Optimal Bandwidth-Buffer Tradeoff for VBR Media Transmission over the Relay-Server

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

Institute of Information Science
Academia Sinica
Taipei, Taiwan, ROC

TEL: 886-2-27883799, FAX: 886-2-7824814
{william, mcc, hoho, mtko}@iis.sinica.edu.tw
Abstract

In a client-server system, the minimum bandwidth required to transmit a pre-recorded VBR media can be computed in $O(n)$. As the frame number $n$ is usually very large, this resource management procedure is not suitable for online computation. Although an $O(n \log n)$ algorithm has been proposed to characterize the bandwidth-buffer tradeoff for the optimal resource management (a native algorithm takes $O(n^3)$), this algorithm is not suitable for a general network system with additional relay-server. In this paper, we extend the problem model to consider the relay-server between client and server. This proposed model is good for scalable multimedia and fault-tolerance. Besides, the additional buffer in relay-server can be utilized to further smooth traffic and support more requests. In this paper, an $O(n \log n)$ algorithm is proposed to decide the optimal bandwidth-buffer tradeoff for the relay-server. Based on the pre-computed tradeoff function, we can simply design a good QoS control procedure to allocate the suitable bandwidth for the available buffer size.
1. Introduction

To support continuous media playback, requests in a multimedia system (such as digital library [3] and video-on-demand [4]) require guaranteed QoS (quality-of-service) control and resource management in disks and networks [5-7]. Different from the constant-bit-rate (CBR) traffic, media data are usually variable-bit-rate (VBR) due to the compression technology applied (such as MPEG) [1-2]. It makes the design of a good multimedia scheduler more complicated -- especially the transmission schedule over a general multimedia network. In a general multimedia network with multiple network nodes, the available resources (such as the memory buffer and the network bandwidth) are limited and various in different nodes and different connections. For a coming request, the system needs to know the available resources and to decide the admission of the coming request for supporting guaranteed QoS. To admit as many requests as possible, a good QoS control and resource management procedure should be provided [11,18].

Generally, there is a tradeoff between the available buffer size and the allocated network bandwidth. The increasing of buffer size can reduce the required network bandwidth. If the minimum network bandwidth is selected to fit the best system configurations, more media streams can be admitted. Notably, using different network bandwidths to transmit a VBR media may require different memory buffers and provide different system performances. Besides, different network nodes may present different limitation in the available system resources. Based on DAVIC [24], client’s request is sent to a level-1 gateway (called application-server) to allocate a suitable information flow (called transmission path). The minimum network bandwidth requirement for the allocated transmission path should be decided to satisfy the resource constraints and optimize the system performance. If the available resources are not enough to support this coming request, the request will be rejected.

In the previous years, different approaches [7-15] were proposed to minimize the required resources in transmitting a pre-recorded VBR media stream. In [7], we presented a linear-time traffic smoothing algorithm based on the Lazy scheme (L-scheme) and the Aggressive scheme (A-scheme). By applying the L-scheme, we can decide the minimum client buffer and delay time required to transmit the VBR media by the allocated network bandwidth. Then, the A-scheme is applied to minimize the idle rate of the allocated bandwidth under the available client buffer. We have shown that the optimal resource requirements can be decided in O(n) time (n is the number of video frames) [7]. However, as n is usually very large (n = 216000 for a two-hours MPEG-1 movie), this QoS control and resource management procedure is not suitable for online computation. To facilitate
resource management and QoS control, we need to explore the relations among the required resources.

Recently, some approaches have been presented to off-line compute the optimal tradeoffs among different resources [16-17]. They presented a good QoS control and resource management procedure to provide the flexibility in determining a suitable network bandwidth for the available buffer size. 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. Given a pre-recorded VBR media, a native algorithm requires O($n^3$) time complexity to compute the optimal bandwidth-buffer tradeoff. It is really time-consuming. In [16], given a media stream, we presented an O($n\log n$) algorithm to characterize this bandwidth-buffer tradeoff under the minimum delay time and the minimum bandwidth idle rate. This function depends only on the considered media stream and can be applied to the transmission from any server to any client. The QoS control procedure takes only O(1). However, this solution method does not consider the network model with additional relay-server.

