ANALYSIS OF VIDEO CODEC BUFFER AND DELAY UNDER TIME-VARYING CHANNEL

Zhifeng Chen and Yuriy Reznik

InterDigital Communications, LLC
9710 Scranton Rd. Suite 250, San Diego, CA 92121 USA

ABSTRACT

In this paper, we analyze the effect of time-varying channels to video codec buffer specially for low-delay applications. We derive the sufficient conditions under which an encoder can design a bitstream for any time-varying channel without decoder buffer overflow and underflow. We then apply those conditions to design a bandwidth adaptive rate control in x264 and test it under LTE simulator. Our test results show significant improvement of delay and delay jitter over traditional leaky bucket models.

Index Terms— Leaky bucket model, HRD, VBV, Rate control, LTE, H.264/MPEG-4 AVC

1. INTRODUCTION

Thanks to the advances in wireless networks and improvements in processing and graphics capabilities of mobile devices, mobile video telephony is now becoming part of our daily lives [1]. Yet, some technical challenges in the design of video phone applications still exist. On one hand, such applications require low-delay, jitter-free delivery of video, but on the other hand, they are constrained by time-varying behavior of mobile networks.

In most video coding standards, a hypothetical reference decoder (HRD) [2] or a video buffering verifier (VBV) [3] is proposed to help design of conforming encoders and bitstreams. One major part of HRD/VBV is to ensure that there will be no underflow and overflow in the decoder coded picture buffer (CPB) where underflow causes delay jitter and overflow causes packet loss. In video encoders, leaky bucket models are usually adopted to control encoded bit rate conforming to the decoder HRD or VBV. Ref. [4] proposes a generalized HRD in H.264/AVC, where several leaky bucket models are specified for a given bit stream instead of just one option in previous standards. The best leaky bucket model can be selected by communication system according to different network connections and delay requirements.

New wireless network standard, e.g. 4G LTE, provides much faster transmission rate than previous wireless standards. Due to the multi-path fading and multi-user characteristics, the transmission bit rate fluctuates significantly, from zero to dozens of megabits per second, every transmission time interval (TTI). Rapid changes in channel behavior make the problem of low-delay rate control for video encoding very challenging. However, the generalized HRD assumes channel has constant bit rate (CBR) or piece-wise variable bit rate (VBR), i.e channel capacity changes much slower than frame rate. To the best of our knowledge, there is no literature strictly prove the conditions guiding encoder to design the bitstream conforming to HRD/VBV after transmitted over a fast time-varying channel.

In this paper, we analyze the effect of time-varying channels to buffer specially for low-delay applications. We derive, for the first time, the sufficient conditions under which an encoder can design a bitstream for any time-varying channel without buffer overflow and underflow. We then apply those conditions to design a bandwidth adaptive rate control in x264 and test it under LTE simulator. Our method requires an estimation of channel bit rate. Both perfect and imperfect channel estimation are simulated. The test results provides an reference about how much the improvement of delay/delay jitter and video quality over traditional CBR/VBR leaky bucket models we may gain if channel estimation is possible to encoder.

The rest of paper is organized as follows. Section 2 offers background information and illustration of problems why traditional leaky bucket models fail if channel bit rate rapidly changes. Section 3 proves the conditions for avoiding overflow and underflow in buffer and explains the bandwidth adaptive rate control method. Section 4 presents experimental results. Conclusions are drawn in Section 5.

2. PROBLEM STATEMENT

Fig. 1 show the relationship among a leaky bucket model and encoding schedule in a CBR case. In H.264/AVC HRD, the decoding process is assumed to be instantaneously, i.e. the decoding schedule is a step function (or staircase function). Correspondingly, in Fig. 1 we also assume the encoding process is instantaneous. Note that our method in this paper is general for non-instantaneous encoding and decoding processes. In Fig. 1, the encoder buffer fullness is defined as $F_e(t) = \mathcal{S}_e(t) - \mathcal{S}_t(t)$, which is constrained by B_e and the decoder buffer fullness is defined as $F_d(t) = \mathcal{S}_r(t) - \mathcal{S}_d(t)$, which is constrained by B_d.

