# GEOMETRICALLY DETERMINING LEAKY BUCKET PARAMETERS FOR VIDEO STREAMING OVER CONSTANT BIT-RATE CHANNELS

*Ping Li , W.S. Lin, S. Rahardja, X. Lin, X.K. Yang, Z.G. Li*

Media Division, Institute for Infocomm Resarch, 21 Heng Mui Keng Terrace, Singapore 119613
Email: imliping@hotmail.com, {wslin, rsusanto, linxiao, xkyang, ezgli}@i2r.a-star.edu.sg

## ABSTRACT

For streaming of pre-encoded bitstreams over constant bit rate (CBR) channels, the channel bandwidth, the receiver buffer capacity as well as the latency requirement vary greatly from application to application. In this paper, we propose an algorithm to determine the minimum buffer size and the minimum start-up delay required for streaming a pre-encoded bitstream over CBR channels at any specific bit rate. The proposed method employs geometric operations to derive the optimal determination for low or high bit rates and sub-optimal determination for medium bit rates. The results have been compared with the H.264/AVC hypothetical reference decoder. The proposed approach provides a theoretical insight and a simple but effective algorithm for determining the leaky bucket parameters for video streaming over CBR channels.

## 1. INTRODUCTION

One of the goals of video streaming is to start playback at the decoder with a limited buffer capacity as fast as possible while the video stream is being downloaded at a specific bit rate. For different video streaming applications, the bandwidths of the transmission channels, the buffer capacities of the decoders as well as the latency requirements may vary greatly. Determining the minimum buffer size and the minimum start-up delay required for the streaming a pre-encoded bitstream over CBR channels is necessary. The Video Buffering Verifier (VBV) [1, 2] in MPEG and the Hypothetical Reference Decoder (HRD) [3, 4] in H.264 are proposed to address the above problem. A VBV/HRD-compliant bitstream guarantees that the bitstream can be delivered to the terminal with a given buffer capacity using a given transmission bit rate and then decoded using a given start-up delay. A closely related concept with VBV/HRD is the leaky bucket model [5, 6], which is a direct metaphor of the encoder output buffer and can be characterized by ($R$, $B$, $\Gamma$), where $B$ is the encoder buffer size, $R$ is the transmission bit rate, $\Gamma$ means the transmission of the bits in the encoder buffer starts $\Gamma$ seconds after the bits for the first frame enter the buffer.

An encoder usually creates a bitstream according to a specific leaky bucket. For streaming of an bitstream created using leaky bucket ($R$, $B$, $\Gamma$), if the channel bandwidth, the buffer capacity and the initial delay for the specific video application happen to be equal to those specified by the given leaky bucket, then there will be not any problem in delivering and decoding the bitstream. However, if any of the three parameters differs from that specified by the leaky bucket, problem will occur if we do not adjust the other two parameters accordingly. In [5], a Generalized Hypothetical Reference Decoder (GHRD) for H.264/AVC is presented to solve this problem. The idea of GHRD is as follows: Given a number of leaky buckets that are known to contain the bitstream, the decoder can determine which leaky bucket to use knowing the available bit rate. If the available bit rate is already specified by one of the leaky buckets, then that leaky bucket is used directly. Otherwise, a linearly interpolated or extrapolated leaky bucket that safely contains the bitstream is used. In this paper, we attempt to geometrically determine the minimum buffer size and the minimum start-up delay required for streaming a pre-encoded bitstream over CBR channels at any bit rates.

The rest of this paper is organized as follows. In Section 2, the leaky bucket model is introduced and the connection between the encoder buffer and decoder buffer in a video streaming system is examined. In Section 3, we describe our algorithm that geometrically determines minimum buffer size and minimum start-up delay for a fixed bit rate. In Section 4, we implement the proposed algorithm in a H.264/AVC video encoder and evaluate its performance. Section 5 concludes this paper.