In this paper, we extend the problem model to consider additional relay-servers. In each relay-server, there are an incoming-transmission schedule (in-schedule, for short) and an outcoming-transmission schedule (out-schedule, for short). Given a media stream, our proposed algorithm can compute the optimal bandwidth-buffer tradeoff to transmit the VBR media on any transmission paths with additional relay-server. Based on the pre-computed tradeoff functions, a good QoS control procedure can be designed to allocate the suitable bandwidth for the available buffer size in each relay-server. It is different from the on-line computation approach which requires O($n$) computation time in each relay server to make the admission control. Notably, the pre-computed tradeoff functions can be applied to any transmission paths allocated. The remainder of this paper is organized as follows. The proposed network model and the related system formulation are described in Section 2 with some primary definitions of problem parameters. Fundamental limits and tradeoffs for providing guaranteed QoS control to VBR media transmission are presented. In Section 3, some observations in the variations of required buffer size for the increasing of transmission bandwidth are presented. Notably, we should consider a set of possible optimal transmission schedules and select the best one. Based on these observations, an O($n\log n$) algorithm is presented in Section 4. Concluding remarks are given in section 5.

2. System Model and Problem Definition
The physical layout of our considered system architecture is shown in Fig. 1(a). It is a general multimedia network with additional relay-servers between the client and the content-servers (servers, for short) with a special application-server [24]. The application-server contains the complete information of the available resources in the system. As the operation steps shown in DAVIC [24], client’s request should be sent to the application-server to decide a suitable transmission path (in sequence, the suitable content-server, relay-servers and the client). The best transmission path is defined to support as many requests as possible. Give the available buffers, the allocated bandwidths should be minimized. If we can specify the optimal bandwidth-buffer tradeoff for each media stream, a good transmission path can be easily decided in a linear time. Fig. 1(b) shows a simple example with possible transmission paths to define the proposed problem. Notably, the tradeoff functions depend only on the media stream and can be applied to different transmission paths. Some basic problem parameters are defined as follows.

\[ P_i : \] the \( i \)-th network node (client, server, or relay-server) in the possible transmission path.

\[ T_i : \] the transmission schedule of incoming media data for the \( i \)-th network node \( P_i \).

\[ b_i : \] the available buffer size in \( P_i \). It is specified to compute a suitable transmission schedule \( T_i \).

\[ r_i : \] the network bandwidth allocated for \( T_i \).

---

**Fig. 1.** (a) The physical layout of the considered system contains the client, the content server, the application-server and relay-servers. (b) We can simply consider a possible transmission path from server to client.
As $T_i$ contains the cumulative data size transmitted, it can be easily proved that $T_i$ is a non-decreasing function of time $t$. Besides, as the incoming data must be always ahead of the outcoming data, the curve of an in-schedule should not be lower than that of the related out-schedule.

Assume that the transmission path is $P_m, P_{m-1}, ..., P_2, P_1$ from server $P_m$ to client $P_1$. $m$ is the maximum number of network nodes in the possible transmission path. The pre-recorded VBR media $V = \{f_0, f_1, ..., f_{n-1}\}$ ($n$ is the frame number and $f_i$ is the related frame size) is assumed to be stored on the server $P_m$ based on some data layout schemes [6,20]. When a request is accepted for serving, the related media data can be successfully retrieved from the storage system to the server buffer at the proper time [5-6,20-21]. Then, the transmission schedule $T_{m,i}$ is applied to transmit the media data from the server buffer to the relay-server $P_{m,i}$. According to the transmission schedule $T_{i}$, media data in the buffer of $P_{i+1}$ are transmitted to $P_i$. At last, media data are transmitted to the client $P_f$. The client frame-by-frame consumes media data in the client buffer for continuous playback.

In this paper, we focus our problems on QoS control and resource management in network transmission. The media data are transmitted into the network node $P_i$ by the in-schedule $T_i$ and transmitted out by the out-schedule $T_{i-1}$. In each relay server $P_{i}$, the in-schedule $T_i$ is decided by the available resources (such as the buffer size $b_i$) and the out-schedule $T_{i-1}$ specified. Notably, out-schedule $T_{i-1}$ of $P_i$ is just the in-schedule of $P_{i-1}$. This hierarchical schedule model is proposed for our next-generation multimedia systems [4-5]. Based on this system model, we can utilize the memory buffers in relay-servers to further smooth the VBR traffic and admit more requests. Besides, this model has a good property in system scalability and fault-tolerance [19]. More performance and resource analysis for scalable multimedia are shown in [19]. The decreasing of required bandwidth in additional relay-servers could be proved as follows.