The leaky bucket model actually specifies a combination of $\{\max[R(t)],\ B_e,\ F_e(t_e^0)\}$ for the encoder, where $R(t)$ is the channel capacity over time. It has been proved that the

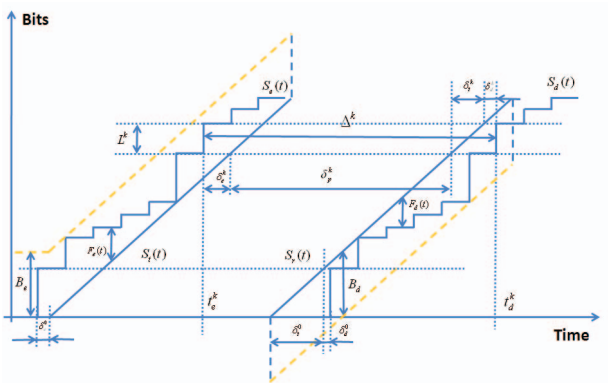

Fig. 1. Leaky bucket model in the CBR case. B_e and B_d denote buffer sizes of encoder and decoder correspondingly. $\mathcal{S}_e(t)$, $\mathcal{S}_t(t)$, $\mathcal{S}_r(t)$, $\mathcal{S}_d(t)$ show cumulative numbers of bits over time (or *schedules*) at the encoder, transmitter, receiver, and decoder correspondingly. L^k is the length of k-th encoded frame, which is pushed into encoder buffer at t_e^k and pulled out of decoder buffer at t_d^k. Quantities δ_e^k, δ_p^k, δ_t^k, δ_d^k are delays due to encoder buffering, propagation, transmission, and decoder buffering correspondingly. Δ^k is the end-to-end delay.

complement of encoder buffer fullness is equal to the decoder buffer fullness for the CBR cases [4]. Therefore, a bit stream designed to avoid overflow and underflow in encoder under the corresponding leaky bucket model will conform to the HRD in decoder. As a result, in video coding standards, HRD or VBV only specifies $\{\max[R(t)], B_d, F_d(t_d^0)$ (equivalent to $\delta_t^0 + \delta_d^0)\}$ for the decoder. In Ref. [4], authors also proved that the complement of encoder buffer fullness is a tight (achievable) upper bound to the decoder buffer fullness in piece-wise VBR case

$$F_d(t) \leq B_e - F_e(t). \tag{1}$$

However, as shown in Fig. 2, although the exactly same encoding schedule as in Fig. 1, (1) fails and both underflow and overflow happens due to time-varying channels.

Fig. 2 shows an example of decoder underflow caused by transmission of frame 1 (index begins from 0). This frame was regulated by the traditional leaky bucket model in order to meet HRD but can no longer be delivered on scheduled arrival time $\hat{t}_d^1$ due to very low instantaneous channel bit rate. So the decoder has to wait till point t_d^1 in order to start decoding of this frame. This introduces decoder jitter, which, if happens frequently, may be problematic for low-delay applications. Fig. 2 also shows possible decoder overflow and encoder underflow situations that can occur under time-varying channels. These examples explain motivation for our work, which is to derive the sufficient conditions for avoiding buffer underflow and overflow under rapidly-varying channels and based on that to design bandwidth adaptive rate control mechanism that can ensure HRD compliant bitstreams.

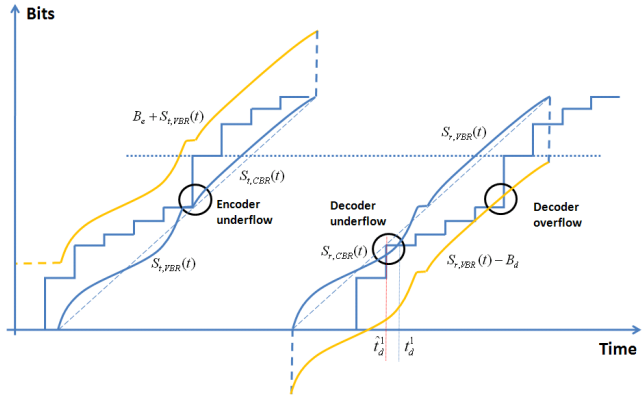

Fig. 2. Behavior of the leaky bucket model under time-varying channel. $\mathcal{S}_{t,VBR}(t)$ shows the actual transmission schedule, while $\mathcal{S}_{t,CBR}(t)$ corresponds to constant rate channel model, assumed by the encoder. Circled regions show encoder and decoder underflows caused by changing behavior of the channel.

