An $O(n \log n)$ R-B Curve Drawing Algorithm for the Admission Control of VBR Video Transmission

Ming-Tat Ko, Jan-Ming Ho, Meng Chang Chen and Ray-I Chang

Institute of Information Science, Academia Sinica, Taiwan

{mtko,hoho,mcc,william}@iis.sinica.edu.tw

Abstract

The available resources such as memory buffer and network bandwidth are limited and varying for different application systems. Given a VBR stream, we have already proposed an algorithm to construct a transmission schedule with minimum buffer, minimum workahead and maximum network utilization for the given transmission rate. Notably, there is a trade-off between the allocated buffer size and the obtained network utilization to playback a VBR stream. To facilitate resource management and admission control for QoS guarantees, we need to explore the relation of the required client buffer and the obtained network utilization for the allocated transmission rate. Having these relation functions, whenever a new request is presented, the admission control procedure can easily check the required resources against the available resources and decides to admit this new request or not. The session setup protocol is as simple as a request-reply. Note that, the allocated transmission rate turns out to be a piecewise linear decreasing function of the memory buffer size. A native algorithm takes $O(n^3)$ to compute the rate-buffer (R-B) function where n is the number of video frames. In this paper, an $O(n \log n)$ R-B curve drawing algorithm is proposed. The same scheme can be extended to compute the relations between the transmission rate and the obtained network utilization.

[1] The research described in this paper was partially supported by NSC under grant NSC86-2223-E-001-019 and NSC86-2223-E-001-018.

1 Introduction and Problem Statement

Most of multimedia applications require quality-of-service (QoS) guarantees. Whenever a new request is presented, an admission control procedure is applied to decide if the obtained QoS is affected. If the available resources are not enough to support the new request, this request will be rejected. Memory buffer and network utilization are considered as two major resource measurements for QoS guarantees. The network utilization is defined as the total network bandwidth consumed divides the total network bandwidth allocated. A good admission control scheme is designed to admit as many requests as possible with the minimum buffer size and the maximum network utilization. Generally, an admission control scheme can be classified as statistical [5] or non-statistical (deterministic) [8]. As the statistical scheme can only theoretically guarantees a QoS bound, we focus on the deterministic scheme in this paper to provide guaranteed services. In modern ATM (asynchronous transfer mode) network, there are two major network service types, i.e., variable-bit-rate (VBR) and constant-bit-rate (CBR). In citegk96, a large-scale network framework with connections and switches are developed for simulations. It shows that, using CBR network services, the queuing delay is less than one cell time per switch even under heavy load. Moreover, as CBR services simply allocate fixed-size chunks to each service, they also have a very low load in management complexity. In contrast to VBR network services, the queuing delay is usually the most significant factors in end-to-end delays. Admission control and resource management for VBR services are usually quite complicated in both analysis and implementation. However, due to compression technology [3], media streams are usually VBR in nature. It is very important to design a good admission control schemes to the deterministic CBR network services for VBR media transmission.

In the previous, different approaches (such as D-BIND [12], RED-VBR [6], RCBR [5], MVBA [10, 9] and CRTT [7]) were proposed. However, they usually requires lots of memory buffer (i.e., CRTT requires 22.3MB memory buffer to playback *Star War* movie). In [2], we have presented an optimization algorithm called *LA* (Lazy-Aggressive) to construct a deterministic CBR transmission schedule. Given a video stream

and a transmission rate, the LA scheme can compute the maximize network utilization with the minimum buffer size and the minimum work-ahead in $O(n)$ time complexity. n is the number of data frames in the video stream. Notably, using different transmission rates to transmit a VBR stream stream may require different buffer size and obtain different network utilization. Besides, different clients may have different limitation in memory buffer size and network utilization. A good admission control procedure is necessary to give clients the flexibility to determine a suitable transmission rate for the available buffer size and the acceptable network utilization. In order to facilitate QoS guaranteed admission control, we need to explore the relations among the required resources and how they vary as a function of the applied transmission rate. In this paper, we want to compute the rate-buffer (R-B) function. The same scheme can be extended to compute the rate-utilization function. By applying these functions, whenever a new request is presented, the admission control procedure can easily check the required resources against the available resources and decides to admit this new request or not. The session setup protocol is as simple as a request-reply.

Generally, the increasing of allocated buffer size can increase the obtained network utilization. It is a trade-off. Thus, more video streams can be admitted if a suitable transmission rate is selected to fit the system configuration. By following the operations of LA scheme, it can be easily found that the R-B characteristic function is piecewise non-increasing when the transmission rate is increased. The R-B function can be computed with time complexity $O(n^3)$ by a brute force algorithm. In this paper, an $O(n \log n)$ algorithm is proposed to draw the R-B curve for a stored VBR video stream. The remainder of this paper is organized as follows. The operations of LA scheme for a given transmission rate are first review in Section 2 with some primary definitions. In Section 3, some observations in the variations of client buffer size for the increasing of transmission rate are presented. The related mathematical formulations are also given. Based on these observations, an $O(n \log n)$ algorithm is presented in Section 4 to compute the R-B function. Concluding remarks and future directions are given in section 5.

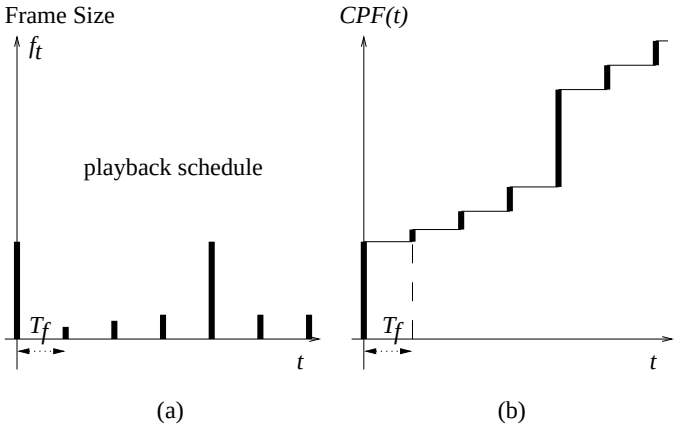

Figure 1: (a) A VBR stream. (b) The related cumulative playback function.

