buffer mirroring.

the beginning. The computation procedures and the concepts, including *Lemma 1*, discussed for FFS can also be applied to fig. 2(b) with minor modification. Note that fig. 2(a) and fig. 2(b) are a pair of mirrors for each other. During path retrieving, the actual buffer occupancy status is obtained from fig. 2(a), which is the buffer mirror of fig. 2(b).

At the first frame I_1, RFS generates multiple states with different accumulated motion values. Each state corresponds to the optimal result that we can achieve at that preload. For those preloads that are not exactly matched with any of the states at I_1, they are the same as their nearby lower preloads. With RFS, we only need to run the scheme once and get all the optimal results for different preloads. It saves a lot of time to compute optimal paths for different start-up delay requirements. On the contrary, FFS has to run exhaustive searching for different preloads, which is very time-consuming.

IV. Reverse Frame Selection for VBR Channels

For VBR channels, the optimal frame selection becomes extremely difficult because we do not know when and how the channel is going to change in the future, i.e., we can not perform global optimization when the channel variation is unpredictable. Suppose we know the average bandwidth and the range of bandwidth variations. One possible approach (Approach 1) is to compute the optimal path according to the minimum channel bandwidth in order to completely avoid buffer underflow. We may also compute the path according to the maximum bandwidth, but it is likely to cause high occurrence of buffer underflows during transmission. Another possible approach (Approach 2) is to compute the optimal frame selection path according to the average bandwidth, and during transmission, the just-in-time (JIT) algorithm [5] is applied for on-the-fly adaptation to bandwidth changes. If the current bandwidth is larger than the average, JIT can gradually raise the path in buffer to reduce the probability of future buffer underflows. But the amount that the path can be raised is limited by the buffer size. During bad channel conditions when buffer underflow occurs, JIT adaptively drops frames from the path. Clearly, with these approaches, the number of actually transmitted frames can not be more than those on the pre-computed path.

In this paper, we propose to use the RFS scheme for video streaming over VBR channels. In particular, we sample the bandwidth variation range into a finite sequence of channel rates. For a given client buffer size, we run the RFS scheme for each sampled channel rate. During transmission, if the starting buffer state is b_1 and the current bandwidth is C, we first classify it into one of the pre-selected channel rates C_1 and then retrieve the optimal path for channel rate C_1 with starting buffer state b_1. Later at frame N_2, if the buffer state is b_2 and the bandwidth is changed to C_2, we can retrieve the optimal path starting at frame N_2 with buffer state b_2 for the new channel rate C_2. In this way, the global optimality is approximately preserved under dynamic changing bandwidth conditions. The key advantage here is that we only need to run RFS k times, where k is the number of bandwidth samples, and it can be applied to any type of VBR channels as long as the bandwidth changes are within the variation range.

V. Experimental Results

In this section, we describe two experiments to evaluate both the performance of different algorithms for video streaming over VBR channels and the effectiveness of non-optimal state elimination for reducing the computational cost of dynamic programming. We use two MPEG-4 video traces, *foreman* and *stefan* (see table I), in the experiments. Both videos are encoded in the pattern "PBBP...", with a GOP size of 90. The *foreman* video has moderate motion and is encoded at relatively low bit rate. The *stefan* video contains lots of high-motion frames and is encoded at a much higher bit rate.

A. Performance of Different Algorithms for VBR Channels

We compare the frame selection results of four algorithms: Approach 1 and 2 (see section IV), RFS, and UB. UB gives the upper bound assuming a priori known channel behavior, which is hardly attainable without an accurate channel model. To simulate VBR channel conditions, we divide each video into three 100-frame segments. Different segments are subject to different bandwidth conditions. For *foreman* video, the VBR channel rate sequence is R(f)={300, 210, 120} kbps with an average bandwidth of 210 kbps. For *stefan* video, R(s)={810, 600, 390} kbps with an average bandwidth of 600 kbps. In addition, R(f) and R(s) are each tested under two buffer

TABLE I

THE PROPERTIES OF TWO MPEG-4 VIDEO TRACES.

Video Title	Length (sec)	No. of Frames	Ave. Bitrate (kbps)	Total Motion Values
foreman	10	300	288.25	63.20
stefan	10	300	977.00	127.0

TABLE II

THE SELECTION RESULTS UNDER VBR CHANNEL CONDITIONS

Constraint Conditions	Algorithms			
	Approach 1	Approach 2	RFS	UB
R(f)+B(f1)	140 / 30.88	202 / 38.29	243 / 46.25	249 / 47.98
R(f)+B(f2)	156 / 37.47	222 / 49.51	266 / 54.60	270 / 55.42
R(s)+B(s1)	185 / 76.26	242 / 97.29	247 / 100.0	247 / 100.1
R(s)+B(s2)	209 / 83.84	255 / 104.5	256 / 104.5	255 / 104.6

conditions: {B(f1)=50, B(f2)=100} kbytes for *foreman* and {B(s1)=150, B(s2)=250} kbytes for *stefan*. For each buffer, the preload is set at half of the buffer size.