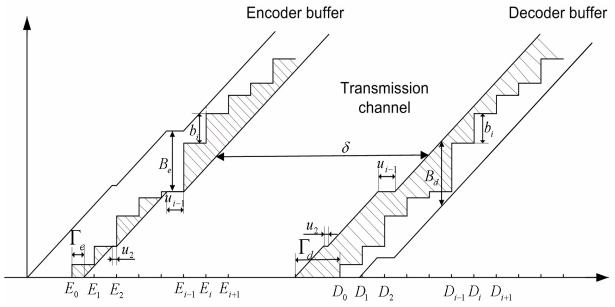


**Fig. 1**. Virtual encoding, transmission and decoding processes in a video streaming system

## 2. LEAKY BUCKET MODEL

A video streaming system comprises of a virtual encoder, a virtual transmission channel and a virtual decoder. As shown in Fig. 1, the virtual encoder reads the bits from the given pre-encoded bitstream and pushes bits $b_0$ for frame $f_0$ into the encoder buffer instantaneously at time instant $E_0$. After an initial encoder buffer delay $\Gamma_e$, bits in the encoder buffer are drained to a CBR channel until the buffer becomes empty. Subsequently, bits $b_i$ for frame $f_i$ are instantaneously pushed into the encoder buffer at instant $E_i$. At the decoder side, bits received from the channel are pushed into the decoder buffer at a constant bit rate except when there is no bits transmitted from the channel. After an initial decoder buffer

delay $\Gamma_d$ after the first bit enter the decoder buffer, bits $b_0$ are then instantaneously removed from the buffer at time instant $D_0$ for reconstructing frame $f_0$ . Subsequently, bits $b_i$ are instantaneously removed from the decoder buffer at time instant $D_i$ to reconstruct frame $f_i$. At the same time, bits from the channel are continuously pushed into the buffer at a constant bit rate except when no bits are received from the channel.

The virtual encoding, transmission, and decoding processes are graphically illustrated in Fig. 1, where $B_e$ is the encoder buffer size; $B_d$ is the decoder buffer size; $u_{i-1}$ denotes the buffer underflow between time instants $E_{i-2}$ and $E_{i-1}$, during which encoder buffer is empty and the transmission process is paused; $\delta$ is a constant end-to-end delay from the encoder to the decoder; $R$, $b_i$, $E_i$, $D_i$ are as described before. Obviously, to guarantee the decodability of the transmitted bitstream, we must make sure the decoder buffer is neither underflowed nor overflowed so that the decoder can get $b_i$ at time instant $D_i$ to reconstruct frame $f_i$.

With above notations, we can easily prove the following observation: *if there is no overflow and underflow in the encoder buffer, the decoder buffer will neither overflow nor underflow if the decoder buffer size is equal to the encoder buffer size, i.e., $B_e = B_d = B$, and if the sum of the initial encoder and decoder buffer delays is set to $B/R$, i.e., $\Gamma_e + \Gamma_d = B/R$, and if the encoding schedule is the same as the decoding schedule.* Due to space stringency, we do not include the proof in this paper.

Based on above observation, we can say if an bitstream is contained by a leaky bucket $(R, B_e, \Gamma_e)$, certainly the bitstream can be successfully delivered to the decoder with a buffer size $B_e$ using the transmission bit rate $R$ and then decoded using the start-up delay $\Gamma_d = B_e/R - \Gamma_e$ at the same frame rate as that of virtual encoder. Thus, once we can find out the lowest or highest bounds of $B_e$ and $\Gamma_e$ for the encoder buffer, the minimum resources required to transmit and decode the bitstream when bit rate is fixed can be easily determined. In the following discussion, our focus will be on the virtual encoder side and we use $(R, B_e, \Gamma_d)$ to characterize the leaky bucket for the encoder buffer since $\Gamma_e$ and $\Gamma_d$ are compliment of each other.