2 Minimize Buffer Requirement for a Given Rate

Before introducing the proposed R-B curve drawing algorithm, we at first review the buffer minimization problem and the LA method in this section. Assume that a video stream $\mathcal{V}$ consists of a sequence of frames $\mathcal{V} = \{f_0 f_1 \dots f_{n-1}; T_f\}$ where f_i denotes the i-th frame as well as its frame size and T_f is the time interval between adjacent frames. The cumulative frame size of $\mathcal{V}$ is defined as $F_i = F_{i-1} + f_i$, where $0 \leq i < n$ and $F_{-1} = 0$. Assume that a video stream is playback at $t = 0$. The cumulative playback function $CPF(t)$ is defined as

$$CPF(t) = \begin{cases} 0, & t < 0 \\ F_i, & i * T_f \leq t < (i+1) * T_f \text{ where } 0 \leq i < n, \\ F_n, & t \geq (n-1) * T_f. \end{cases}$$

Given a VBR stream as shown in Fig. 1(a), the curve of its $CPF(t)$ is illustrated in Fig. 1(b). In a video server, media data in the storage system are at first retrieved to the server buffer by the retrieval schedule (such as SCAN-EDF [1]). According to the transmission schedule, data in the server buffer are then transmitted at the proper time to the client buffer. At last, the client consumes video data in the client buffer frame by frame as shown in Fig. 1. The buffer minimization problem on designing a transmission schedule is to minimize the client buffer size for a given transmission rate.

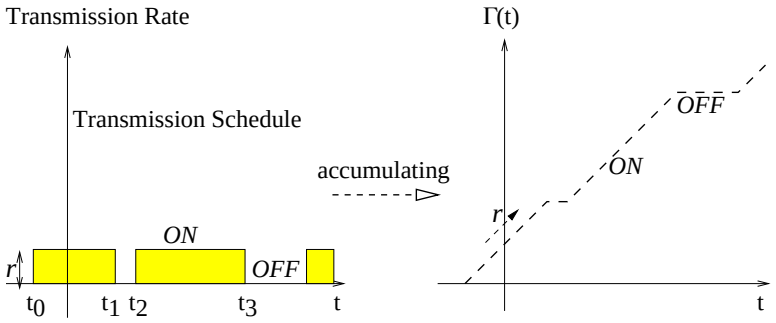

Figure 2: The transmission schedule and its cumulative transmission function for ON-OFF CBR transmission model.

In the *LA* scheme, we apply the *ON-OFF* [11] resource model for CBR network transmission. A transmission schedule is defined as $\gamma = \{r; t_0, t_1, \ldots, t_{2m+1}\}$. It transmits data at rate r in time slot (t_{2i}, t_{2i+1}) and sends nothing in time slot $(t_{2i+1}, t_{2i+2}) where 0 \leq i < m$. A typical example of the ON-OFF model and its cumulative transmission function is shown in Fig. 2. The cumulative transmission function $\Gamma(t)$ is defined as the integration of $\gamma(t)$. For the simplification, We denote $\Gamma(k * T_f)$ as Γ_k in this paper without loss the generality. As the transmission data must be always ahead of the playback data required, the cumulative transmission function should not be lower than the cumulative playback function. Thus, a feasible transmission schedule must satisfy $\Gamma(t) \geq CPF(t)$. Notably, $\Gamma(t)$ - $CPF(t)$ is just the amount of transmitted data temporarily stored in the client buffer at time t. The required maximum client buffer can be defined as $b_{\mathcal{V},\gamma} = \max\{\Gamma(t) - CPF(t); \forall t\}$. As video is playback at $t = 0$, the value $w = -t_0$ is referred to as the *work-ahead*. The network utilization is defined as the ratio of the data transmitted by the transmission schedule over the maximum data that can be transmitted by the same network connection time. In this paper, we focus only on the optimization of required client buffer to draw the rate-buffer (R-B) curve.

Given a constant r as the transmission rate, we have presented *Algorithm Lazy* [2] to compute the minimum buffer transmission schedule $\gamma_L = \{r; t_0, t_1, \ldots, t_{2m+1}\}$. Given a video stream $\mathcal{V}$ and a transmission rate r, *Algorithm L* computes a sequence L_k as follows:

$$\begin{cases} L_i = |\mathcal{V}| & \forall i \geq n-1, \text{and} \\ L_k = \max\{F_k, L_{k+1} - r * T_f\} & \forall 0 \leq k < n-1. \end{cases}$$

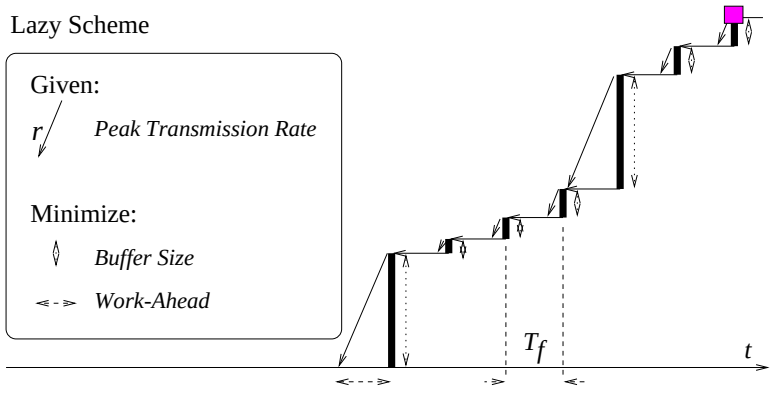

Figure 3: A simple processing example of Algorithm L.

An example illustrating the computation of the transmission schedule γ_L by *Algorithm L* is shown in Fig. 3. It can be easily proved that L_k can present a feasible transmission schedule to transport $\mathcal{V}$. Note that, as γ_L captures the intuition to transmit data as late as possible, the required client buffer size $b_{\mathcal{V},\gamma_L}$ is minimum.

