Online Traffic Smoothing for Delivery of VBR Media Streams

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

Institute of Information Science, Academia Sinica

NanKang, Taipei, Taiwan

Abstract[*] - Traffic smoothing for delivery of online VBR media streams is one of the most important problems in designing multimedia systems. Given an available client buffer, a window size and a window-sliding size, previous window-based online smoothing methods try to reduce peak bandwidth allocated in each window. However, as bandwidths allocated in different windows are minimized independently, these methods require a large peak bandwidth for transmitting entire stream. In this paper, a new window-based method is proposed. It introduces two new ideas, the dynamic window-sliding size and the aggressive work-ahead, for delivery of online VBR media streams. Our ADWS (aggressive and dynamic window-sliding) method can automatically decide the suitable window-sliding sizes for different windows. Thus, the allocated peak bandwidth can be further reduced. By examining various media streams, ADWS is shown to be effective and efficient. Considering the online transmission of movie *Star Wars* with a 90 KB client buffer, ADWS yields 13% less in peak bandwidth, compared with the best-known window-based online smoothing algorithm SLWIN(1). Its computation cost (the window-sliding number) is 75% of that required by SLWIN(1).

[*] *This work was partially supported by NSC under grants NSC88-2213-E-001-011, NSC88-2213-E-001-012, and NSC88-2213-E-001-025. A primitive version of this work appeared in IEEE INFOCOM'99, NY, U.S.A.*

I. Introduction

In a distributed multimedia system, VBR media streams need to be smoothly transmitted from servers (across network) to clients by following transmission schedules. To guarantee media QoS (quality-of-service), a transmission schedule must satisfy real-time requirements for jitter-free playback. Additionally, allocated resources (such as network bandwidth and memory buffer) should be minimized and fully utilized to support as many requests as possible. A network transmission schedule can be categorized as either client-controlled [1] or server-controlled [2-9]. In a client-controlled scheme, each client sends feedback messages to servers by monitoring the playback status (such as the occupancy of client buffer) to adjust servers' transmission rates. It is simple in implementation. However, as network is possibly congested, client-controlled schemes have drawbacks in large response overhead. To guarantee deterministic network services, server-controlled schemes have received great attention recently for delivery of VBR media streams. In general, media data can be pre-recorded or online-generated. As shown in [2-15, 22-24, 28], if media data is pre-recorded, the entire traffic of media stream can be accurately characterized offline to minimize specified cost functions (such as peak bandwidth or bandwidth variability). However, if media streams are generated online, offline smoothing methods that require the complete input traffic will not be applicable [16, 21]. Although some prediction methods have been presented to estimate the sizes of future frames based on their precedent frames [19-20], the predicted frame sizes may not be correct and, therefore, the quality of service for media playback may not be guaranteed.

In reality, media systems may tolerate an acceptable playback delay in exchange for a smaller bandwidth. It motivates the design of window-based online smoothing methods [21]. Let the server start its media transmission only after a suitable delay time W. At any time point, media data already generated but not transmitted yet are stored in server buffer. These data can be viewed as in a time

window (*W* is the window size) just preceding the time point. As the frame sizes of media data in this window are known, we can use any smoothing algorithm designed for delivery of pre-recorded media to this window. In [21], a variation of the smoothing algorithm proposed in [5] is applied. It is called SLWIN(*k*) where the given constant *k* (called *window-sliding size*) is the period for executing traffic smoothing operations. Note that different traffic smoothing methods [2-9] have concluded that the peak bandwidth requirement can be highly condensed if the burst traffic is pre-transmitted as early as possible before the client buffer is overflowed. However, in the previous static window-sliding methods such as SLWIN(*k*) [21], transmission start times are periodic for traffic in different windows. Therefore, the traffic already generated (in the next window) cannot be pre-transmitted immediately even traffic in the current window is transmitted out. Although bandwidths allocated in different windows are minimized individually, these static window-sliding methods do not lead to the minimization of peak bandwidth allocated for transmitting the entire stream. As experiments shown in [21], if the window-sliding size is large, the peak bandwidth required will be large. On the other hand, if the window-sliding size is small, the computation will be time consuming. For example, although SLWIN(*W*) uses only *n/W* window-sliding operations (called *window-sliding number*) where *n* is the number of input frames, it requires a large peak bandwidth. By highly overlapping the smoothing windows, SLWIN(1) can reduce the required peak bandwidth. However, its window-sliding number is *n*. It is hard to decide the best window-sliding size *k* for SLWIN(*k*).

To alleviate these drawbacks, this paper proposes a new window-based online traffic smoothing method called ADWS (aggressive and dynamic window-sliding). ADWS applies the aggressive work-ahead scheme and the dynamic window-sliding size that follows a dependent manner to reduce the peak bandwidth allocated in consecutive windows. In each window, the peak bandwidth allocated in the precedent windows is used as the initial bandwidth for traffic smoothing. The burst traffic could be pre-transmitted as early as possible to condense the peak bandwidth requirement. In addition, the

window-sliding size for each window can be dynamically adjusted to minimize the window-sliding number. It is not necessary to pre-assign the window-sliding size. In the remainder of this paper, we describe terminology and problem definition in Section II. Then, the ADWS algorithm is introduced in Section III. In Section IV, we have examined the ADWS method on several benchmark streams. Experiments show that ADWS is effective and efficient for various patterns of burst traffic. Conclusion is presented in Section V.