## 3. DETERMINE MINIMUM BUFFER SIZE AND MINIMUM START-UP DELAY

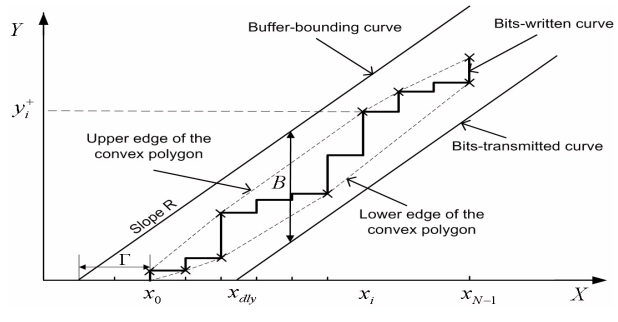


**Fig. 2**. Virtual encoding process and some important notations used in this paper

The virtual encoding process using leaky bucket $(R, B, \Gamma)$ at a fixed frame rate $F$ is graphically illustrated in Fig. 2, where $N$ is the number of the video frames in the video sequence. In the figure, the step-wise curve in the figure denotes the amount of bits written into the encoder buffer and is referred to as *bits-written* curve; the lowest curve (strait when no underflow) denotes the amount of bits transmitted over the channel and is referred to as the *bits-transmitted* curve; the highest curve (strait when no underflow) denotes the buffer bounding on the bitstream and is referred to as *buffer-bounding* curve. Clearly, *buffer-bounding* curve must be parallel to *bits-transmitted* curve and *bits-written* curve must be below *buffer-bounding* curve and above *bits-transmitted* curve.

With the above notations, our problem to determine minimum buffer size and minimum start-up delay for fixed bit rate is equivalent to finding a pair of parallel curves ( *bits-transmitted* curve and *buffer-bounding* curve) that tightly bounds the *bits-written* curve.

The vertical coordinates of the all the feature points denoted using "$\times$" in Fig. 2 can be obtained in the actual encoding process. The horizontal coordinates of the feature point at time instant $x_i$ equals to $x_0 + i/F$, where $x_0$ can be set to 0 since it does not affect our geometric analysis. All the vertexes of the convex polygon, which is shown in the dotted line in the figure, are selected as the feature points.

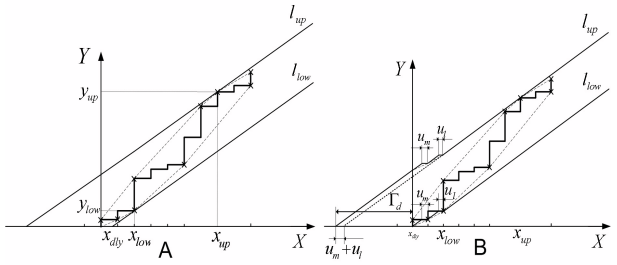


**Fig. 3**. Determination of minimum buffer size and minimum start-up delay when transmission bit rate is small and $x_{low} \leq x_{up}$

Fig. 3(A) shows virtual encoding process when no buffer underflow occurs, in which the *buffer-bounding* curve and *bits-written* curve are two straight lines and are denoted as $l_{up}$ and $l_{low}$ respectively. As we see, $l_{up}$ crosses the feature point $(x_{up}, y_{up})$ and $l_{low}$ crosses the feature point $(x_{low}, y_{low})$. The minimum buffer size $B_{min}$ and minimum start-up delay $\Gamma_{min}$ can be easily computed as follows:

$$B_{min} = (y_{up} - y_{low}) - (x_{up} - x_{low})R \quad (1)$$

$$\Gamma_{min} = B_{min}/R - (x_{low} - y_{low}/R) \quad (2)$$

Fig. 3(B) shows the virtual encoding process when the encoder buffer start-up delay $(x_{dly} - x_0)$ is reduced while the bit rate $R$ remains unchanged. In this case, two buffer underflows $u_m$ and $u_l$ occur before time instant $x_{low}$. From the figure, we observe that the buffer size and start-up delay computed by Equations 1 and 2 are optimal as long as $x_{low} \leq x_{up}$ since any underflow at this bit rate will result in an increase of $\Gamma$ although $B$ remains unchanged.

Fig. 3 only shows the case when $x_{low}$ is smaller than $x_{up}$, which usually happens when $R$ is small. If $R$ is increased further, $x_{low}$ may become greater than $x_{up}$ and the above derivations may not be true. In the following discussion, $R_1$ denotes the bit rate that separates the $x_{low} \leq x_{up}$ from $x_{low} > x_{up}$. $R_1$ can be easily computed and at most cases is approximate to the observed average bit rate of encoded bitstream. Fig. 4(A) shows the case when

$x_{low}$ is greater than $x_{up}$ and when no encoder underflow occurs. At this case, we can also compute a buffer size using Equation 1. However, the obtained buffer size is not optimal for the given bit rate $R$.