- By applying the minimum-bandwidth transmission schedule algorithm [7-15], we can guarantee that: $r_i < r_{i-1}$ for all $i$. The required bandwidth in the relay server can be reduced.

  **proof:** If $b_i \to \infty$, we have $r_i \to 0$. If $b_i \to 0$, there is at least a transmission schedule $T_i$ satisfying $T_i \geq T_{i-1}$ and $r_i > r_{i-1}$. Thus, $r_i$ is between 0 and $r_{i-1}$. We have $r_i < r_{i-1}$ for all $i$.

Assume that the media stream starts the playback at time 0. Given a pre-recorded VBR media $V$, the cumulative-playback-function (CPF) $F(t)$ for the time $t$ is defined as

\[
F(t) = F(t-1) + f_i \quad \text{and} \quad F(-1) = 0.
\]
In this paper, we define $T_0 = F$. Thus, the client can be simply viewed as a special relay-server with the pre-specified out-schedule $F$ to decide the in-schedule $T_1$. Notably, there are $m-1$ transmission schedules ($T_1, T_2, \ldots, T_{m-1}$) should be considered for QoS control and resource management. Our goals are defined as the following two problems:

- **Input**: a pre-recorded VBR media $V$ and the bounded buffers ($b_1, b_2, \ldots, b_{m-1}$)
- **Output**: transmission schedules $T_1, T_2, \ldots, T_{m-1}$ to minimize the related network bandwidths $r_1, r_2, \ldots, r_{m-1}$.

**Input**: a pre-recorded VBR media $V$

- **Output**: the optimal bandwidth-buffer tradeoff in the $i$-th relay-server ($i = 1$ to $m$).

In this paper, we simple consider a system with $m = 3$. $P_1$ is the client. $P_3$ is the content server. The relay server $P_2$ is just the head-end (or called subscriber) for a VOD system. Different from the conventional problem models [7-17], we should consider the utilization of the relay-server buffers to further reduce the required bandwidth and support more requests.

**Fig. 2.** In relay-server $P_1$, given the buffer constraint, there are lots of out-schedules (i.e. LA, LAL, and S-LA) have the same optimal bandwidth requirement. We want decide the best out-schedule to minimize the incoming bandwidth allocated for $P_1$.

In our previous paper [7], given the CPF $T_0 = F$, we can construct an optimal in-schedule $T_1$ to minimize the delay-time $d_1$, the incoming bandwidth $r_1$ allocated and the related bandwidth idle rate $u_1$ under the given buffer constraint $b_1$. As the obtained transmission schedule can optimize both the resource allocation and utilization, it is called an *optimal* transmission schedule. In the relay-server $P_2$, if a specific out-schedule $T_1$ is presented, the optimal in-schedule $T_2$ can be simply computed by viewing this $T_1$ as
a kind of CPF for $P_2$ (just like the CPF $T_0$ for $P_1$). However, in a relay server, the related optimal transmission schedules (incoming or outcoming) are generally not unique. As shown in Fig. 2, given a buffer size, there are at least three optimal transmission schedules (LA, LAL (LA-Lazy), and S-LA (smoothed-LA) [7]). These optimal transmission schedules have the same bandwidth requirement and the same bandwidth idle rate.

In conventional approaches [7-17], the computation of the in-schedule would depend not only on the buffer size specified but also on the related out-schedule. Different out-schedules would lead to different in-schedules and require different resources. Although we can compute the optimal in-schedule for each given out-schedule, there are infinite possible out-schedules should be considered. Thus, the best in-schedule to minimize the bandwidth allocated for these out-schedules would not be trivially decided by conventional approaches [7-15]. Although we have presented an algorithm to explore the optimal bandwidth-buffer tradeoff between two network nodes [16], this method is valid only for a specific out-schedule. Without losing the generality, we focus on the transmission problems for $P_2$ in this paper. Our proposed method can compute the optimal transmission schedule and bandwidth-buffer tradeoff. The same idea can be extended to the $i$-th relay server.