II. Terminology and Problem Definition

At client sites, media frames are played periodically. Define a media stream as $V = \{ f_0 f_1 \ldots f_{n-1}; T_f\}$ where f_i is the i-th media frame (it also represents the frame size), T_f is the period for playing frames and n is the number of frames in this media stream. Without loss of generality, we can assign the period $T_f = 1$ throughout this paper. The playback schedule can be described by cumulative media frame sizes called *cumulative playback function (CPF)* [2-3]. Assume that the client starts playback at time 0. The i-th cumulative media size can be computed by $F_i = F_{i-1} + f_i$ for $i = 0, 1, \ldots, n\text{-}1$, and $F_x = 0$ for all $x < 0$. The CPF $F(t)$ is shown as follows:

$$F(t) = CPF(t) = \begin{cases} 0 & t < 0; \\ F_i & i \le t < (i+1); \text{ for } i = 0, 1, \ldots, n-1; \\ F_{n-1} & n \le t. \end{cases} \quad (1)$$

It is a stair function with jumps at time $t = i$, for the integer $i = 0, 1, \ldots, n\text{-}1$, where $F(t)^-$ and $F(t)^+$ denote the lower and the upper corner values. Let r_i denotes the bandwidth applied to transmit the media stream between time i and time $(i+1)$. Based on the definition of CPF, we define the *cumulative transmission function* (CTF) $G(i) = G(i\text{-}1) + r_i$ as the amount of data sent by the server before time i. Notably, a jitter-free CTF should ensure that a complete data frame is received at the client before its display without buffer overflow and underflow. Let b denote the given client buffer

size. The cumulative transmitted data at time t (= $G(t)$) should not be larger than $H(t) = \mathbf{min}\{F_n, F(t) + b\}$. Additionally, $G(t)$ should not be smaller than $F(t)$ for continuous playback. As shown in Fig. 1, a jitter-free transmission schedule $G(t)$ should satisfy $F(t)^+ \leq G(t) \leq H(t)^-$ where d is the delay time of the transmission schedule; namely, the server starts transmission at time $-d$.

The performance of a transmission schedule is usually measured by the amount of resource required for serving jitter-free delivery. Given the available client buffer, we want to design a jitter-free transmission to minimize the peak bandwidth allocated. As the transmission schedule can be represented by a sequence of transmission rates, the peak bandwidth allocated for media transmission can be measured as follows.

$$\text{Peak bandwidth} = \mathbf{max}\{ r_i : \text{for all } i \}. \quad (2)$$

Without loss of generality, a smoothed transmission schedule G can be also viewed as a sequence of *transmission segments*. For each transmission segment $\langle G(i), G(j); r_i \rangle$, a line segment from the start-point $(i, G(i))$ to the end-point $(j, G(j))$ is presented. It denotes the operation of media transmission starts at time i and ends at time j with rate $r_i = (G(j) - G(i)) / (j - i)$ where $r_i = r_{i+1} = ... = r_{j-1}$. Note that, the end-point of a transmission segment is just the start-point of its next transmission segment.

As demonstrated in previous studies [2-9], large initial delay and client buffer can act as a good reservoir to regulate the difference between transmission and playback rates. However, the available buffer size and the acceptable delay time are bounded and highly dependent on the service provided. For a pre-recorded media stream, as all the frame sizes are known, the transmission schedule and resource tradeoff can be computed offline [2, 18]. However, for an online generated media stream, at any time t, we only know the sizes of frames generated before time t (they are f_0, f_1, ..., and f_t). The previous offline algorithm is not applicable for delivery of online generated media streams.

Recently, a window-based smoothing method called SLWIN(k) [21] was proposed to smooth online traffic where k is a given constant called *window-sliding size*. Let D be the acceptable initial delay for media playback on the client. We have $D = W + d$ where W is the *window size* and d is the pre-loading time. They are given constants. As shown in Fig. 2, assume that the client starts media playback at $t = 0$. The first frame f_0 is generated at time $-D$ and must be received by the client before $t = 0$ for playback. In the initial stage, the traffic smoothing algorithm is operated when the first W frames (f_0, f_1, ..., and f_{W-1}) are generated. Then, the server bases on the obtained smoothing schedule to start the media transmission at time $-d$. After d unit-times of pre-loading, the client receives frame f_0 and starts the playback at time 0. Note that, as the sizes of media frames in this window are known, the traffic smoothing algorithms designed for the pre-recorded streams (such as CBA [6], MVBA [5], LA [2] and CRTT [9]) can be directly applied. In SLWIN(k), the operation for traffic smoothing is periodically executed once k new frames are generated. At time $k-d$, new frames $f_{W+1}f_{W+2}$ ... f_{W+k} are generated and some media data of f_0f_1 … f_W are transmitted. The algorithm is executed again to smooth the remaining traffic of f_0f_1 ... f_W and the new frames $f_{W+1}f_{W+2}$ ... f_{W+k}. At any time i, frames f_x (for all $x < i+1$) are already transmitted, and frames f_y (for all $y > i+W$) are not generated yet. There are at most W frames in window $w_i = f_{i+1}f_{i+2}$ … f_{i+W} can be considered for smoothing. As the smoothing algorithm takes the same computation complexity to each window, the total computation cost is proportional to the window-sliding number. The SLWIN(k) method runs the traffic smoothing algorithm O(n/k) times. If $k = 1$, it requires O(n) window-sliding number. If $k = W$, it requires O(n/W) window-sliding number.