Theorem 1 $b_{\mathcal{V}}(r) = \max_{k=0}^{n}\{L_k - F_{k-1}\}$ *is the minimum client buffer size for transporting video stream* $\mathcal{V}$ *at peak transmission rate* r *(detail proof is shown in [2]).*

We can use a similar proof to show that γ_L has the minimum work-ahead for transporting $\mathcal{V}$. Given the work-ahead and the buffer size, we have also presented *Algorithm Aggressive* to maximize the network utilization [2]. Thus, given a stored video stream and a transmission rate, the maximize network utilization, the minimum client buffer and the minimum work-ahead can be computed in $O(n)$ time complexity by *Algorithm Lazy-Aggressive (LA)*. In this paper, as our goal is to explore the rate-buffer tradeoff in stream transmission, only the buffer minimization procedure is presented. Interested readers may refer to [2] for further discussions.

3 Relations of Buffer Size and Transmission Rate

In this section, observations about the changes of client buffer sizes to different transmission rates are given. We consider the minimum buffer transmission schedules obtained by *Algorithm Lazy* from transmission rate 0

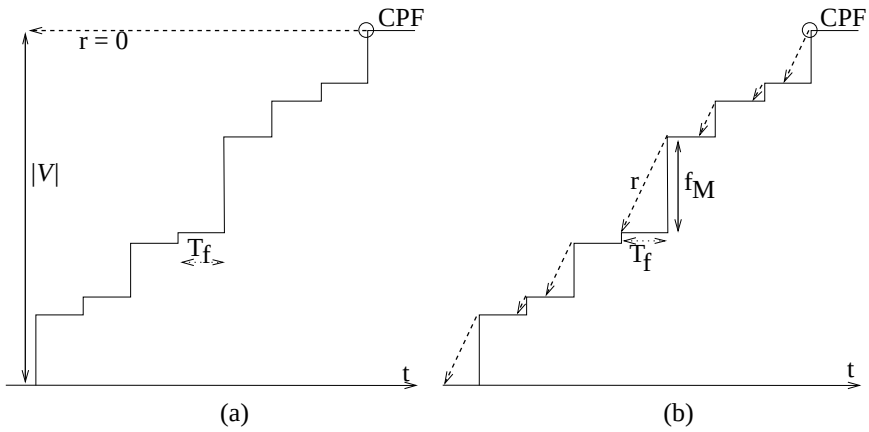

Figure 4: (a) When the transmission rate is close to 0, the client buffer size is nearly the same as the size of the video stream. (b) If $r * T_f$ is larger or equal to the maximum frame size, the client buffer size is exactly the maximum frame size.

to ∞. In a minimum buffer transmission schedule, define $\hat{b}_{\mathcal{V}}(r,t)$ as the size of data stored in the buffer at time t which transmits video stream $\mathcal{V}$ with rate r. Assume that, in the above transmission schedule, the maximum buffer requirement is happen at time $i * T_f$ for some an integer i. This integer i is called the *maximum buffer point* and is denoted by $i_M(r)$. That is, $\hat{b}_{\mathcal{V}}(r) = \hat{b}_{\mathcal{V}}(r, i_M(r) * T_f) = \max\{\hat{b}_{\mathcal{V}}(r, i * T_f)|, 0 \leq i \leq n-1\}$. Let's first consider the boundary cases with very small and very large transmission rates. When r is close to 0 as shown in Fig. 4(a), the client buffer size $\hat{b}_{\mathcal{V}}(r)$ is nearly the same as the video size $|\mathcal{V}|$ and $i_M(r) = 0$. On the other hand, if $r * T_f$ is not less than the maximum frame size $f_M = \max_{i=0}^{n-1} f_i$ as shown in Fig. 4(b), the size of data stored in the buffer is $\hat{b}_{\mathcal{V}}(r, i * T_f) = f_M$. The required client buffer size is the maximum frame size f_M and $i_M(r) = M$.

Define an ON segment (or segment, for short) as a sequence of continuous ON states that started at one OFF point and ended at the next OFF point. Notably, in the boundary case with transmission rate 0, there is only one ON segment. In the boundary case with rate $r >= f_M/T_f$, the transmission schedule γ_L is decomposed into n transmission segments. In this paper, we denote the segment S_x by $[i_s^x(r), i_e^x(r)]$, where $i_s^x(r)$ is its start OFF point and $i_e^x(r)$ is its end OFF point. Note that, the maximum buffer point must be in one of these ON segments as shown in Fig. 5(a). Define the point which achieves the largest buffer

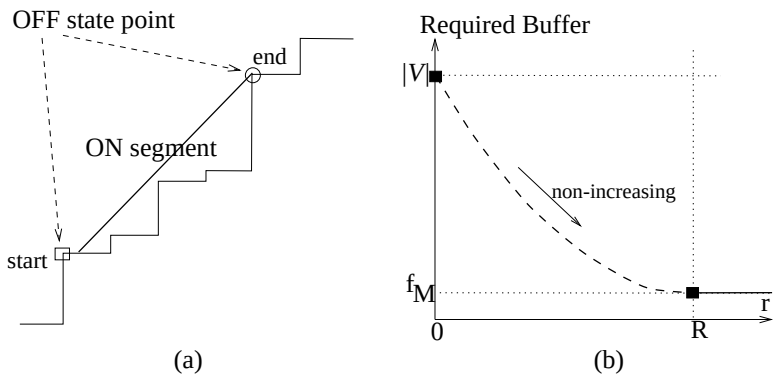

Figure 5: (a) The ON segments and the OFF state points. (b) The relation between the required client buffer size and the transmission rate.

requirement in a segment as the *largest buffer point* of this segment. If there are more than one points in a segment that have the largest buffer size, we select the one with the largest index as the largest buffer point. The maximum buffer point is at the largest buffer point of *maximum buffer segment*. If there are more than one segments that have the maximum buffer requirement, the one with the largest index is selected as the maximum buffer segment. As the rate is changed, the ON segment may be separated further and the largest buffer point may shift to another index point. Besides, even the maximum buffer segment may switch to another segment. Although the maximum buffer segment and maximum buffer point may be changed, it can be easily found that the size of data stored in the buffer is non-increasing as the transmission rate is increased. Thus, the maximum buffer size $\hat{b}_{\mathcal{V}}(r)$ is non-increasing as the transmission rate is increased. This observation can be easily shown by following the induction rule and *Algorithm Lazy*. A rough sketch of the minimum buffer size $\hat{b}_{\mathcal{V}}(r)$ for different transmission rates r is depicted in Fig. 5(b). The curve is continuous and non-increasing. As shown in Fig. 6, if the maximum buffer segment and the maximum buffer point are not changed, the required buffer size is linearly decreasing by a constant slope when the transmission rate is increasing. This observation can be easily shown by the triangular formula. The required client buffer size is non-increasing piecewise linear as the transmission rate is increased.