3. Optimal Bandwidth-Buffer Tradeoff between Relay-Servers

In relay-server $P_1$, we define the optimal in-schedules obtained by the LA algorithm and the LAL algorithm [7] as $T_{La}$ and $T_{LAL}$. All these optimal in-schedules $T_i$ require the same delay time $d_i$, the same network bandwidth $r_i$ and the same bandwidth idle rate $u_i$. Notably, these in-schedules for $P_1$ are the possible optimal out-schedules for $P_2$. By L-scheme and A-scheme [7], we can prove that:

- Given the buffer size $b_i$, for any optimal in-schedule $T_i$, $T_{La}(t) \geq T_i(t) \geq T_{LAL}(t)$ for any time $t$.
- All these optimal in-schedules $T_i$ start the connection at the same start-connection-time $s_i$ and end the connection at the same end-connection-time $e_i$.

We want to select one of these optimal in-schedules of $P_1$ as the best out-schedules of $P_2$ to minimize the required bandwidth $r_2$. Thus, the multimedia system can support as many requests as possible.

3.1. Minimize Required Bandwidth and Buffer in a Relay-Server

We can easily prove that all the optimal transmission schedules $T_i$ (in-schedules for
$P_1$ and out-schedules for $P_2$ are bounded by $T^{LA}$ and $T^{LAL}$. Assume that an in-schedule $T_2$ is specified. We can easily decide the related out-schedule $T_1$ from the optimal solution region (between $T^{LA}$ and $T^{LAL}$) to obtain the minimum buffer requirement $b_2$.

$$T_1(t) = \min\{ T_2(t), T^{LA}(t) \}$$

$$b_2 = \max\{ T_2(t) - T_1(t); \text{for all } t \}$$

Fig. 3 shows a simple example to demonstrate these relations. Notably, the buffered data size $T_2(t) - T_1(t) > 0$ only when $T_2(t) > T^{LA}(t)$ and $T_1(t) = T^{LA}(t)$. We can rewrite the buffer equation as

$$b_2 = \max\{ T_2(t) - T^{LA}(t); \text{for all } t \}$$

As the initial value $T_2(e_1) - T^{LA}(e_1) = 0$, $e_1$ is the end-connection time of $T_1$, we can prove $b_2 \geq 0$. We can always find out an optimal transmission schedule $T_1$ to let $b_2$ be dependent on $T^{LA}$ and $T_2$. As the values of $T^{LA}(t)$ are specified, the required buffer $b_2$ is dependent on $T_2(t)$. Given $b_2$, we want to find an optimal in-schedule $T_2$ between $T^{LA} + b_2$ and $T^{LAL}$ as shown in Fig. 3(a).

To minimize the required buffer and bandwidth, the value of $T_2(t)$ should be as small as possible under the bounded constraint $T_2(t) \geq T^{LAL}(t)$ and the initial value $T_2(e_1) = T^{LAL}(e_1) = T^{LA}(e_1) = T_1(e_1) = |V|$. We can easily compute the optimal transmission schedule by the following O($n$) algorithm.

**Algorithm: the optimal in-schedule $T_2$ and the optimal out-schedule $T_1$**

1. Pre-compute the schedules $T^{LAL}$ and $T^{LA}$. Initialize $T_2(e_1) = |V|$.
2. Compute $T_2(t) = \max\{ T_2(t+1) - r_2, T^{LAL}(t) \}$ for all $t$.
3. Compute $T_1(t) = \min\{ \max\{ T_2(t+1) - r_2, T^{LAL}(t) \}, T^{LA}(t) \}$ for all $t$.

Based on this algorithm, we can rewrite the buffer size $b_2 = \max\{ \max\{ T_2(t+1) - r_2, T^{LAL}(t) \} - T^{LA}(t); \text{for all } t \} = \max\{ T_2(t+1) - r_2 - T^{LA}(t); \text{for all } t \}$.

By following the proposed linear-time algorithm, we can analyze the boundary cases with very small and very large available buffer. When the available buffer $b_2$ is close to 0, the data incoming-transmitted should be outcoming-transmitted immediately. The incoming bandwidth $r_2$ would be close to the outgoing bandwidth $r_2$. On the other hand, if the available buffer $b_2$ is the same as the media size $|V|$, we can apply the well-known stored-and-forward scheme. The required bandwidth would be very low. The optimal bandwidth-buffer tradeoff function is bounded in the required buffer.
Fig. 3. Assume that an in-schedule is specified. We can easily construct an out-schedule to decide the minimum buffer. Notably, the buffer would depend only on the out-schedule obtained by LA.