3. ANALYSIS AND ALGORITHM

3.1. Basic constraints

From previous discussion, it follows that in order to avoid overflow and underflow in the encoder buffer, the encoding schedule $\mathcal{S}_e(t)$ shall satisfy

$$\mathcal{S}_t(t) < \mathcal{S}_e(t) < \mathcal{S}_t(t) + B_e, \tag{2}$$

where $\mathcal{S}_t(t)$ depends on the channel capacity and initial encoder delay δ_e^0 as shown in Fig. 1 and Fig. 2 . Similarly, in order to avoid overflow and underflow in the decoder buffer, the decoding schedule shall satisfy

$$\mathcal{S}_r(t) - B_d < \mathcal{S}_d(t) < \mathcal{S}_r(t). \tag{3}$$

where $\mathcal{S}_d(t)$ is a function of initial decoder delay δ_d^0. Therefore, δ_e^0 and δ_d^0 has critical impact on the HRD performance. We will see that in the experimental section. If data packets have constant propagation delay δ_p, the reception and transmission schedules become connected as follows

$$\mathcal{S}_r(t) = \mathcal{S}_t(t - \delta_p). \tag{4}$$

We next produce bounds for encoding and decoding schedules for time-varying channels with a continuous function $R(t)$, which is the most general case including CBR and piece-wise VBR.

3.2. Constraints under time-varying channels

By t_e^{k-} and t_e^{k+} let us denote time points right before and after adding the k-th frame bits into encoder buffer. It is easy to prove that a sufficient and necessary condition of (2) is

$$\mathcal{S}_e(t_e^{k-}) > \mathcal{S}_t(t_e^{k-}) \text{ and } \mathcal{S}_e(t_e^{k+}) < \mathcal{S}_t(t_e^{k+}) + B_e, \tag{5}$$

where $\mathcal{S}_e(t_e^{k-})$ is the cumulative number of encoded bits at time t_e^{k-} and $\mathcal{S}_t(t_e^{k-})$ is the cumulative number of transmitted

bits at time t_e^{k-}. Assume the time axis begins from encoding the first frame, that is $t_e^0 = 0$. From Fig. 1 and Fig. 2, we can observe that

$$\mathcal{S}_e(t_e^{k-}) = \mathcal{S}_e(t_e^{(k-1)+}) = \sum_{i=0}^{k-1} L^i, \tag{6}$$

and

$$\mathcal{S}_t(t_e^{k-}) = \mathcal{S}_t(t_e^k) = \mathcal{S}_t(t_e^{k+}) = \int_{\delta_e^0}^{t_e^k} R(t)dt. \tag{7}$$

Therefore, (5) becomes

$$S_t(t_e^{k+1}) < \sum_{i=0}^{k} L^i < S_t(t_e^k) + B_e, \tag{8}$$

that is,

$$\int_{\delta_e^0}^{t_e^{k+1}} R(t)dt < \sum_{i=0}^{k} L^i < \int_{\delta_e^0}^{t_e^k} R(t)dt + B_e, \tag{9}$$

for all $k \geqslant 0$.

Following the same arguments, it is easy to prove that a sufficient and necessary condition of (3) is

$$S_r(t_d^{k+1}) - B_d < \sum_{i=0}^{k} L^i < S_r(t_d^k), \tag{10}$$

that is,

$$\int_{t_d^0-\delta_t^0-\delta_d^0}^{t_d^{k+1}} R(t)dt - B_d < \sum_{i=0}^{k} L^i < \int_{t_d^0-\delta_t^0-\delta_d^0}^{t_d^k} R(t)dt. \tag{11}$$

Different from CBR and piece-wise VBR, (9) and (11) may be not satisfied simultaneously due to time-varying $R(t)$. However, if the transmitter has the capability of channel estimation for encoder, the estimated channel capacity $\hat{R}(t)$ can help design L^k in encoder to meet (9) and (11). In the following, we will 1) derive the sufficient conditions for L^k to avoid overflow and underflow in encoder and decoder; and 2) determine the tightest upper bound for B_e and B_d. In some cases, we also need to balance L^k and Δ^k.