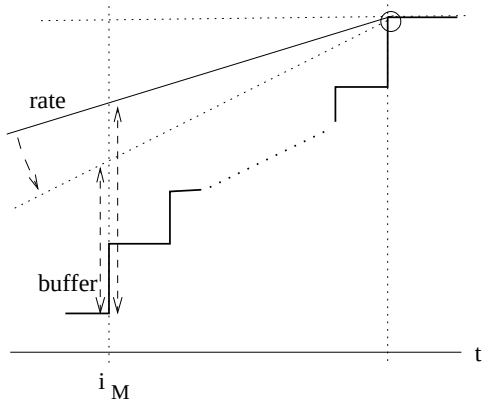

Figure 6: As shown by the triangular formula, generally, the required client buffer size is linearly decreasing by a constant slope when the transmission rate is increasing.

In the following, we try to increase the transmission rate by a small value and observe the possible change of required buffer size. Let $S_x = [i_s^x(r), i_e^x(r)]$ be a segment and its largest buffer point is at $i_L^x(r)$. Firstly, we consider a small range of transmission rates around r such that no segment is separated, no largest buffer point is shifted and the maximum buffer segment remains the same one. In this case, for the time i in S_x, we have $\hat{b}_{\mathcal{V}}(r + \Delta r, i) = \hat{b}_{\mathcal{V}}(r, i) - \Delta r * T_f * (i_e^x(r) - i)$. In particular, taking i as $i_L^x(r)$, we may see that the largest buffer size in the segment changes linearly with the slope $T_f * (i_e^x(r) - i_L^x(r))$. In other words, the slope of change for the required buffer size in a segment depends on the distance between the end OFF state point $i_e^x(r)$ and the largest buffer point $i_L^x(r)$. In particular, the decreasing slope of required buffer size in the maximum buffer segment at r, called *buffer-decreasing-slope*, is the slope of R-B curve at r.

The buffer-decreasing-slope may change at one of the following two cases. (1) The end OFF state point of the maximum buffer segment changes. (2) The largest buffer point of the maximum buffer segment changes. The first case happens when the maximum buffer segment is separated. In the following, we look into the case that a segment $S_x = [i_s^x(r), i_e^x(r)]$ is separated. As the transmission rate increases to a certain rate, say r', a new OFF state is inserted into S_x at some index j and S_x is separated further into the left sub-segment $S_{x1} = [i_s^x(r), j]$ and the right sub-segment $S_{x2} = [j + 1, i_e^x(r)]$. Such a transmission rate is called a *separating rate*. At rate r', the points in the left sub-segment have a new end OFF state point, and the

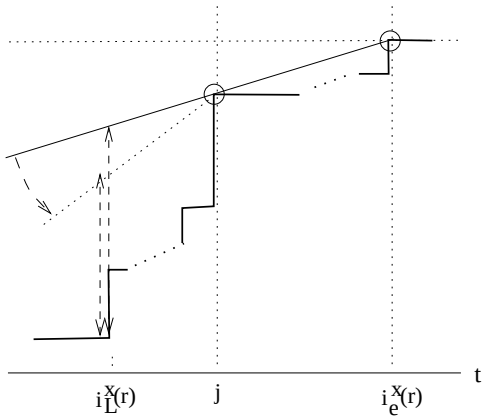

Figure 7: If the largest buffer point is ahead of the new OFF state point, the buffer-decreasing-slope is decreased.

decreasing slope of the buffer size at the point changes accordingly. For the points in the right sub-segment, the decreasing slope of the buffer size does not change. In particular, if S_x is the maximum buffer segment and the largest buffer point $i_L^x(r)$ is in the left sub-segment (as shown in Fig. 7), the buffer-decreasing-slope changes from $-T_f * (i_e^x(r) - i_L^x(r))$ to $-T_f * (j - i_L^x(r))$. If the largest buffer point is in the right sub-segment, the buffer-decreasing-slope would not change.

In general, the buffer-decreasing-slope gets larger as the rate increases cross separating-rates. However, there is a special case that the largest buffer point $i_L^x(r)$ is j as shown in Fig. 8. In this case, the required buffer size keeps to be the size of frame f_j even when the transmission rate increases further. The buffer-decreasing-slope then decreases to the boundary value 0. Now that the required buffer size does not change any more, the algorithm for drawing of rate-buffer curve stops.

Next, we consider the second case that the segment is not separated but the largest buffer point changes as the rate increases from r to r' (see Fig. 9). That is $i_e^x(r) = i_e^x(r')$ and $i_s^x(r) = i_s^x(r')$ but $i_L^x(r) \neq i_L^x(r')$. It is obvious that $i_L^x(r) < i_L^x(r')$ and $i_s^x(r') \leq i_L^x(r') \leq i_e^x(r')$. From the definition of $i_L^x(r')$, the required buffer size at point $i_L^x(r')$ is smaller than that at point $i_L^x(r)$ with transmission rate r. Besides, the required buffer size of point $i_L^x(r')$ is larger than that of point $i_L^x(r)$ with transmission rate r'. As $\hat{b}_{\mathcal{V}}(r, i_L^x(r') * T_f) < \hat{b}_{\mathcal{V}}(r, i_L^x(r) * T_f)$ and $\hat{b}_{\mathcal{V}}(r', i_L^x(r') * T_f) > \hat{b}_{\mathcal{V}}(r', i_L^x(r) * T_f)$, there exists r" such that $r < r$" $< r'$ and

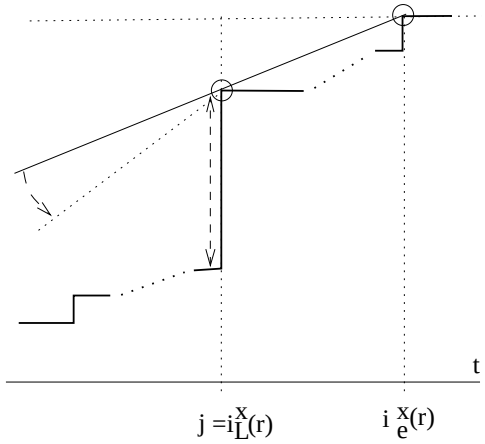

Figure 8: If the new largest buffer point is just at the end OFF state point, the buffer-decreasing-slope would be changed to 0.