3.2. Buffer-Points on $T^{LA}$ and Segment-Points on $T^{LAL}$

We have shown that the buffered data size in $P_2$ would depend on $T^{LA}$ and the in-schedule $T_2$. As we know, $T^{LA}$ is an on-off function with interleaved on-transmission segments (on-segment, for short) and off-transmission segments (off-segment, for short) shown in Fig. 4(a). Notably, there are at most $n$ on-segments and $n+1$ off-segments. From the triangular formula, we can find that the start point of each on-segment in $T^{LA}$ would have the maximum buffer requirement. These points (the start points of on-segments in $T^{LA}$) are called the buffer-points. In Fig. 4, we mark each buffer-point by a "box". We can simply consider the changes of bandwidth-buffer tradeoff on buffer-points of $T^{LA}$.

Considering the in-schedule $T_2$, there would be only one on-segment if the buffer size $b_2 \rightarrow |V|$. By decreasing the available buffer size, the off-segments are introduced and started at the points called the segment-points. From the definition of $T_2$, they are just the start points of the off-segments on $T^{LAL}$. As shown in Fig. 4 we mark each segment-point by a "circle". Notably, the maximum buffer requirement during transmission would be happened at one of these buffer-points. Besides, only at these segment-points, the constructed transmission schedule would be separated into different on-segments.

It can be easily found that there are at most $n$ buffer-points and $n$ segment-points. In each on-segment of $T_2$, define the buffer-point of $T^{LA}$ which has the maximum buffered data in this on-segment as the related segment-buffer-point. Notably, this segment-buffer-point must be one of these buffer-points in the related on-segment. The segment-buffer-point that achieves the maximum buffer requirement $b_2$ is called the maximum-buffer-point or the schedule-buffer-point for the schedule $T_2$. The on-segment with the
maximum-buffer-point is called the maximum buffer segment.

Now, we consider the maximum buffer segment in $T_2$ and try to decrease the available buffer size from $b$ to $b'$ as shown in Fig. 4(b). Assume that the maximum-buffer-point is not changed and no new off-segment is created. We can find that

- The required bandwidth is linearly increased when the available buffer is linearly decreased.

**Proof:** As shown in Fig. 4(b), $r$ and $r'$ are the required bandwidth rates for the available buffer sizes $b$ and $b'$, respectively. From the triangular formula, we have $(r' - r) = - (b' - b) / k$ where $k$ is the difference between the maximum-buffer-point and the end-transmission point in the related on-segment. As $k$ is a constant value, the required bandwidth is linearly increased when the available buffer is linearly decreased.

As shown in Fig. 4(b), in some a range of the available buffer sizes (i.e. $[b', b]$), the required bandwidth is increased by a constant slope $k$ when the available buffer is increased. This linear tradeoff slope is called the buffer-decreasing-slope (or the bandwidth-increasing-slope).

Notably, the same results can be applied to the segment-buffer-points in other on-segments. Thus, although the maximum-buffer-point may switch to another segment (or shift to another index point), the available buffer is also linearly increased and continuously changed when the required bandwidth is linearly decreased. Notably, as the applied buffer-decreasing-slope may be changed (i.e. the maximum-buffer-point is changed), the function would be piecewise-linear. The optimal bandwidth-buffer tradeoff is piecewise-linear and continuously decreasing.

**Fig. 4.** From our definitions, the start point of each on-segment in $T_{LA}$ (marked by a "box") would have the maximum buffer requirement. Besides, the obtained off-segments would be happened only at the off-segments in $T_{LAL}$ (marked by "circles").

### 3.3. Segment-Separating-Rates and Equal-Buffer-Rates
To formulate the optimal bandwidth-buffer tradeoff, we should identify the bandwidth rates at which the related buffer-decreasing-slope would be changed. Besides, as the maximum-buffer-point may switch to any other on-segments, we should keep track the changes of the bandwidth-buffer tradeoff in each on-segment. In this section, we extend the buffer-decreasing-slope concept $k = - \frac{(b' - b)}{(r' - r)}$ to each on-segment. From this extended definition of buffer-decreasing-slope $k$ (the difference between the segment-buffer-point and the end-transmission point in the related on-segment), the slope may change at one of the following two cases.