3.3. Sufficient conditions for L^k and tightest upper bound for B_e and B_d

From (4), we can rewrite left side in (10) as

$$\begin{aligned} &S_r(t_d^{k+1}) - B_d = S_t(t_e^{k+1} + \Delta^{k+1} - \delta_p) - B_d \\ &= \textstyle\int_{\delta_e^0}^{t_e^{k+1}} R(t)dt + \int_{t_e^{k+1}}^{t_e^{k+1}+\Delta^{k+1}-\delta_p} R(t)dt - B_d. \end{aligned} \tag{12}$$

By comparing this expression with the left side of (9), we observe that the first inequality in (11) becomes satisfied, i.e. avoiding decoder overflow, if both (13) and the first inequality in (9) becomes satisfied,

$$\int_{t_e^{k+1}}^{t_e^{k+1}+\Delta^{k+1}-\delta_p} R(t)dt < B_d. \tag{13}$$

From (13), it is easy to prove that the tightest upper bound of B_d shall be $(\Delta_{\max} - \delta_p) \cdot R_{\max}$, where $\Delta_{max} = \max_k \Delta^k$ and $R_{\max} = \max_t R(t)$.

The first inequality in (9) can be rewritten as

$$L^k > \int_{\delta_e^0}^{t_e^k} R(t)dt - \sum_{i=0}^{k-1} L^i + \int_{t_e^k}^{t_e^{k+1}} \hat{R}(t)dt. \tag{14}$$

From the definition of buffer fullness, we see that $F_e(t_e^{k-}) = \sum_{i=0}^{k-1} L^i - \int_{\delta_e^0}^{t_e^k} R(t)dt$. Therefore, (14) is simplified as

$$L^k > \int_{t_e^k}^{t_e^{k+1}} \hat{R}(t)dt - F_e(t_e^{k-}). \tag{15}$$

In other words, by using $B_d = (\Delta_{\max} - \delta_p) \cdot R_{\max}$ and (15) for designing the encoding schedule, we can ensure the resulted bitstreams will not cause overflow in the decoder buffer. Note that (15) also ensures no underflow in the encoder buffer.

Consider now the right side in (10). We note that

$$\begin{aligned} &S_r(t_d^k) = S_t(t_e^k + \Delta^k - \delta_p) \\ &= \textstyle\int_{\delta_e^0}^{t_e^k} R(t)dt + \int_{t_e^k}^{t_e^k+\Delta^k-\delta_p} R(t)dt, \end{aligned} \tag{16}$$

By comparing this expression with the right side of (9), we know that B_e should satisfy

$$B_e \geqslant \int_{t_e^k}^{t_e^k+\Delta^k-\delta_p} R(t)dt, \tag{17}$$

which can be ensured by setting $B_e = B_d$.

By (16) and channel estimation, the second inequality in (10) can be rewritten as

$$L^k < \int_{t_e^k}^{t_e^k+\Delta^k-\delta_p} \hat{R}(t)dt - F_e(t_e^{k-}). \tag{18}$$

(18) define sufficient conditions for avoiding decoder buffer underflow. (18) and (17) further ensure the encoder has no overflow.

Note that (15) and (18) are general forms for any channel. The thresholds for CBR and piece-wise VBR cases can be easily obtained from them.

3.4. Discussion of delay-related constraints

From Fig. 1 and Fig. 2, we know that

$$\Delta^k = \delta_e^k + \delta_p + \delta_t^k + \delta_d^k. \tag{19}$$

In H.264/AVC, δ_e^0 is called *initial_cpb_removal_delay* and $\delta_t^0 + \delta_d^0$ is called *initial_cpb_removal_delay_offset*. Both of them was defined in Buffering period SEI (supplemental enhancement information) message [2].

From (15) and (18), it follows that end-to-end delay Δ^k must satisfy:

$$\Delta^k - \delta_p \geqslant t_e^{k+1} - t_e^k. \tag{20}$$

By (19) and (20), we know that to avoid overflow and underflow, the sum of *initial_cpb_removal_delay* and *initial_cpb_removal_delay_offset* in SEI should be designed at least larger than the frame period, in other words, $\delta_e^k + \delta_t^k + \delta_d^k \geqslant t_e^{k+1} - t_e^k$.