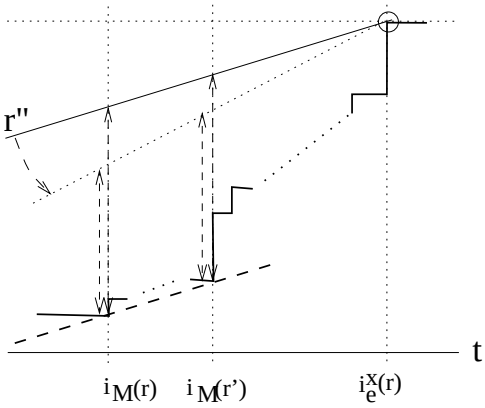

Figure 9: The end OFF state point is not changed and the largest buffer point is changed at an intra-segment equal-buffer-rate.

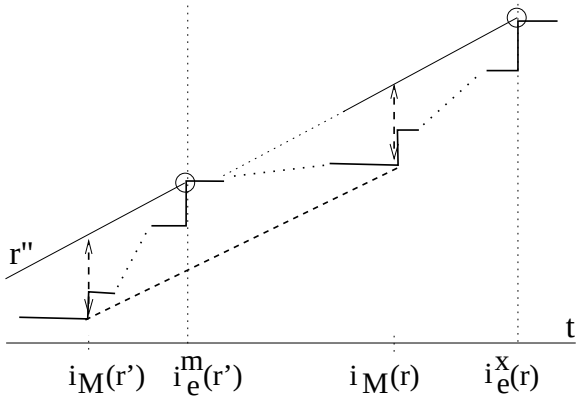

Figure 10: Both the end OFF state point and the largest buffer point are changed at an inter-segment equal-buffer-rate.

$\hat{b}_{\mathcal{V}}(r", i_L^x(r) * T_f) = \hat{b}_{\mathcal{V}}(r", i_L^x(r') * T_f)$. The rate r" is called the *equal-buffer-rate*. By the above inequalities, we have $(i_e^x(r) - i_L^x(r)) > (i_e^x(r) - i_L^x(r'))$ and $i_s^x(r) \leq i_L^x(r) < i_L^x(r') \leq i_e^x(r)$. The new rate of client buffer size decreasing is $(i_e^x(r) - i_L^x(r'))$.

Note that, as shown in Fig. 10, as the rate increases, the maximum buffer point may shift to other segment. Assume that the new maximum buffer point $i_M(r')$ is in the segment $S_m = [i_s^m(r'), i_e^m(r')]$. $i_M(r')$ $= i_L^m(r')$ in segment S_m. From the definition, the required buffer size of point $i_M(r')$ is smaller than that of point $i_M(r)$ at rate r. Besides, the required buffer size at point $i_M(r')$ is larger than that at point $i_M(r)$ for the transmission rate $r' > r$. There must be one equal-buffer-rate r" exist $(r < r" < r')$. It implies that $(i_e^x(r) - i_M(r)) > (i_e^m(r') - i_M(r'))$ where S_x is the maximum buffer segment at rate r. At rate r', the new buffer-decreasing-slope decreases to $(i_e^m(r') - i_M(r'))$.

The required buffer size is non-increasing piecewise-linear as described above. Thus, it would be enough to precisely model the rate-buffer relation by a suitable finite set of the transmission rates and their required buffer sizes. From the above observations of separating-rate and equal-buffer-rate, it suffices to compute the characteristic curve over a finite set of critical rates of $r^c = (F_i - F_j)/(i - j), 0 \leq i, j \leq n - 1$. By considering these $O(n^2)$ critical rates, the naive algorithm takes $O(n^3)$ time to compute the rate-buffer curve. In general, the slope of rate change decreases at either equal-buffer-rate or separating-rate. As there

are at most n OFF states, we should consider at most n separating-rates. Besides, as there are at most n different distances between the end OFF state point and the largest buffer point, we should consider at most n equal-buffer-rates. There are at most n different buffer-decreasing-slopes in the rate-buffer curve. In this paper, an $O(n \log n)$ algorithm is proposed to draw the rate-buffer (R-B) curve.

4 Implementation of an $O(n \log n)$ R-B Curve Drawing Algorithm

As discussed in the last section, the R-B curve is piecewise linear. Besides, the rate at which the R-B curve changes slope must be a separating-rate or an equal-buffer-rate. We call these points are the *turning points*. Before introducing our $O(n \log n)$ algorithm for drawing the R-B curve, we present the data structures for efficiently retrieving the separating-rates and equal-buffer-rates. As shown in Fig. 1, the $CPF(t)$ is represented by an increasing staircase of t. The direction of the staircase is from the left to the right. We denote the inner and the outer corner points at i as $c_{in,i}$ and $c_{out,i}$ respectively for all $0 \leq i \leq n$. Let S_x be a segment at rate r starting at the index $i_s^x(r)$ and ending at $i_e^x(r)$. By definition, the ray of slope r pointing downwards from the outer corner point $c_{out,i_e^x(r)}$ will hit the staircase at a point in the line segment $\overline{c_{out,i_s^x(r)}, c_{in,i_e^x(r)+1}}$. Keeping the start point of the ray fixed and let the slope go up, at a rate r', the ray will touch an outer corner point $c_{out,j}$ of the staircase for some $i_s^x(r) < j < i_e^x(r)$. At this rate, the segment is separated into two segments. The left segment starts at the index $i_s^x(r)$ and ends at j and the right segment starts at j and ends at $i_e^x(r)$. At the same time, the ray is also separated into two rays, one starts at $c_{out,i_e^x(r)}$ for the right segment and the other starts at $c_{out,j}$ for the left segment. As the slope going up, the segments are separated further until every segment is a corner point.