- The end-transmission point is changed.
- The segment-buffer-point is changed.

The first case may happen when the related on-segment is separated. Look into the case that a segment is separated as shown in Fig. 5. The obtained transmission schedule $T_2$ with the bandwidth rate $r'$ just touches $T^{LAL}$ at the index $i'$. When the bandwidth rate increases from $r'$ by a small value, a new off-segment is inserted with the start point $i'$. Such a bandwidth rate $r'$ is called a segment-separating-rate.

![Fig. 5. When the bandwidth rate increases from the segment-separating-rate $r'$ by a small value, a new off-segment would be inserted at the index $i'$. The original on-segment is separated into the right sub-segment and the left sub-segment. There are two possible cases should be considered.](image-url)
Notably, the original on-segment is separated into the right sub-segment and the left sub-segment. If the segment-buffer-point is at the right sub-segment (as shown in Fig. 5(a)), the related end-transmission point is not changed and the buffer-decreasing-slope would not change. If the segment-buffer-point is at the left sub-segment (as shown in Fig. 5(b)), the related end-transmission point is changed to \( i^r \). Thus, the buffer-decreasing-slope would be changed to \( k^r \). We can find that \( k^r < k \). In Fig. 5, the related changes of the bandwidth-buffer tradeoff in each on-segment are also shown. Notably, when the bandwidth rate is increased, segments may be separated further. The number of on-segments is increased from 1 to \( n \) when the available buffer is decreased from \(|V|\) to 0. The new added on-segment will introduce a new bandwidth-buffer tradeoff as shown in Fig. 5. We select the maximum buffer requirement as the available buffer size.

If the on-segment is not separated, we should consider the second case with the changed segment-buffer-point. This case is happened at the equal-buffer-rate as shown in Fig. 6. The equal-buffer-rate \( r^e \) is defined as the slope of the line segment from \( T^{LA}(i) \) to \( T^{LA}(j) \). The index \( i \) and \( j \) are two different buffer-points defined in the section 3.2. Assume that these two time points are at the same on-segment for the transmission schedule \( T_2 \) with the bandwidth rate \( r^e \). There is a parallelogram between the transmission schedule \( T_2 \) and the line segment from \( T^{LA}(i) \) to \( T^{LA}(j) \). From this parallelogram, we can easily prove that the buffered data size at time \( i^r \) is the same as the buffered data size at time \( j^r \). Thus, we call \( r^e \) as an equal-buffer-rate.

If the bandwidth rate is slightly smaller than \( r^e \), the buffered data size at time \( i^r \) would be larger than that at time \( j^r \). Assume that \( i^r \) is just the segment-buffer-point for the related on-segment (as shown in Fig. 6). When the bandwidth rate is slightly larger than \( r^e \), the segment-buffer-point would be changed from \( i^r \) to \( j^r \). Thus, the buffer-decreasing-slope is changed from \( k \) to \( k^r \). It can be easily proved that the value of buffer-decreasing-slope would be decreased (\( k^r < k \)). We have a decreasing buffer-decreasing-slope for the optimal bandwidth-buffer tradeoff.
4. Optimal Bandwidth-Buffer Tradeoff

Notably, in the above section, we consider only the bandwidth-buffer tradeoff in each on-segment. However, the maximum-buffer-point may also shift from one on-segment to another on-segment. This case should be handled by comng the bandwidth-buffer tradeoff in each on-segment to find the maximum buffer requirement. It is the basic idea of our proposed algorithm. Before describing our proposed $O(n \log n)$ algorithm, we first identify all the segment-separating-rates and the related equal-buffer-rates in each on-segment by a linear-time procedure. Then, a construction algorithm of the optimal bandwidth-buffer tradeoff function is proposed. Based on this tradeoff function, an $O(1)$ QoS control procedure can be designed to allocate the minimum network bandwidth for the available buffer size in the relay-server [16].