III. Proposed Approach

Consider a media stream which contains medium-rate, low-rate and high-rate traffic in the first

window $w^- = w_{x-W}$, the second window $w = w_x$ and the third window $w^+ = w_{x+W}$, respectively. The transmission schedule constructed by SLWIN(W) is shown in Fig. 3(a). As the peak bandwidth allocated in w is minimized without considering the bandwidth already allocated in w^-, it requires a large peak bandwidth to transmit the traffic in w^+. Note that, as shown in Fig. 3(b), we can achieve a better smoothing schedule (with smaller peak bandwidth) if the bandwidth allocated in w^- is applied in w and w^+. This example motivates us to design a new window-based online traffic smoothing method that follows a dependent manner to improve the required bandwidth in consecutive windows. (The smoothing result obtained can be found in Fig. 5(b).)

A. Aggressive Work-Ahead Scheme

Different from the previous approaches, our algorithm smoothes traffic in different windows aggressively and dependently. To clearly demonstrate the effectiveness of our aggressive work-ahead scheme, we first assign a fixed window-sliding size k to our algorithm. Given $k = W$, a comparison between our scheme and the previous SLWIN(k) scheme is shown in Fig. 4. Fig. 4(a) shows the transmission schedule obtained by SLWIN(k). It contains four transmission segments ($<G(0), G(W); r_0>$, $<G(W), G(x); r_W>$, $<G(x), G(2W); r_x>$ and $<G(2W), G(3W); r_{2W}>$). As SLWIN(k) minimizes the bandwidth allocated in its last transmission segment without considering the peak bandwidth r_W already pre-assigned, a small rate r_{2W} (where $r_{2W} << r_W$) is assigned and the network channel is occupied until $e_0 = 3W$ (the end time of traffic) without aggressive work-ahead. By testing our scheme with the same media stream and the same window-sliding size, the obtained result is shown in Fig. 4(b). Based on the aggressive work-ahead scheme, the peak bandwidth allocated in precedent windows is applied as an input parameter to smooth the traffic in the current smoothing window. As shown in Fig. 4(b), in the third window, the network channel is released at time e_1. We can prove that $e_1 < e_0$ (our scheme has a shorter connection time of network channel than SLWIN(k)).

✍ Given the same window-sliding size k to smooth the same online media stream, our scheme can achieve a shorter connection time of network channel than SLWIN(k).

Proof: (It can be easily proved by aggressive work-ahead.)

By extending the above example to consider a small window-sliding size $k < W$, we can prove that traffic smoothed by our scheme has smaller peak bandwidth than that smoothed by the conventional SLWIN(k) scheme. The proof is shown in Appendix.

B. Dynamic Window-Sliding Size

The conventional SLWIN(k) scheme uses a static window-sliding size k (where k ? W). It executes the traffic smoothing procedure to determine the transmission schedule periodically whenever k new frames are generated. Therefore, even the traffic data in the preceding window are already transmitted as shown in Fig. 4(b), the new generated frames cannot be smoothed and transmitted immediately. Although a small window-sliding size can lead to a small peak bandwidth requirement, the computation cost increases proportionally. It is hard to decide the suitable window-sliding size to minimize both the computation time and the peak bandwidth allocated. In this paper, a dynamic window-sliding size is applied. We can execute the traffic smoothing algorithm to the new generated frames in the next window immediately whenever the scheduled frames have already been transmitted as shown in Fig. 4(c). Based on our dynamic window-sliding scheme, the applied window is hopping (window-sliding size = W) for a near-CBR (constant bit rate) media stream. When a bursty traffic is present, the window-siding size is automatically decreased to make a better control. Thus, the required peak bandwidth can be further reduced. Since the window-sliding size selected by our scheme is at least 1, we can prove that the total window-sliding number required is much smaller than n -- the window-sliding number of SLWIN(1).

✍ Our scheme requires a smaller window-sliding number than the conventional SLWIN(1) scheme to smooth the same online media stream.

Proof: (As described above, the proof is trivial.)

Comparing with SLWIN(1), our scheme requires a smaller window-sliding number. If the same window-sliding number is applied with the given window-sliding size k, our scheme can achieve at most the same peak bandwidth as SLWIN(k). In Fig. 5(a), a simple example shows that SLWIN(W/2) requires five window-sliding operations to smooth the given media stream. By examining the same media stream, our scheme requires only four window-sliding operations. Additionally, our obtained peak bandwidth is smaller than that of SLWIN(W/2). The proposed scheme is shown effective and efficient.