We may exploit all the separating rates by increasing the rate from 0 to ∞ and for each segment finding the next separating rate with which its ray touches a corner point in the segment. However, the time complexity of the algorithm will be $O(n^2)$. In this paper, we exploit all the separating rates by another algorithm in $O(n)$ time. The algorithm is similar to the incremental algorithm for constructing the convex upper envelope of the outer corner points of the staircase. All the outer corner points are under the convex upper envelope. At

the beginning, we exploit the separating rates of staircase from corner point $c_{out,n-1}$ to $c_{out,n}$, and then that of staircase from $c_{out,n-2}$ to $c_{out,n}$, and so on until that of the whole staircase from $c_{out,0}$ to $c_{out,n}$. Assume that the convex upper envelope of the staircase from $c_{out,k}$ to $c_{out,n}$ is already constructed and we are going to construct the convex upper envelope of the staircase from $c_{out,k-1}$ to $c_{out,n}$. The algorithm is to connect $c_{out,k-1}$ and $c_{out,k}$ by a linear segment and check whether the angle counterclockwise from $\overline{c_{out,k-1}, c_{out,k}}$ to $\overline{c_{out,k}, c_{out,k+1}}$ is less than or equal to π. If it does, the convex upper envelope of the staircase from $c_{out,k}$ to $c_{out,n}$ together with the line segment $\overline{c_{out,k-1}, c_{out,k}}$ is that of the staircase from $c_{out,k-1}$ to $c_{out,n}$. If it is not, test the angle from $\overline{c_{out,k-1}, c_{out,j}}$ to $\overline{c_{out,j}, c_{out,j+1}}$ for $j > k$ sequentially until the angle is less than or equal to π. Compared with the algorithm constructing the convex upper envelope, besides the line segments in the convex upper envelope of the whole staircase, that in the convex upper envelope of partial staircases need to be recorded in the algorithm exploiting all separating rates.

The relation between a segment and its two separated sub-segments can be intuitively represented by a binary tree structure as shown in Fig. 11. The separating-rate and the separating point of the segment of the whole staircase is stored in the root node. The left pointer and the right pointer are used to refer the left sub-segment and the right sub-segment respectively. It can be easily found that the separating-rate of root node would be smaller than that of either the left branch node or the right branch node. Hence, the binary tree is of a heap structure on the separating rates. The convex upper envelopes of partial staircases form a planar graph with all the outer corner points as the vertices. Since the line segments of the convex upper envelopes correspond to separating rates, there are $O(n)$ separating rates.

Next, we consider the intra-segment equal-buffer-rates. The observation is as follows. For a segment of the staircase, let us consider the convex lower envelope of its inner corner points. All the inner corner points of the segment are above the convex lower envelope. Let the largest buffer point of the segment be at i. Then the inner corner point $c_{in,i}$ must be a vertex of the convex lower envelope. Otherwise, there is a line segment $l = \overline{c_{in,j}, c_{in,k}}$ in the convex lower envelope such that $j < i < k$. If the rate r is larger than (smaller than resp.) the slope of l, the buffer size at k (j resp.) is larger than that at i. That is a contraction to that

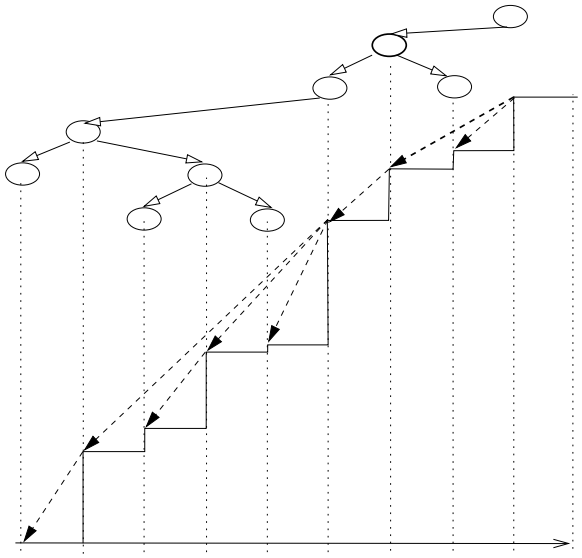

Figure 11: The constructed tree data structure for different separating-rates.

i is the largest buffer point. Let $l_m = \overline{c_{in,i}, c_{in,k}}$, $i < k$, be a line segment of the convex lower envelope. As the rate increases to the slope of l_m, we will see that the buffer size at i is the same with that at k. The largest buffer point of the segment then changes from i to k for the larger rate.

To find the intra-segment equal-buffer-rate efficiently for all the segments, we construct the convex lower envelopes of equal-buffer segments from the leaf of the binary tree of separating segments to the root. Let S be a segment from index i_s to i_e and let its left separating sub-segment S_l be from i_s to i_m and its right separating sub-segment S_r be from $i_m + 1$ to i_e. Assume that the convex lower envelopes of the two sub-segments are already constructed. Let $\overline{c_{in,j}, c_{in,k}}$ be the tangent line segment of the two convex lower envelopes. The convex lower envelope of S is then consisted of the part of convex lower envelope of S_l from c_{in,i_s} up to $c_{in,j}$, the line segment $\overline{c_{in,j}, c_{in,k}}$ and the part of the convex lower envelope of S_r from $c_{in,k}$ to c_{in,i_e}. The constructing algorithm of the convex lower envelopes of all the segments is similar to the one for constructing the convex upper envelopes. The convex lower envelopes of all the segments also form planar graph with all the inner corner points as vertices. Thus, there are $O(n)$ intra-segment equal-buffer-rates.