Another delay related constraint is to balance L^k and Δ^k. H.264/AVC HRD also specifies two delay modes, i.e. strict delay (delay jitter is not allowable) and low delay (delay jitter is allowable). Notice that in (19), the decoder buffering delay should be non-negative, i.e. $\delta_d^k \geq 0$ and the transmission delay δ_t^k depends on the k-th frame bits L^k. Therefore, the k-th frame bits L^k should be allocated such that:

$$\delta_t^k \leq \Delta^k - \delta_p - \delta_e^k. \tag{21}$$

In case of strict delay bound, that is $\Delta^k = \Delta$, where Δ is a given nominal delay, δ_t^k should be less than $\Delta - \delta_p - \delta_e^k$ In case delay jitter is allowable, it might make more sense to strike a certain balance in choices of constraints for Δ^k and L^k, which directly related to the delay/delay jitter and video quality/quality variation. The emphasis on choice of best rate points L^k may cause large delay jitter. On the other hand, the emphasis on Δ^k may cause large variation in bits allocated to different frames L^k and hence inconsistent video quality.

Note that for more general channel models where the propagation delay is time-varying, δ_d^k can be extended to include the difference between δ_p^k and δ_p^0. In this case, for strict delay mode, $\delta_t^k \leq \Delta - \delta_p^k - \delta_e^k$; for low delay mode, Δ^k should include the extra propagation delay of the k-th frame $\delta_p^k - \delta_p^0$.

3.5. Target frame bit estimation

Our final task is to define a rate assignment algorithm that meets conditions (15) and (18). Theoretically, there are infinite $\mathcal{S}_e(t)$ which meets (15) and (18). A simple but robust method is to design L^k to meet the average of the upper bound and lower bound. This method allow maximum margins for both high threshold and low threshold and therefore robust to the imperfect channel estimation. The major limitation of this method is the variation of quality frame by frame resulted from time-varying channel and strictly low end-to-end delay constraint. Another method is to design L^k to minimize the quality variation given (15) and (18). However, it requires a very accurate rate-distortion models to estimate L^k from targeted quality and some rate-distortion optimization are required [5, 6], which is beyond the scope of this paper. We will use the first method in this paper to show the performance of our bandwidth adaptive rate control algorithm.

Given the target end-to-end delay Δ^k, propagation delay δ_p, frame rate f_s, before encoding the k-th frame, encoder should design L^k based on its buffer fullness $F_e(t_e^{k-})$ and estimated channel capacity $\hat{R}(t)$ as follows

$$L^k = \frac{1}{2}\left[\int_{t_e^k}^{t_e^k + \frac{1}{f_s}} \hat{R}(t)dt + \int_{t_e^k}^{t_e^k + \Delta^k - \delta_p} \hat{R}(t)dt\right] - F_e(t_e^{k-1}). \tag{22}$$

3.6. Target residual bit estimation

For a hybrid video coder with block-based coding scheme, the encoded bit rate L^k consists of residual bits L_{resi}^k, motion information bits L_{mv}^k, and other header information bits L_{header}^k. That is,

$$L^k = H_{resi}^k \cdot N_{resolution} + L_{mv}^k + L_{header}^k, \tag{23}$$

where H_{resi}^k is the entropy of residual (bits per pixel) and $N_{resolution}$ is the normalized video resolution considering color components. Compared to H_{resi}^k, L_{mv}^k and L_{header}^k are less affected by quantization step size Q^k. Therefore, L_{mv}^k and L_{header}^k can be first estimated from the statistics in the previous frames and then H_{resi}^k will be used to determine the Q^k by any bit rate model.