C. Proposed Online Traffic Smoothing Algorithm

In this paper, two new ideas (the aggressive work-ahead scheme and the dynamic window-sliding size) are introduced to online traffic smoothing. Assume that media frames in the current window are $f_{i+1}f_{i+2}$ ... f_{i+W} where the first un-transmitted frame is f_j ($i+1 \le j \le i+W$); namely, $f_{i+1}f_{i+2}$ ... f_{j-1} are already transmitted in the precedent window. For example, in the first window w_0, we have $i = -d$ and the index of the first un-transmitted frame is $j = 0$. In this paper, without loss of generality, we assign $d = 1$. The pre-transmitted traffic size Q, also called the work-ahead (or backlog), can be computed as follows.

$$Q = F(j\text{-}1) - F(i) \qquad (3)$$

The start-point of the first transmission segment in this window is $(s, G(s)) = (i, G(i))$ where $G(i) = F(i) + Q = F(j\text{-}1)$. Based on the start-point $(s, G(s))$, we apply the traffic smoothing algorithm proposed in [5] with the aggressive work-ahead scheme to determinate the end-point $(e, G(e))$. The

transmission rate applied is $r_s = (G(e) - G(s)) / (e - s)$. By interpolation, $G(t) = G(s) + (t - s) * r_s$ for $t = s$ to e. Note that, for providing jitter-free playback, $G(t)$ should satisfy $F(t)^+ \leq G(t) \leq H(t)^-$ for all t. Define the highest test rate $R_H(t)$ and the lowest test rate $R_F(t)$ at time t as follows.

$$R_H(t) = (H(t)^- - G(s)) / (t - s)$$

$$R_F(t) = (F(t)^+ - G(s)) / (t - s) \quad (4)$$

We first initialize $t = j - 1$ and $t_H = t_F = j$ where f_j is the first un-transmitted frame. Then, increase the test end-point $t = t + 1$ to find the end-point $(e, G(e))$ where $e = t_H$ for $R_H(t_H) < R_F(t)$ and $e = t_F$ for $R_F(t_F) > R_H(t)$. Otherwise, we reset $t_H = t$ if $R_H(t_H) \leq R_H(t)$, and reset $t_F = t$ if $R_F(t_F) \geq R_F(t)$. The variables $R_H(t_H)$ and $R_F(t_F)$ represent the smallest $R_H(x)$ and the largest $R_F(x)$ for $x = s$ to t. The detailed description about our ADWS algorithm in each window $f_{i+1}f_{i+2} \ldots f_{i+W}$ (where the first un-transmitted frame is f_j, $i+1 \leq j \leq i+W$) is shown as follows.

ALGORITHM: ADWS for a Traffic Window

/* $f_{i+1}f_{i+2} \ldots f_{i+W}$ are in the current window. f_j ($i+1 \leq j \leq i+W$) is the first un-transmitted frame. */

/* Namely, $f_{i+1}f_{i+2} \ldots f_{j-1}$ are the pre-transmitted frames in the precedent window. */

/* The peak bandwidth allocated for the precedent windows is r_{max}. It is an input parameter. */

00: Initialize the start point $(s, G(s)) = (i, G(i))$ where $G(i) = F(j-1)$.

01: Initialize the test end-point at time $t = j - 1$ and $t_H = t_F = j$.

02: **repeat**

03: $t = t + 1$; /* Try the next test end-point where $T_f = 1$. */

04: **if** $(R_H(t_H) < R_F(t))$ **{** /* The segment is upper bounded by $H(t_H)$. */

05: $r_s = R_H(t_H)$; $e = t_H$; $G(e) = H(t_H)^-$;

06: output segment: $< G(s), G(e); r_s >$;

07: r_{max} = **max**{ r_{max}, r_s };

08: $s = t = e$; $t_H = t_F = s + 1$; /* Go to the next transmission segment. */

09: **} else if** ($R_F(t_F) > R_H(t)$) **{** /* The segment is lower bounded by $F(t_F)$ */

10: $r_s = R_F(t_F)$; $e = t_F$; $G(e) = F(t_F)^+$;

11: output segment: < $G(s)$, $G(e)$; r_s >;

12: r_{max} = **max**{ r_{max}, r_s };

13: $s = t = e$; $t_H = t_F = s + 1$; /* Start the next transmission segment. */

14: **} else if** (t ? $i+W$) **{** /* Not the last frame in the current window. */

15: **if (**($R_H(t_H)$? $R_H(t)$) **and** ($H(t_H)^- < F(i+W)$)) **{** $t_H = t$; **}**

16: **if** ($R_F(t_F)$? $R_F(t)$) **{** $t_F = t$; **}**

17: **}**

18: **until** ($t = i+W$) /* The last frame in the current window. */

19: r_s = **max**{ **min**{ r_{max}, $R_H(t_H)$ }, $R_F(t_F)$ }; /* Apply the *aggressive work-ahead* scheme. */

20: $e = s$ + ?($F(i+W)$-$G(s)$) / r_s ?; /* The window-sliding size is (e - i) > 0. */

21: $G(e) = F(i+W)$;

22: output segment: < $G(s)$, $G(e)$; r_s >;

23: r_{max} = max{ r_{max}, r_s };

24: $i = e$; $j = i + W + 1$; /* Start the next window. */

/* The next window will start with the start point (i, $G(i) = F(j - 1)$) and the test end-point $t = j$. */