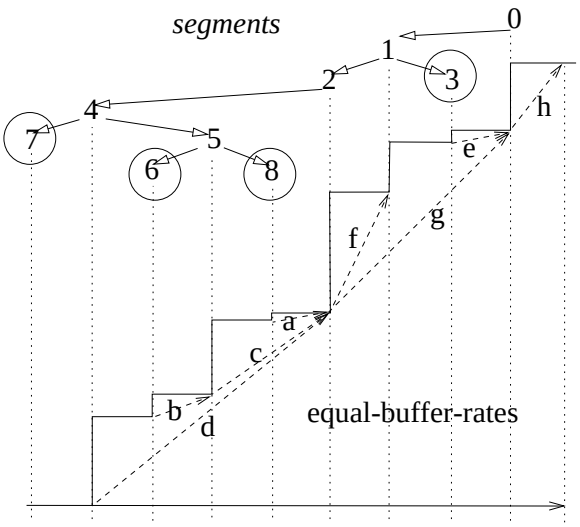

Figure 12: The constructed tree data structure for different equal-buffer-rates.

With the data structures for the separating rates and the intra-segment equal-buffer-rates, we may maintain the structure of separating segments and the largest buffer point of each segment at a rate r as follows. We process the rates in a decreasing order. Assume that r is a separating rate of segment S_x and two sub-segments S_{x1} and S_{x2} are created. The convex lower envelopes of S_{x1} and S_{x2} are constructed by removing the tangent line of the convex lower envelopes of S_x. Following the convex lower envelope, the largest buffer point and the buffer size can be found. If r is an intra-segment equal-buffer-rate of segment S_x, we just change the largest buffer point of S_x. Note that the decreasing slope of the largest buffer size is changed as the location of the largest buffer point is changed. Since there are $O(n)$ such rates to be processed and the largest buffer point finding traverse the line segments of the convex lower envelopes once, it takes total $O(n)$ to maintain the segment structure and the largest buffer points.

Now let us consider the inter-segment equal-buffer-rates. Let $S_1, S_2, \cdots, S_k$ be the segments at rate r and $B_x(r)$ is the largest buffer size of S_x at rate r for all $1 \leq x \leq k$. Assume that $B_1(r) \leq B_2(r) \leq \cdots \leq B_k(r)$ as shown in Fig. 13. For each segment, the decreasing slope of the largest buffer size is shown by dashed line. The upper envelope of these decreasing slopes is shown by a solid polyline to represent the maximum buffer size required. Let the starting point, the ending point and the largest buffer point of S_x are denoted by $i_s^x(r)$,

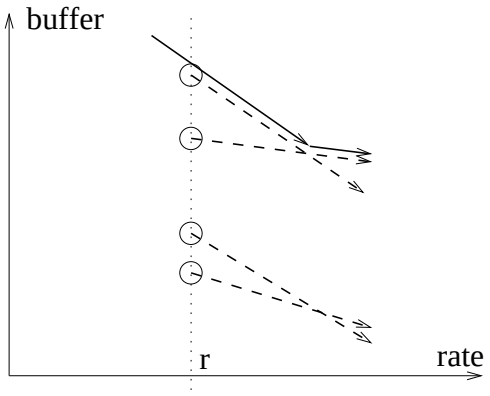

Figure 13: The largest buffer size for different transmission segments at rate r. The upper envelop represents the rate-buffer curve.

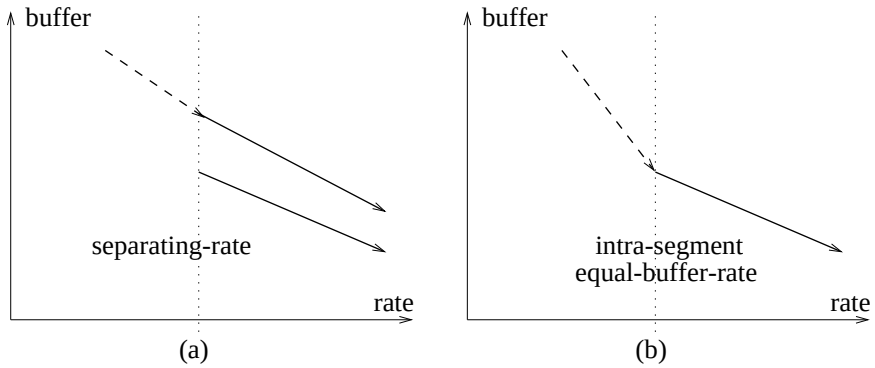

Figure 14: (a) At the separating-rate, the decreasing slope of the largest buffer size in the original segment may be decreased. Besides, a new segment is created with a new decreasing slope of the related largest buffer size. (b) At the intra-segment equal-buffer rate, the decreasing slope of the largest buffer size in the related segment is decreased.

$i_e^x(r)$ and $i_L^x(r)$ respectively for all $1 \leq x \leq k$. The decreasing slope of the largest buffer size in the segment S_x is $i_L^x(r) - i_e^x(r)$. Thus, as the rate goes up, though its largest buffer size is smaller that the maximum buffer size, a segment S_x with $i_L^x(r) - i_e^x(r)$ smaller than $i_L^1(r) - i_e^1(r)$ may become the maximum buffer segment. The inter-segment equal-buffer-rate can be obtained by computing the intersections of the buffer decreasing slopes in different segments. Note that, in each segment, the decreasing slope of the largest buffer size is changed only at the related separating rate (as shown in Fig. 14(a)) or the intra-segment equal-buffer rate (as shown in Fig. 14(b)).

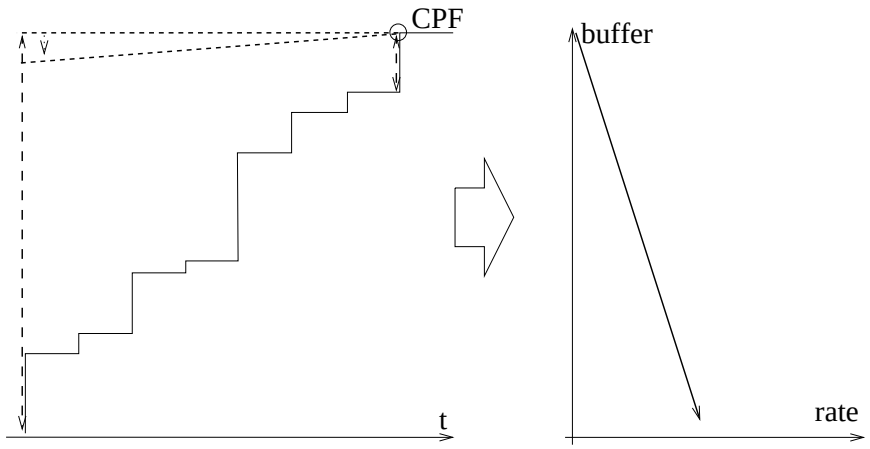

Figure 15: The initial envelope for the proposed algorithm to draw the R-B curve.