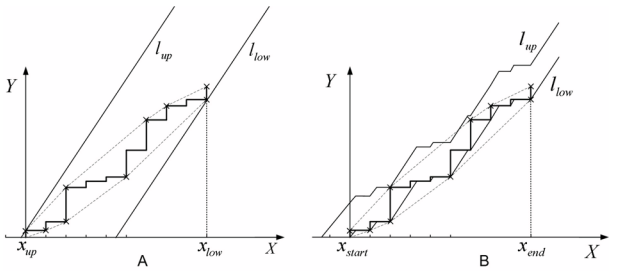


**Fig. 4**. The situation when transmission bit rate is high and $x_{low}$ is greater than $x_{up}$

As illustrated in Fig. 4(B), because of the small initial encoder buffer delay and the large transmission bit rate, the encoder buffer often becomes empty and results in the frequent pausing of the transmission process. Clearly, the *bits-transmitted* curve is now a complex piece-wise curve that corresponds to the transfer-pause-transfer-style transmission process. Thus, to determine the minimum buffer size, we need to find two parallel piece-wise curves that tightly bounds the *bits-written* curve. In that case, the tangent points may no longer be the vertexes of the convex polygon whose coordinates are known to us and it is difficult to determine the optimal minimum buffer size.

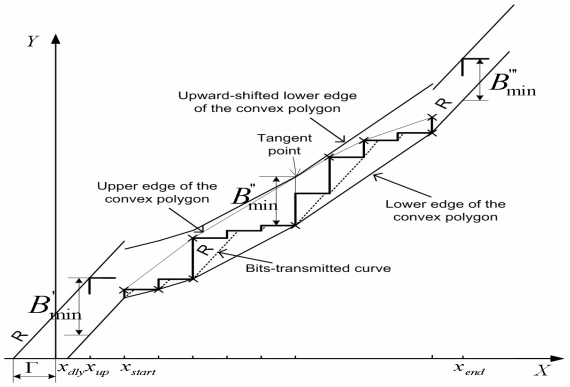


**Fig. 5**. Determination of minimum buffer size and minimum start-up delay under a fixed bit rate when $x_{low}$ is greater than $x_{up}$

The determination of minimum buffer size and minimum start-up delay when bit rate is high and $x_{low} > x_{up}$ is illustrated in Fig. 5, where $x_{start}$ denotes the time instant when underflow starts to occur and $x_{end}$ denotes the time instant when underflow ends to occur. Obviously, $x_{end}$ equals $x_{low}$. As shown in Fig. 5, when we attempt to determine the local minimum buffer size $B''_{min}$ that contains the bitstream segment from the time instant $x_{start}$ to $x_{end}$, it is safe to use the lower edge and upper edge of the convex polygon to denote the *bits-transmitted* and *bits-written* curves. In this way, $B''_{min}$ can be computed as the max vertical distance between the two edges of the convex polygon from $x_{start}$ to $x_{end}$. The other two local minimum buffer sizes $B'_{min}$ and $B'''_{min}$ that respectively contain the bitstream segments before $x_{start}$ and after $x_{end}$ can be easily computed using Equation 1. After we obtain the three local minimum buffer sizes, the largest is selected as the minimum buffer size $B_{min}$ that contains the entire bitstream.

Obviously, if the transmission bit rate is higher than $R_2 = b_{max} \times F$, where $b_{max}$ is the max frame size, then the buffer underflow will occur in every frame interval and thus $B''_{min}$ can be directly set to $b_{max}$. Clearly, an optimal buffer size is obtained at this case.

From Fig. 5, we see if the convex polygon can not tightly bound the *bits-written* curve, the max vertical distance $B''_{min}$ may be very large and thus the performance of our algorithm when $R_1 < R < R_2$ may not be good. In this case, it may be better if we linearly interpolate [1] a buffer size based on the buffer sizes at $R_1$ and $R_2$, which are optimal. Or, we can even use the GHRD to derive the minimum buffer size.

## 4. RESULTS AND DISCUSSIONS

In this section, our algorithm is implemented in a H.264/AVC reference video encoder JM6.1e [7] to evaluate its performance. Three video sequences *news.qcif*, *forman.qcif* and *container.qcif* are tested. For all three sequences, the first 300 frames are encoded at a frame rate of 30 frames per second. Only the first frame is encoded as "I" frame and all the rest 299 frames are encoded as "P" frames. The other test conditions are as follows: MV resolution = 1/4 pel, RDO = OFF, search range = 16 and reference frames = 1. The results for three sequences demonstrate the same observation and we only present the results for *forman* in the following discussion.