As shown in step 19 of the ADWS algorithm, if and only if f_t is the last frame in the current window, we apply the aggressive work-ahead scheme to produce the last transmission segment with rate **max**{ **min**{ r_{max}, $R_H(t_H)$ }, $R_F(t_F)$ } where r_{max} is the peak bandwidth allocated. In step 20, we

calculate the dynamic window-sliding size. Note that, for each window considered in our scheme, we need to decide the dynamic window-sliding size required. This is an extra computational cost in addition to the proposed algorithm. However, as this extra cost is constant (O(1) as shown in step 20 of the ADWS algorithm), the total computation complexity of the ADWS algorithm is the same as that of the conventional SLWIN(k) algorithm. In this paper, we assume that the transmission quality of network channel is good. However, our algorithm can be easily extended to address the effect of the transmission loss of video data. Let p_{ij} be the j-th data packet of the i-th frame that is lost in transmission. The current traffic window considered is started from the x-th frame to the $(x+W)$-th frame. If $x < i$, we can easily add p_{ij} to the current traffic window for smoothing. The lost packet data p_{ij} will be transmitted and received before playing the i-th frame.

IV. Experiment Results

In this paper, we examine the proposed method on various benchmark streams [25-26]. The detailed statistics of these test streams are shown in Table I, which include the average bit-rate (AVG), maximum frame size (MFS) and standard deviation of frame size (STD). For each benchmark stream, we evaluate the proposed method by two different parameters: the peak bandwidth allocated and the required window-sliding number. Comparisons are made to the offline scheduler and the previous online schedulers [21]. The window size W used in experiments is the number of frames in a GoP (group of picture). Additionally, the related playback delay is $(W + 1)*T_f$ that is acceptable for real-world applications. The initial value of peak bandwidth is assigned to be zero. Note that, if the smoothing window starts with a large I-frame, the bandwidth required will be high. In this paper, as the applied window size is the GoP size, the second window for traffic smoothing has to be started with an I-frame. As suggested in [21], we extend the size of the first window size to W+1 to include

this I-frame. This adjustment is intuitive.

A. Comparisons to Previous Methods

To precisely demonstrate the effect of our ADWS method, a real-world movie stream *Star Wars* is first tested. A plot of the frame number versus the frame size with an overlaying average is shown in Fig. 6. The maximum frame size is 22.62 KB. The average rate is 0.36 Mbps. As shown in Fig. 7(a), this stream is an over two-hours-long MPEG-encoded movie with high-bursty VBR traffic. It has large frame size and high frame size variation as shown in many other real-world movie streams. Given an acceptable buffer size b = 90 KB for real-world applications, Fig. 7(b) shows that our required window-sliding number is only 75% of the window-sliding number required by SLWIN(1) – the best static window-sliding method. When comparing the required bandwidth, our ADWS method is over 13% smaller than SLWIN(1) (see Fig. 7(c)). Experiments show that, when transmitting a real-world high-bursty movie stream, our ADWS method can utilize the allocated bandwidth effectively and efficiently. The required bandwidth is smaller than that obtained by previous methods under the same initial delay and buffer size. On the other hand, the ADWS method has the same performance as previous methods for low buffer size. Comparing the required window-sliding number, ADWS is better than SLWIN(1). We have tried to implement a fast algorithm for SLWIN(1) [31]. However, the CPU time consumed is still very large. Although the SLWIN(W) method with a W-frame hopping-window could compute faster than ADWS, it has the drawbacks in requiring much higher bandwidth.

B. Smoothing Effect of High/Low Traffic Burst

The characteristic of burst is different due to the different video contents. Fig. 8(a) and Fig. 9(a) show the media streams of the *Advertisements* video stream and the *Lecture* video stream, respectively.

Both *Advertisements* and *Lecture* are encoded by the UCB software MPEG encoder [25]. By applying constant distortion coding, these two video streams have the same frames number and GoP size. Additionally, their average frame rates and variations in frame sizes are similar. These two media streams are different at the *scale of burst*. In *Lecture*, the same speaker and his slides are shown along with only zooming and panning. Since the contents of frames are similar, the traffic burst is low. However, in *Advertisements*, the frame contents change from one scene to another scene in a sequence of advertisements. The traffic burst of *Advertisements* is higher than that of *Lecture*. Note that, as the burst of *Advertisements* is presented at the early coming traffic, the peak bandwidth requirement for transmitting the stream can be identified at the early processing windows. The required bandwidth can be reduced by introducing a large delay. As shown in Fig. 8 (c), the required peak bandwidth is the same as that allocated by SLWIN(1). They are close to the bandwidth allocated by the optimal offline scheduler. When comparing by the window-sliding number (see Fig. 8(b)), our ADWS method with additional aggressive work-ahead scheme can achieve better results than SLWIN(1). Comparing to SLWIN(*W*), our ADWS method can achieve over 22% improvements for the required bandwidth. The required window-sliding number and peak bandwidth for online transmitting *Lecture* are shown in Fig. 9(b) and (c), respectively. Comparing to Fig. 8, our obtained improvements for *Lecture* are smaller than that obtained for *Advertisements*. As shown in Fig. 8(c) and Fig. 9(c), the relation between the required bandwidth and the available buffer size for *Advertisements* is more complex than that for *Lecture*. Given the same buffer size and playback delay, we require larger bandwidth to transmit the high-burst *Advertisements* stream. These results demonstrate that the performance for online traffic smoothing depends on the high/low of traffic burst.