Table II shows the frame selection results using different algorithms under different constraint conditions. Each result is represented by two values: the integer value indicates the total number of selected frames and the decimal value indicates the accumulated motion values. From table II, Approach 1 has the worst performance under all conditions because it is too conservative. Approach 2 exhibits average performance under constrained buffer conditions and achieves better results with increased buffer sizes. A larger buffer allows Approach 2 to raise the frame selection path during good channel conditions and subsequently reduces the chance of dropping frames during bad channel conditions. In contrast, our proposed RFS is able to perform consistently well, with results comparable to those of UB, under all conditions especially for the constrained buffer conditions.

B. Effectiveness of Non-Optimal State Elimination

Fig. 3 shows the number of states at each frame with and without non-optimal state elimination. For RFS, we only implemented the scheme with non-optimal state elimination. From fig. 3, the two plots for FFS and RFS with non-optimal state elimination are more or less symmetrical around the middle frame. We set the parameters: $bandwidth = 600\ kbps$, $buffer = 150\ kbytes$, and $preload = 75\ kbytes$ (the preload parameter is used by FFS). As shown in fig. 3, the initial state increment is almost linear, which implies an exponential growth of states with frame number (note the use of a log scale in the vertical axis). The curves then become relatively flat because of the buffer size constraint. The discontinuous points are mainly due to the inter-frame dependency constraints. From the two FFS plots, we can see that the non-optimal state elimination can effectively reduce the number of states by approximately two magnitudes or one hundred times.

VI. SUMMARY

In this paper, we have studied the problem of optimal frame selection for video streaming over both CBR and VBR channels. Our contributions are threefold. First, we have proposed the elimination of non-optimal states, which can effectively reduce the computational cost of dynamic programming by several magnitudes. Second, we have proposed the RFS scheme for CBR channels, which can find all the optimal results for different preloads in one round. Third, we have also extended the RFS scheme for VBR channels, for which the experimental results have demonstrated that our proposed RFS scheme can achieve very good performance, close to global optimization, under dynamic bandwidth conditions.

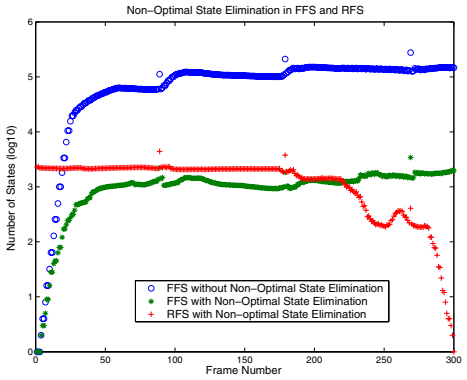

Fig. 3. Effectiveness of non-optimal state elimination for *stefan* video at bandwidth = 600 kbps and buffer = 150 kbytes.

REFERENCES

[1] J. D. Salehi, Z. L. Zhang, J. F. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," *IEEE Transactions on Networking*, vol. 6, no. 4, pp. 397–410, Aug. 1996.

[2] W. Feng and J. Rexford, "A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video," in *Proceedings of IEEE INFOCOM97*, Apr. 1997, pp. 58–66.

[3] W. Feng, "Rate-constrained bandwidth smoothing for the delivery of stored video," in *SPIE Multimedia Compting and Networking*, Feb. 1997, pp. 316–327.

[4] J. K.-Y. Ng and S. Song, "A video smoothing algorithm for transmitting MPEG video over limited bandwidth," in *Proceedings–Fourth International Workshop on Real-time Computing Systems and Applications*, Oct. 1997, pp. 229–236.

[5] Z.-L. Zhang, S. Nelakuditi, R. Aggarwal, and R. P. Tsang, "Efficient selective frame discard algorithms for stored video delivery across resource constrained networks," in *Proc. IEEE INFOCOM'99*, Mar. 1999, pp. 472–479.

[6] X. S. Zhou and S.-P. Liou, "Optimal nonlinear sampling for video streaming at low bit rates," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 535–544, Jun. 2002.

[7] W. chi Feng and M. Liu, "Extending critical bandwidth allocation techniques for stored video delivery across best-effort networks," *International Journal of Communication Systems*, vol. 14, pp. 925–940, Sep. 2001.

[8] T. Gan, K.-K. Ma, and L. Zhang, "Dual-plan bandwidth smoothing for layer-encoded video," *IEEE Transactions on Multimedia*, vol. 7, no. 2, Apr. 2005.

[9] P. A. Chou and Z. Miao, "Rate-distoration optimized streaming of packetized media," *Micorsoft Research Technical Report*, Feb. 2001.

[10] J. Chakareski and P. A. Chou, "Application layer error-correction coding for rate-distoration optimized streaming to wireless clients," *IEEE Trans. on Communications*, pp. 1675–1687, Oct. 2004.

[11] J. Chakareski, S. Han, and B. Girod, "Layered coding vs. multiple descriptios for video streaming over multiple paths," *Multimedia Systems*, pp. 275–285, Jan. 2005.

[12] H. Yi, D. Rajan, and L.-T. Chia, "Global motion compensated key frame extraction from compressed videos," in *ICASSP*, vol. 2, Mar. 2005, pp. 453–456.