Now, we present the $O(n \log n)$ algorithm for drawing the R-B curve. Note that the turning points of the R-B curve can be either the separating rates, the intra-segment equal-buffer-rates, or the inter-segment equal-buffer-rates. The separating rates and the intra-segment equal-buffer-rates can be pre-computed by $O(n \log n)$ time. However, since the inter-segment equal-buffer-rates are dependent on the structure of separating segments, we exploit them dynamically.

Our algorithm processes the critical rates in the increasing order. We at first sort the exploited separating rates and the intra-segment equal-buffer-rates $\{r_1, r_2, \cdots, r_k\}$ into an increasing sequence. Since there are $O(n)$ such rates, the sorting takes $O(n \log n)$ time. Let $r_1, r_2, \cdots, r_k$ be the increasing sorted rates. The smallest rate $r_1 = 0$ is considered and the initial envelope of R-B curve is shown as Fig. 15 to represent the decreasing slope of buffer size. When a rate r_i is selected for processing, one or two lines are added to the current envelope as shown in Fig. 14. These lines represent the possible change of decreasing slope for the R-B curve. By finding the intersection points of the current envelope and the added lines, it takes $O(\log n)$ to compute the new envelope as shown in Fig. 16. Note that, these intersection points are just the equal-buffer rates to correctly draw the R-B curve.

Since a turning point occur at each inter-segment equal-buffer-rate, in the algorithm only $O(n)$ inter-segment equal-buffer-rates are processed. It takes $O(\log n)$ time to find one such rate. Thus, together with

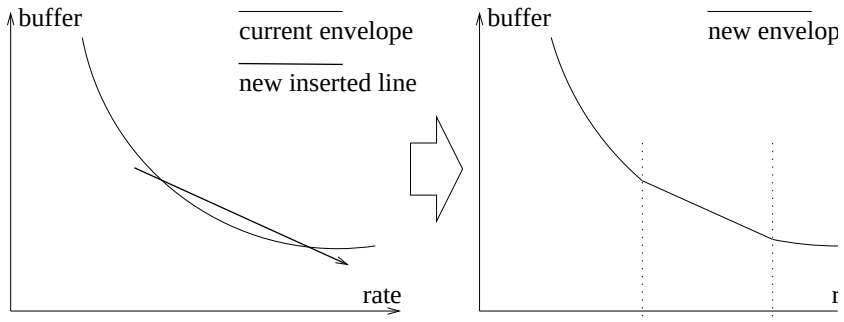

Figure 16: The new envelope is computed by inserting a line to the current envelope of R-B curve.

the time to maintain the segment structure and the largest buffer point of each segment, the complexity of the algorithm is $O(n \log n)$.

5 Conclusion

Network bandwidth and memory buffer are two most important resources for a video-on-demand system. However, small network bandwidth allocation will lead large memory buffer requirement, and vise versa. It is necessary to allocating the minimum buffer size for any given transmission rate for admission control. This approach shows great flexibility to allow various clients to set up their best transmission schedules. For example, the optimality may be in network bandwidth that drive the network bandwidth as small as possible with the cost of extra buffer. In this paper, an $O(n \log n)$ algorithm is proposed to draw a characteristic function to investigate the tradeoff between the transmission rate and the minimum required buffer size. The same idea can be applied to compute the other characteristic functions such as the tradeoff between the transmission rate and the work-ahead (or the resource utilization) in $O(n \log n)$ time.

References

[1] E. Chang and A. Zakhor. Scalable video data placement on parallel disk arrays. In *IS&T/SPIE Symposium on Electronic Imaging Science and Technology*, February 1994.

[2] Ray-I Chang, Meng Chang Chen, Ming-Tat Ko, and Jan-Ming Ho. Optimizations of stored VBR video transmission on CBR channel. 1996.

[3] M. Garrett and W. Willinger. Analysis, modeling and generation of self-similar VBR video traffic. In *Proc. ACM SIGCOMM*, pages 269–280, August 1994.

[4] M. Grossglauser and S. Keshav. On CBR service. In *Proc. IEEE INFOCOM*, March 1996.

[5] M. Grossglauser, S. Keshav, and D. Tse. RCBR: a simple and efficient service for multiple time-scale traffic. In *Proc. ACM SIGCOMM*, August 1995.

[6] E. W. Knightly, D. E. Wrege, J. Liebeherr, and H. Zhang. Fundamental limits and tradeoffs of providing deterministic guarantees to VBR video traffic. In *Proc. ACM SIGMETRICS*, pages 47–55, may 1995. D-BIND model.

[7] J. M. McManus and K. W. Ross. Video on demand over ATM: Constant-rate transmission and transport. In *Proc. IEEE INFOCOM*, March 1996.

[8] H. G. Perros and K. M. Elsayed. Call admission control schemes: a review. *IEEE Communication Magazine*, pages 82–91, November 1996.

[9] A. R. Reibman and A. W. Berger. Traffic descriptors for VBR video teleconferencing over ATM networks. *IEEE/ACM Transactions on Networking*, June 1995.

[10] 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. In *Proc. ACM SIGMETRICS*, 1996.

[11] K. Sohraby. On the theory of general ON-OFF source with applications in high-speed networks. In *IEEE INFOCOM*, pages 401–410, 1993.

[12] H. Zhang and E. W. Knightly. A new approach to support delay-sensitive VBR video in packet-switched networks. In *Proc. 5th Workshop on Network and Operating Systems Support for Digital Audio and Video*, 1995.