C. Smoothing Effect of Large/Small Group of Picture Sizes

Comparing *Princess Bride* and *CNN News*, these two media traces are encoded by hardware MPEG encoders. As the hardware encoder uses variable distortion coding to maintain target rate, there is no long-term burst in these two streams. These two streams have the same average frame rate and the similar stream length. They are different in the maximum frame size (MFS) and the standard deviation (STD) of frame sizes. Fig. 10(a) and (b) show the cumulative playback functions of the first 100 frames for *Princess Bride* and *CNN News*, respectively. By following the traffic smoothing algorithm, a stream with large MFS and STD would require more system resources for jitter-free transmission as shown in Fig. 10(c). Comparing these two steams, *CNN News* has larger MFS and STD than *Princess Bride* does. The required peak bandwidth for *CNN News* (= 4.30 Mbps) is larger than that for *Princess Bride* (= 2.85 Mbps). Our experiments show that, if a video stream has no long-term burst, the obtained results will depend on its MFS and STD. Notably, in *Princess Bride* and *CNN News*, our required window-sliding number are nearly the same as those for SLWIN(*W*) -- the fastest static window-sliding method -- when a small buffer is allocated (< 100 KB). Our ADWS method can allocate a small network bandwidth for online media transmission. Additionally, the required computation cost is also small. Our ADWS method is shown effective and efficient for either high-bursty or low-bursty online traffic.

V. Conclusion

This paper proposes a traffic smoothing method for delivery of online media stream. The ADWS method dynamically decides the suitable window-sliding size and applies the aggressive work-ahead scheme to smooth the VBR traffic. We have explored the ADWS algorithm by transmitting various benchmark video streams. By modeling of the traffic, the relations between different characteristic of input stream (the high/low burst of traffic, the MFS and STD of frame sizes) and different behaviors

of obtained scheduling results are discussed. Our ADWS method is shown effective and efficient. The theoretical analysis shown in Section III and the empirical results shown in Section IV are verified. Note that, our allocated peak bandwidth is the same as that of SLWIN(k) if the applied window size W ? n (n is the number of frames in the media stream). Although the increasing of window size may reduce the required peak bandwidth, the delay time for media playback would be also increased. Defining the window size as multiple of GoP scale may not be acceptable for real-world applications. Therefore, our applied window size is given as the number of frames in one GoP. The related playback delay is nearly one second. Based on the aggressive work-ahead scheme, we minimize bandwidths allocated for consecutive windows in a dependent manner. The selection of initial bandwidth would affect the obtained results. For example, in Fig. 4(c), we can achieve a small peak bandwidth if the initial bandwidth is given as $(G(e_2) - G(0)) / (e_2 - 0) = G(3W) / e_2$. In this paper, for the simplification, our initial bandwidth is decided by the frame sizes in the first window. However, these frames may not be suitable for characterizing the entire stream. Our future work is to design a more intelligent method to select the suitable initial bandwidth. Although SLWIN(1) coupled with our aggressive work-ahead scheme may allocate as small bandwidth as other policies, it is computational expensive. More effort should be taken to reduce its computation cost. It is necessary to design a new algorithm to further minimize the peak bandwidth allocated for delivery of online media streams with low time complexity and high resource utilization.

Acknowledgement

The authors would like to thank Dr. Don Towsley and Dr. Zhi-Li Zhang for providing the source code of their smoothing algorithm [5].

Appendix:

✍ Given the same window-sliding size k to smooth the same online media stream, the transmission schedule obtained by our scheme has at most the same peak bandwidth as that obtained by SLWIN(k).

Proof:

As the window-sliding size k is the same, we have the same traffic windows considered for smoothing. Let $G(.)$ and $G'(.)$ be the transmission schedule obtained by our scheme and the conventional SLWIN(k) scheme, respectively.

(1) Consider the first traffic window where $G'(-D) = G(-D) = 0$ and the constant D is the initial delay. As the media frames smoothed by our scheme and the SLWIN(k) scheme are the same, both schemes have the same peak bandwidth. By applying the aggressive work-ahead scheme, we can guarantee $G'(x) \geq G(x)$ for $x = -D$ to $W-D$.

(2-1) Given the window-sliding size k, the start point of the second window is $k-D$. From definitions of the window size W and the window-sliding size k, we have $1 \leq k \leq W$. Therefore, $-D < k-D \leq W-D$ and $G'(k-D) \geq G(k-D)$.

(2-2) Note that the media data need to be smoothed by our scheme is at most the same as that by SLWIN(k). By applying the aggressive work-ahead scheme, our scheme has at most the same peak bandwidth as SLWIN(k). Additionally, we can guarantee $G'(x) \geq G(x)$ for $x = k-D$ to $k+W-D$. (It can be easily shown by contradiction.)

(3) The same idea can be applied to other windows to deduce that, given the same window-sliding size k to smooth the same online media stream, the transmission schedule obtained by our scheme has at most the same peak bandwidth as that obtained by SLWIN(k).