4.1. Identify Segment-Separating-Rates and Related Equal-Buffer-Rates

As we know, both $T^{LA}$ and $T^{LAL}$ can be represented by a set of on-segments and off-segments. The start point of an on-segment is an inner-corner of the transmission schedule. An outer-corner of the transmission schedule is the start point of an off-segment. By increasing the bandwidth rate from 0 to $\infty$, a linear-time algorithm with an $O(n)$ heap structure is proposed to exploit all the segment-separating-rates. It is similar to construct the convex upper envelope of the outer-corners in $T^{LAL}$ as shown in Fig. 7(a). All these outer-corners are under the convex upper envelope to represent these on-segments by a tree structure. Based on this tree structure, we can identify the related equal-buffer-rates
in each on-segment. As shown in Fig. 7(b), it is similar to hierarchically construct the the convex lower envelope of the inner-corners in $T^{IA}$ based on the tree structure of on-segments.

In this paper, we denote the outer-corners of $T^{IAL}$ by $c^O_k$ for $k = 1$ to $p$. The inner-corners of $T^{IA}$ is denoted by $c^I_k$ for $k = 1$ to $q$. Notably, $p \leq n$ and $q \leq n$.

To construct a tree structure of all these on-segments, we first define the angle that counterclockwises from a line segment to its end point $x$ (line $xx$) as $p$. The step-by-step description of the proposed construction algorithm is shown as follows.

**Algorithm: Segment-Separating-Rates**

1. We first push the corner point $c^O_p$ to the heap twice as the line $c^O_p c^O_p$.
2. Assume that the convex upper envelope from $c^O_k$ to $c^O_p$ is already constructed. Now, we want to construct the convex upper envelope from $c^O_{k-1}$ to $c^O_p$.
3. Pop the corner point $c^O_x$ from the heap. Check whether the angle that counterclockwises from line $c^O_{k-1}c^O_k$ to line $c^O_kc^O_x$ is less than or equal to $p$.
4. If the angle is less than or equal to $p$:
   4.1. The constructed convex upper envelope from $c^O_k$ to $c^O_p$ together with the new segment $c^O_{k-1}c^O_k$ is the resulted convex upper envelope from $c^O_{k-1}$ to $c^O_p$.
   4.2. Push the corner points $c^O_x$ and $c^O_k$ to the heap.
5. If the angle is larger than $p$:
   5.1. We let the index $y = x$ and pop the next corner point in the heap as $c^O_x$ sequentially. Test the angle that counterclockwises from $c^O_{k-1}c^O_y$ to $c^O_y c^O_x$ until the angle is less than or equal to $p$.
   5.2. The constructed convex upper envelope of the staircase from $c^O_k$ to $c^O_p$ together with the new segment $c^O_{k-1}c^O_y$ is the resulted convex upper envelope from $c^O_{k-1}$ to $c^O_p$.
   5.3. Push the corner points $c^O_x$ and $c^O_y$ to the heap.
6. $k = k - 1$. Go to step (2).

The relation between an on-segment and its two separated sub-segments can be intuitively represented by a binary tree structure as shown in Fig. 7(a). In each tree node, the value of segment-separating-rate and the index of the related separating point are stored.

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. The
A binary tree is a heap structure on the values of separating-rates. It is called the segment-separating-tree.

**Fig. 7.** (a) Construct the segment-separating-tree (the convex upper envelope of the outer-corners in $T^{LAL}$). (b) Based on the segment-separating-tree to construct the equal-buffer-rates in each separating segments (the convex lower envelope of the inner-corners in $T^{LA}$).

By tracking the tree structure of the segment-separating-rates, we want to identify all the equal-buffer-rates used in the related on-segment as shown in Fig. 8. Notably, in an on-segment, assume that the segment-buffer-point is at point $c_i$. We can prove that

- The inner-corner $c'_i$ must be a vertex of the related convex lower envelope for this on-segment.

**Proof:** If $c'_i$ is not a vertex of the related convex lower envelope, there would have a line segment $c'_j c'_k$ in the convex lower envelope such that $c'_j < c'_i < c'_k$. If the bandwidth rate is larger than the slope of line $c'_j c'_k$, the buffer size at $c'_k$ is larger than that at $c'_i$. That is a contraction to that $c'_i$ is the segment-buffer-point. The same conclusion can be obtained when the bandwidth rate is smaller than the slope of line $c'_j c'_k$.