4. EXPERIMENTAL RESULTS

4.1. Experimental Setup

To test our rate adaptation logic we have employed H.264/AVC codec, coupled with LTE channel simulator, implemented according to 3GPP TS 36.814 model. We have tested two encoders: x264 encoder with its original rate control[7], and a modified version of x264 encoder incorporating our proposed algorithm. For our rate control algorithm we used information from simulator to obtain channel rate estimates with different channel feedback delay. We've also set our rate control algorithm to meet several end-to-end-delay bounds, defined for each sequence and bit rate. The results are compared with both CBR and VBR in the original x264 encoder. Both VBV and HRD options are enabled. For x264 VBR rate control, we set initial delay and provided the mean and maximum channel rate. Decoding of all bitstreams was done by using standard JM decoder[8]. All tests were performed using standard CIF and 720p - resolution video sequences [9]. The first frame was encoded as an IDR frame and the following frames were encoded as P-frames with reference number equal to 3. Lookahead was disabled and the maximum size for all NAL packets was set to 1400 bytes.

4.2. Results

An illustrative subset of our results is shown in Fig. 3. For these cases, the delay bound (excluding propagation delay δ_p) is set to be very low, i.e. 90ms. LTE channel rate changes significantly every TTI, i.e. every 1ms, as shown in first column; the second column shows fluctuations of frame rates;