Reference

[1] J.Y. Hui, J. Zhang and J. Li, "Quality of service control in GRAMS for ATM local area network," *IEEE JSAC*, 1995.

[2] R.I Chang, M. Chen, M.T. Ko and J.M. Ho, "Optimizations of stored VBR video transmission on CBR channel," *SPIE VVDC*, pp. 382-392, 1997.

[3] R.I Chang, M. Chen, M.T. Ko and J.M. Ho, "Designing the on-off CBR transmission schedule for jitter-free VBR media playback in real-time networks," *IEEE RTCSA*, pp. 1-9, 1997.

[4] J.M. McManus and K.W. Ross, "Dynamic programming methodology for managing prerecorded VBR sources in packet-switched networks," *SPIE VVDC*, 1997.

[5] 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," *ACM SIGMETRICS*, pp.222-231, 1996.

[6] W. Feng and S. Sechrest, "Smoothing and buffering for delivery of prerecorded compressed video," *IS&T/SPIE MMCN*, pp. 234-242, 1995.

[7] W. Feng, F. Jahanian and S. Sechrest, "Optimal buffering for the delivery of compressed prerecorded video," *IASTED International Conference on Networks*, Jan 1996.

[8] W. Feng, "Rate-constrained bandwidth smoothing for the delivery of stored video," *IS&T/SPIE MMCN*, 1997.

[9] J.M. McManus and K.W. Ross, "Video On Demand over ATM: Constant-Rate Transmission and Transport," *IEEE INFOCOM*, March 1996.

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

[11] H. Zhang and E.W. Knightly, "A new method to support delay-sensitive VBR video in packet-switched networks," *NOSSDAV*, 1995.

[12] E.W. Knightly, D.E. Wrege, J. Liebeherr and H. Zhang, "Fundamental limits and tradeoffs of providing deterministic guarantees to VBR video traffic," *ACM SIGMETRICS*, pp. 47-55, 1995.

[13] S. Radhakrishnan and S.V. Raghavan, "A flexible traffic shaper for high-speed networks: design and comparative study with leaky bucket," *CS-TR-3495*, University of Maryland, 1995.

[14] E.W. Knightly and P. Rossaro, "Smoothing and multiplexing tradeoffs for deterministic performance guarantees to VBR video," *Int. Symp. Multimedia Communication and Video Coding*, 1995.

[15] J. Rexford and D. Towsley, "Smoothing variable-bit-rate video in an internetwork," *SPIE VVDC*, 1997.

[16] A.R. Reibman and A.W. Berger, "Traffic descriptors for VBR video teleconferencing over ATM networks," *IEEE/ACM ToN*, June 1995.

[17] M. Grossglauser and S. Keshav, "On CBR Service," *IEEE INFOCOM*, March 1996.

[18] R.I. Chang, M.C. Chen, J.M. Ho and M.T. Ko, "Characterizing the minimum required resources for admission control of pre-recorded VBR video transmission by an O($n$log$n$) algorithm," *IEEE Int. Conf. Computer Communications and Networks* (IC3N), pp. 674-681, 1998.

[19] S.S. Lam, S. Chow and D.K.Y. Yau, "An algorithm for lossless smoothing of MPEG video," *ACM SIGCOMM*, 1994.

[20] T. Ott, T.V. Lakshman and A. Tabatabai, "A scheme for smoothing delay-sensitive traffic offered to ATM networks," *IEEE INFOCOM*, 1992.

[21] J. Rexford, S. Sen, W. Feng, K. Kurose, J. Stankovic and D. Towsley, "Online Smoothing of Live, Variable-Bit-Rate Video," *NOSSDAV*, 1997.

[22] M. Chen, D.D. Kandlur and P.S. Yu, "Optimization of the grouped sweeping scheduling (GSS)

with heterogeneous multimedia streams," *ACM Multimedia*, 1993.

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

[24] Y.C. Wang, S.L. Tsao, R.I. Chang, M.C. Chen, J.M. Ho and M.T. Ko, "Fast data placement scheme for video server with zoned-disks," *Proc. SPIE Vol. 3229, Multimedia Storage and Archiving Systems II*, pp. 92-102, 1997.

[25] Oliver Rose, "Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems," University of Wuerzburg, Institute of Computer Science Research Report Series, Report No. 101, February 1995.

[26] ftp://thumper.bellcore.com/pub/vbr.video.trace, ftp://ftp-info3.informatik.uni-wuerzburg.de/pub/MPEG/ traces/

[27] J. Zhang, "Optimal buffering algorithms for client-server VBR video retrievals," Ph.D. Thesis, The State University of New Jersey, 1997.

[28] R.I. Chang, W.K. Shih and R.C. Chang, "Real-time disk scheduling for multimedia applications with deadline-modification-scan scheme," *Real-Time Systems*, vol.19, no.2, 149-168, 2000.

[29] R.I. Chang, C.L. Chen, W.K. Shih, and R.C. Chang, "Design and implementation of a real-time disk scheduling algorithm by the multi-segment cascade scheme," *The 20th IEEE Real-Time Systems Symposium (RTSS99) - WIP*, 1999.