Let $c'_j c'_k$ be a line segment of the convex lower envelope. When the bandwidth rate increases to the slope of $c'_j c'_k$, the buffer size at the segment-buffer-point $c'_i$ is the same as that at $c'_k$. The segment-buffer-point in this on-segment then changes from $c'_i$ to $c'_k$. To explore these equal-buffer-rates, we construct the convex lower envelopes of equal-buffer segments in $T^{LA}$ from the leaf to the root of the related segment-separating-tree. The
proposed algorithm is really similar to the identification algorithm of segment-separating-rates.

**Algorithm: Equal-Buffer-Rates**

1. Select an on-segment in the segment-separating-tree (assume that the convex lower envelopes for its sub-segments are already constructed). Now, we want to construct the convex lower envelopes for this on-segment.
2. Construct $c'_a c'_b$ as the tangent line segment of the convex lower envelopes for these two sub-segments. The slope of $c'_a c'_b$ is the related equal-buffer-rate.
3. The convex lower envelope of this on-segment is just the part of convex lower envelope of its left sub-segment (ended at $c'_a$), the line segment $c'_a c'_b$ and the part of the convex lower envelope of its right sub-segment (started at $c'_b$).

Notably, the related equal-buffer-rates should be larger than the segment-separating-rate of this on-segment, and smaller than the segment-separating-rate of its mother segment. Otherwise, it would not be necessary to be considered in constructing the optimal bandwidth-buffer tradeoff. From the definition of $T_1$, the largest buffer size of an on-segment is 0 if the entire segment is under $T^{LA}$. Thus, the related segment-separating-rate is not necessary to be considered in constructing the optimal bandwidth-buffer tradeoff. We can delete these un-necessary rates to reduce the computation time.

![Diagram of the optimal bandwidth-buffer tradeoff](image)

**Fig. 8. The construction of the optimal bandwidth-buffer tradeoff.**

### 4.2. Construct Optimal Bandwidth-Buffer Tradeoff

With the data structures for the segment-separating-rates and the related equal-buffer-rates, we can maintain the structure of separating on-segments and the related segment-buffer-points/sizes as follows. We process the stored segment-separating-rates in
a decreasing order (from root to leaf). Between two segment-separating-rates, the related equal-buffer-rates are considered. The processing steps are as following.

**Algorithm: Optimal Bandwidth-Buffer Tradeoff**

1. If the selected rate is a segment-separating-rate, the changes of the decreasing largest buffer sizes are as shown in Fig. 5. A new on-segment is introduced and the related buffer-decreasing-slope may be changed.

2. If the selected rate is an equal-buffer-rate, we can just change the segment-buffer-point. Fig. 6 is the related changes of the decreasing largest buffer sizes. Note that the related buffer-decreasing-slope of the largest buffer size is changed.

3. The maximum buffer size is just the upper envelope of these largest buffer sizes for different on-segments. We maintain the optimal bandwidth-buffer tradeoff by finding a new upper envelope from the original upper envelope and the new added on-segment with the decreased buffer-decreasing-slopes as shown in Fig. 8.

Notably, when a rate (segment-separating-rate or equal-buffer-rate) is selected for processing, one or two lines are inserted to the original upper envelope. These lines represent the possible changes of buffer-decreasing-slope for the bandwidth-buffer tradeoff. By finding the intersection points of the added lines to the original upper envelope, we can construct the new upper envelope in \(O(\log n)\) time [23]. Since there are \(O(n)\) such rates should be considered, the time complexity of the proposed algorithm is \(O(n\log n)\). We can extend the same idea to the \(i\)-th relay server to compute the related tradeoff functions for QoS control and resource management.

5. Concluding Remarks

In this paper, an algorithm is proposed to decide the optimal bandwidth-buffer tradeoff for a general multimedia network with additional relay-server. This model is good for scalable multimedia and fault-tolerance to support more requests. Based on the pre-computed tradeoff functions, the QoS control and resource management procedure for the server can be as simple as a table look-up with a constant time complexity. Given buffer size in each relay-server, we can allocate the most suitable network bandwidth to transmit the VBR media. Besides, this approach also shows great flexibility to allow various clients and relay servers to set up their best transmission schedules. As the required initial delay depends only on the transmission rate and the end-point of the first on-segment, we can easily apply the same idea to decide the optimal bandwidth-delay
tradeoff. Our future work is to extend the proposed method to a world-wide network system with heterogeneous computers and multiple relay-servers.

**References**