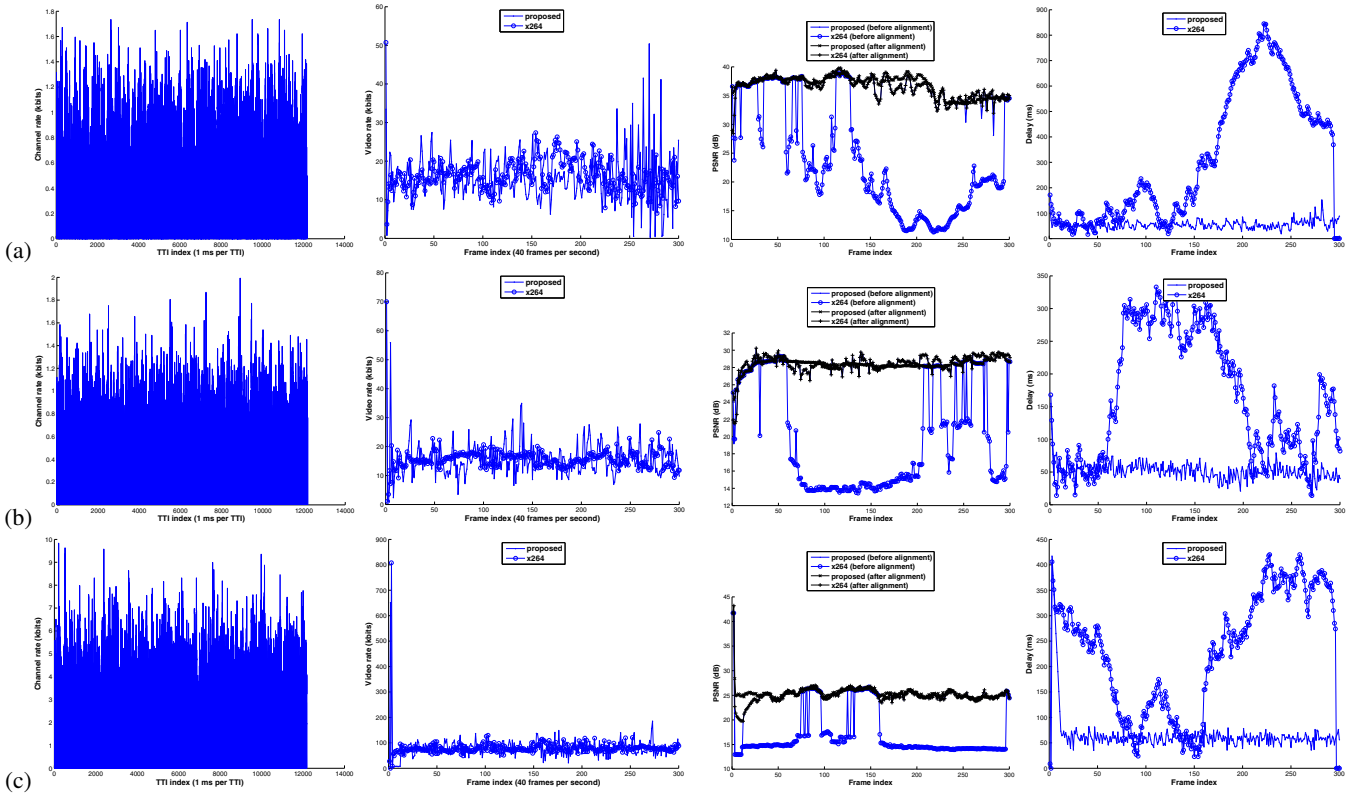

Fig. 3. Comparison of behavior of x264 VBR and our proposed rate control schemes under time-varying channels. Results are produced using the following video sequences: (a) Foreman, CIF, (b) Mobile, CIF, and (c) Parkrun, 720p.

peak signal-to-noise ratio (PSNR) are shown in third column, and last column shows end-to-end delay. It can be observed, that end-to-end delay with native x264 rate control varies significantly. For example, for the first sequence in Fig. 3 the delay grows up to 845ms. On the other hand, our proposed algorithm was able to maintain end-to-end delay close to the 90ms target.

In Fig. 3, PSNR is shown for both before and after aligning the decoded frames. x264 produces much worse PSNR and larger PSNR variation than our rate control algorithm at the rendering moment. After aligning the decoded frames, that is, using the reconstructed frames in encoder for calculation, x264 shows smaller PSNR variation than ours. This is because the significant delay and delay jitter in bitstreams produced by x264 and display has to use previous frames for rendering if frames were not received within the delay bound. In our algorithm, since almost all of frames can be received, decoded and rendered before the delay bound, although the resulted reconstructed frames in the encoder has a little larger PSNR variation, the resulted PSNR in the decoder is close to that in the encoder. Note that the PSNR variation in the encoder without considering the channel variation is the best of what can be achieved in the decoder with considering the channel variation. In other words, the best can be achieved when the delay bound is infinite. In fact, we believe either way of comparing PSNR is not good for low-delay applications. Instead, delay/delay jitter should be a better measure for the performance in this case. In our future work, we will analyze how to balance the delay and quality by using some metrics integrating both of them.

In Table 1 and 2, the delay-rate-PSNR, instead of traditional BD-PSNR/rate, are shown since the comparison is between two three-dimensional surfaces instead of two-dimensional curves. Delay is calculated by Δ^k with $\delta_p = 0$ in (22) and delay jitter is the variance of delay. Due to the space limit, only part of data set are shown in Table 1 and 2.

Under VBR case, bandwidth adaptive rate control saves up to 84.1% delay and 94.4% delay jitter than x264 rate control for foreman-cif sequence. Under CBR case, the former saves up to 70.5% delay and 87.5% delay jitter for the same sequence. This is because x264 VBR may allocate more bits to those scene change frames to meet HRD with maximum channel capacity. However, due to the instantaneous channel capacity is time-varying and may much smaller than the maximum channel capacity, those scene change frames will increase the delay/delay jitter. However, the gain reduces when the end-to-end delay bound increase. If the end-to-end delay bound is in seconds level (as in streaming video), the adaptive method hardly provides any performance gain.

We also compared the performance under imperfect chan-

Table 1. Performance comparison with x264 VBR, HRD/VBV enabled

sequence	init_delay (ms)	rate (kbps)	x264 PSNR (dB)	x264 delay (ms)	x264 jitter (ms)	proposed PSNR (dB)	proposed delay (ms)	proposed jitter (ms)	gain PSNR (dB)	gain delay	gain jitter
foreman-cif	90	400	17.27	341.82	268.95	35.93	54.39	15.10	18.66	84.1%	94.4%
foreman-cif	180	400	18.02	352.81	276.71	36.22	98.39	19.87	18.20	72.1%	92.8%
mobile-cif	90	400	17.90	137.95	70.37	28.17	50.97	12.95	10.27	63.1%	81.6%
mobile-cif	180	400	21.86	139.42	70.40	28.47	94.44	16.64	6.61	32.3%	76.4%
parkrun-720p	90	2000	15.28	213.83	86.41	23.21	66.94	47.92	7.93	68.7%	44.5%
parkrun-720p	180	2000	16.24	240.01	87.88	23.16	114.27	60.14	6.92	52.4%	31.6%
mobcal-720p	90	1000	24.42	84.61	48.18	28.55	46.07	43.51	4.12	45.6%	9.7%
mobcal-720p	180	1000	28.96	86.45	52.13	28.85	73.28	55.73	-0.11	15.2%	-6.9%