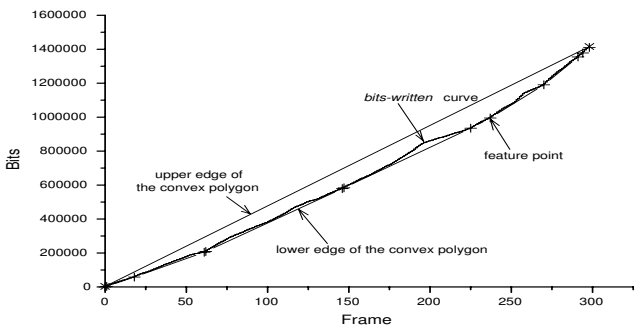


**Fig. 6**. The *bits-written* curve and the convex polygon when *forman* is encoded using a fixed quantization parameter that is equal to 28; the observed average bit rate is 144.0 Kbits/second

Fig. 6 illustrates the actual *bits-written* curve and the feature points obtained in the experiment, where we see the number of the feature points is quite small. Although we encode all 300 frames of the video sequence, only 15 feature points are observed. Table 1 lists the 5 results obtained by our algorithm at 5 different bit rates, which are referred to as $RS1$, $RS2, \ldots,$ $RS5$ respectively. Note

[1] As proved in [5], a linearly interpolated buffer size is able to safely contain the bitstream.

since all the *X* coordinates in the experiments are the indexes of the frames in the video sequence, the start-up delays in Table 1 are measured in *frame intervals*. To convert it to time in seconds, it needs to be divided by the frame rate. By our algorithm, $R_1$ is computed as 142Kbits/s (*RS2*) and $R_2$ is computed as 290Kbis/s (*RS4*). The corresponding buffer sizes are 135371bits and 9656bits respectively. The meaning of $R_1$ and $R_2$ is explained in Section 3.

**Table 1**. Five results obtained by the proposed algorithm

| Results | $RS1$ | $RS2$ | $RS3$ | $RS4$ | $RS5$ |
|---|---|---|---|---|---|
| $R$ (Kbits/s) | 75 | 142 | 225 | 290 | 350 |
| $B_{min}$ (bits) | 673984 | 135371 | 64867 | 9656 | 9656 |
| $\Gamma_{min}$ (frame intervals) | 268 | 1.15 | 0.41 | 0.32 | 0.27 |

*RS1* and *RS2* are optimal because, in these two cases, bit rates are smaller than or equal to $R_1$. *RS4* and *RS5* are also optimal because, in these two cases, $R$ is greater than or equal to $R_2$ and the max frame size $b_{max}$ is selected as the minimum buffer size. *RS3* is not optimal because, in this case, $R_1 < R < R_2$ and we use the upper edge and lower edge of the convex polygon to denote the actual *bits-written* curve and *bits-transmitted* curve.

As described in Section 1, interpolation or extrapolation may be needed by GHRD to determine the leaky bucket to use. At this point, our algorithm has the advantage over the GHRD since our algorithm does not need the interpolation or extrapolation operation and can always computes an optimal result when $R \leq R_1$ or $R \geq R_2$. When $R_1 < R < R_2$, the results by our algorithm is not optimal. In this case, we may use GHRD to determine minimum buffer size. Or else, a linearly interpolated buffer size based on $RS2$ and $RS4$ can be used, as shown by *RS3* in Table 1.

**Table 2**. Leaky bucket parameters $(R, B)$ computed by the Matlab program in [5];we assume the actual transmission bit rate is within the range $[\bar{R} - \bar{R}, \bar{R} + \bar{R}]$, where $\bar{R}$ is the observed average bit rate of the bitstream and is equal to 144Kbis/s in our experiment.

| R (Kbits/s) | 50 | 100 | 150 | 200 | 250 | 300 |
|---|---|---|---|---|---|---|
| B (bits) | 919317 | 424338 | 115992 | 40211 | 12691 | 9656 |