Table 2. Performance comparison with x264 CBR, HRD/VBV enabled

sequence	init_delay (ms)	rate (kbps)	x264 PSNR (dB)	x264 delay (ms)	x264 jitter (ms)	proposed PSNR (dB)	proposed delay (ms)	proposed jitter (ms)	gain PSNR (dB)	gain delay	gain jitter
foreman-cif	90	400	19.49	183.38	118.09	35.90	54.14	14.78	16.41	70.5%	87.5%
foreman-cif	180	400	21.28	172.86	122.51	36.21	98.32	20.04	14.93	43.1%	83.6%
mobile-cif	90	400	18.41	123.34	60.11	28.19	50.75	11.14	9.78	58.8%	81.5%
mobile-cif	180	400	23.97	110.94	59.88	28.42	94.24	16.00	4.46	15.1%	73.3%
parkrun-720p	90	2000	17.10	127.86	65.19	23.87	61.59	24.03	6.77	51.8%	63.1%
parkrun-720p	180	2000	19.43	142.19	78.66	24.13	105.79	25.70	4.70	25.6%	67.3%
mobcal-720p	90	1000	21.85	115.99	56.29	28.89	44.52	36.92	7.04	61.6%	34.4%
mobcal-720p	180	1000	28.03	96.90	52.96	29.15	71.60	49.54	1.12	26.1%	6.4%

nel estimation where the channel feedback has certain delay, ranging from miliseconds to hundreds of miliseconds. Due to the space limit, we did not show such results here. Instead, we give our observation below. Imperfect channel estimation has more impact on very low end-to-end delay case, e.g. 90ms, than on other end-to-end delay cases, e.g. 180/270/360ms. This is because the imperfect channel estimation causes deviation between estimated and true of encoder buffer delay and transmission delay. Since higher target end-to-end delay has higher tolerance to these deviations, there is little impact. Even under imperfect channel estimation, our rate control still shows remarkable gain over both x264 VBR and CBR rate control.

5. CONCLUSIONS

We derived the sufficient conditions under which an encoder can produce a bitstream for any time-varying channel without decoder buffer overflow and underflow. We then applied those conditions to design a bandwidth adaptive rate control with low delay constraint. The algorithm facilitates the design of encoding schedule which produces bitstreams conform to HRD model even transmitted over a time-varying channel. The algorithm was implemented within the H.264/AVC video encoder and tested using LTE simulator. Test results suggest that it achieves very good performance and tight control over end-to-end delay, necessary for real-time applications. Our encoding schedule and rate control is applicable to future video coding standards, such as High Efficiency Video Coding (HEVC).

6. REFERENCES

[1] T. Wiegand and G.J. Sullivan, "The picturephone is here. really," *IEEE Spectrum*, vol. 48, no. 9, pp. 50–54, September 2011.

[2] "ISO/IEC $14496-10$ | ITU-T Rec. H.264, Advanced Video Coding for Generic Audiovisual Services," Nov. 2007.

[3] "ISO/IEC $138180-2$, Information Technology – Generic Coding of Moving Pictures and Associated Audio Information: Video, Annex C, Video Buffering Verifier," 2000.

[4] J. Ribas-Corbera, P.A. Chou, and S.L. Regunathan, "A generalized hypothetical reference decoder for h. 264/AVC," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 674–687, 2003.

[5] Zhifeng Chen and Dapeng Wu, "Rate-Distortion Optimized Cross-Layer Rate Control in Wireless Video Communication," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 3, pp. 352–365, 2012.

[6] Z. He and S.K. Mitra, "Optimum bit allocation and accurate rate control for video coding via ρ-domain source modeling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 10, pp. 840–849, 2002.

[7] "x264 open source video encoder implemenation," http://www.videolan.org/developers/x264.html.

[8] "H.264/AVC reference software [JM 16.0]," http://iphome.hhi.de/suehring/tml/download.

[9] "Collection of test video sequences in YUV format," http://media.xiph.org/video/derf.