Table 2 lists the leaky bucket parameters that are used by GHRD to determine the leaky bucket for use. At any bit rate that is smaller than $R_1$, say $R = 75$Kbits/s, for GHRD, $B_{min}$ can be linearly interpolated and is 671828bits. However, by our algorithm, as shown in Table 1, $B_{min}$ is directly computed as 670984bits, which is slightly smaller than that by GHRD. At any bit rate that is greater than $R_2$, say $R = 350$Kbits/s, for GHRD, the buffer size is linearly extrapolated as 6621bits, which is smaller than $b_{max}$ and is not correct (this problem can be avoided by increasing the number of the leaky bucket parameters in the higher bit rates). However, for our algorithm, since $R \geq R_2$, $B_{min}$ can be directly set to $b_{max}$, i.e., 9656bits. When the bit rates are within the range from 142Kbits/s to 290Kbits/s, the result by GHRD will be better than ours in current experiments and we may use GHRD to derive the buffer size. In that case, we also need to record the leaky bucket parameters whose bit rates are between $R_1$ and $R_2$. For the start-up delay, our algorithm can obtain an optimal result at any bit rate. At this point, our algorithm also has the advantage over GHRD in which interpolation or extrapolation are needed to compute the start-up delay.

From Fig. 6, we see the observed bits variation of the bitstream is quite large since no restriction is imposed on the bits variation by the fixed-QP rate control scheme. The convex polygon can not bound the *bits-written* curve tightly. The non-optimal results when $R_1 < R < R_2$ have large potential to be improved. For example, by inserting several feature points, the single big convex polygon may be broken into several smaller polygons that bounds the *bits-written* curve much more tightly than current one. The $B_{min}$ by our algorithm when $R_1 < R < R_2$ can thus be further reduced. We have implemented this multi-polygon algorithm. However, we do not present it in this paper due to space stringency.

Our algorithm can achieve optimal buffer sizes for all bit rates if $R_1$ equals $R_2$, i.e., if the average frame size equals the max frame size. Thus, for most video streaming applications, in which the bit rate variation can not be too large and $R_2$ is close to $R_1$, our algorithm can achieve very good results.

## 5. CONCLUSION

A novel algorithm that is able to geometrically determine the optimal (when $R \leq R_1$ or $R \geq R_2$) and sub-optimal (when $R_1 < R < R_2$) buffer size and optimal start-up delay for streaming a pre-encoded bitstream over CBR channels at any bit rate is proposed. The proposed approach provides theoretical insight as well as a simple but effective algorithm for determination of the leaky bucket parameters. The devised algorithm require little extra information from the encoding process since only the maximum frame size and the coordinates of a few feature points need to be recorded. Moreover, the computational complexity is very small since it involves merely a few simple geometric operations that are easy to implement.

## 6. REFERENCES


[1] "Annex C, Video Buffering Verifier", in *Information Technology-Generic Coding of Moving Pictures and Associated Autio Information: Video (MPEG-2/H.262)*, 2000, ISO/IEC 138 180-2.

[2] "Levels for MPEG-4 Visual Profiles", ISO/IEC JTC1/SC29/WG11, MPEG2001/N4507, Pattaya, December 2001.

[3] "Annex B, Hypothetical Reference Decoder", in Video Coding for Low Bit Rate Communication, ITU-T Recommondation H.263, January 1998.

[4] E. Viscito, "HRD and related issues", Joint Video Team (JVT) of ITU-T SG16/Q15 (VCEG) and ISO/IEC JTC1/SC29/WG11 (MPEG), Klagenfurt, Austria, July 2002, Doc. JVT-D131.

[5] Jordi Ribas-Corbera, Philip A. Chou, and Shankar L. Regunathan, "A Generalized Hypothetical Reference Decoder for H.264/AVC", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, No. 7, July 2003.

[6] "Hypothtical Referece Decoder", in Joint Video Team (JVT) of ITU-T SG16/Q15 (VCEG) and ISO/IEC JTC1/SC29/WG11 (MPEG), Pattaya, Thailand, December 2001, Doc. JVT-B118.

[7] "JVT Test Model JM", Joint Video Team (JVT) of ITU-T SG16/Q15 (VCEG) and ISO/IEC JTC1/SC29/WG11 (MPEG), Klagenfurt, Austria, July 2002, Doc. JVT-D147.