[30] R.I. Chang, M.C. Chen, J.M. Ho and M.T. Ko, "An effect and efficient traffic-smoothing scheme for delivery of online VBR media streams," *IEEE INFOCOM'99* (Annual Joint Conference of the IEEE Computer and Communications Societies), vol.2, pp.447 -454.

[31] R.I. Chang, M.C. Chen, J.M. Ho and M.T. Ko, "Traffic-smoothing for delivery of online VBR media streams by a dynamic window-based approach," Technical Report TR-IIS-98-013, Institute of Information Science, Academia Sinica, 1998.

Table I. Statistics of the test VBR-encoded MPEG media streams. (AVG: average bit-rate, MFS: the maximum frame size, STD: standard deviation of frame size)

Stream	AVG	MFS	STD
Star Wars	0.36 Mbps	22.62 KB	2.3
Advertisements	0.45 Mbps	10.08 KB	1.9
Lecture	0.33 Mbps	6.14 KB	1.6
CNN News	1.17 Mbps	30.11 KB	4.8
Princess Bride	1.17 Mbps	29.73 KB	3.7

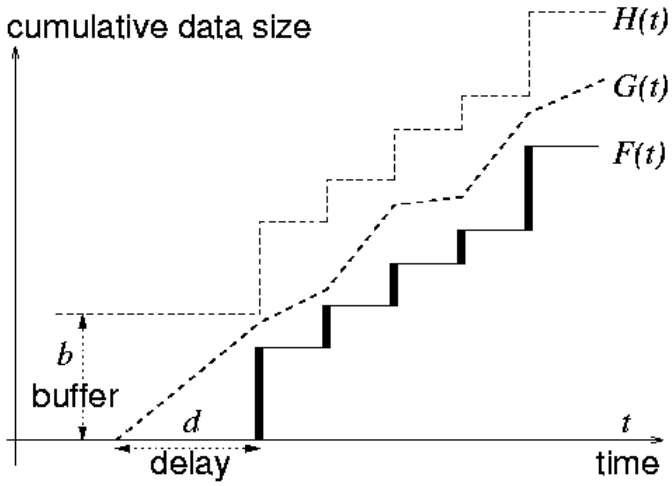


Fig. 1. A jitter-free transmission schedule for the given client buffer size and initial delay. The design goal of a good transmission schedule is to smooth the VBR traffic with the minimum resource allocation and the maximum resource utilization.

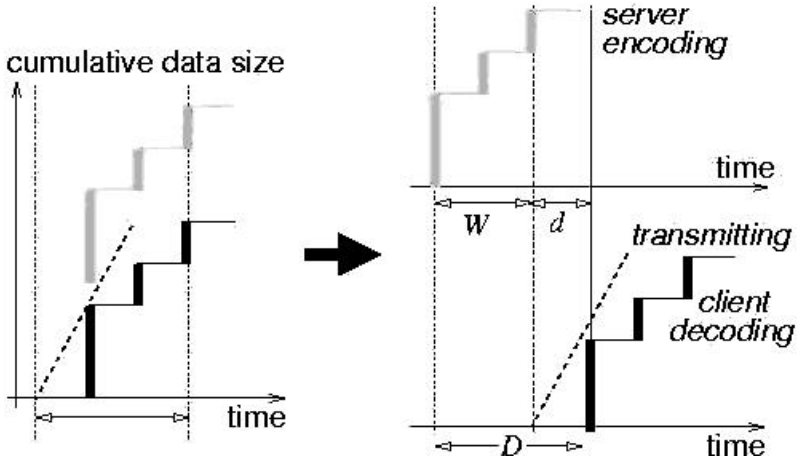
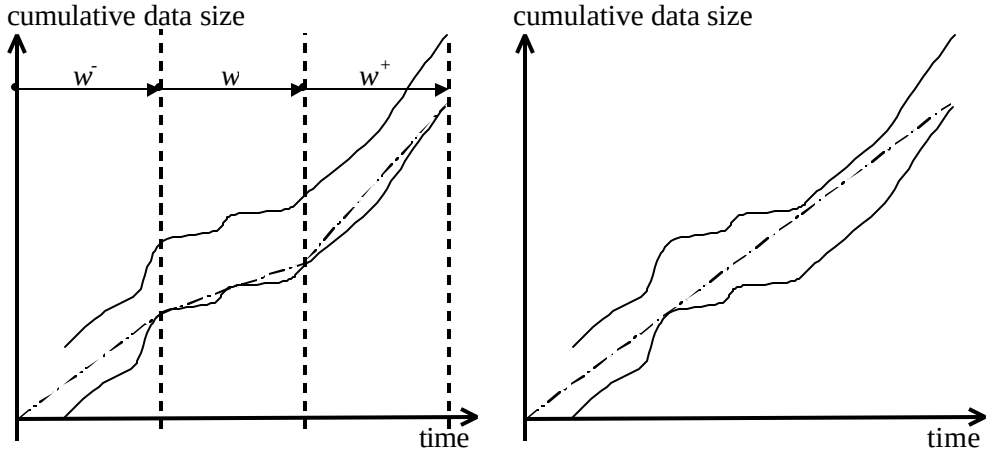
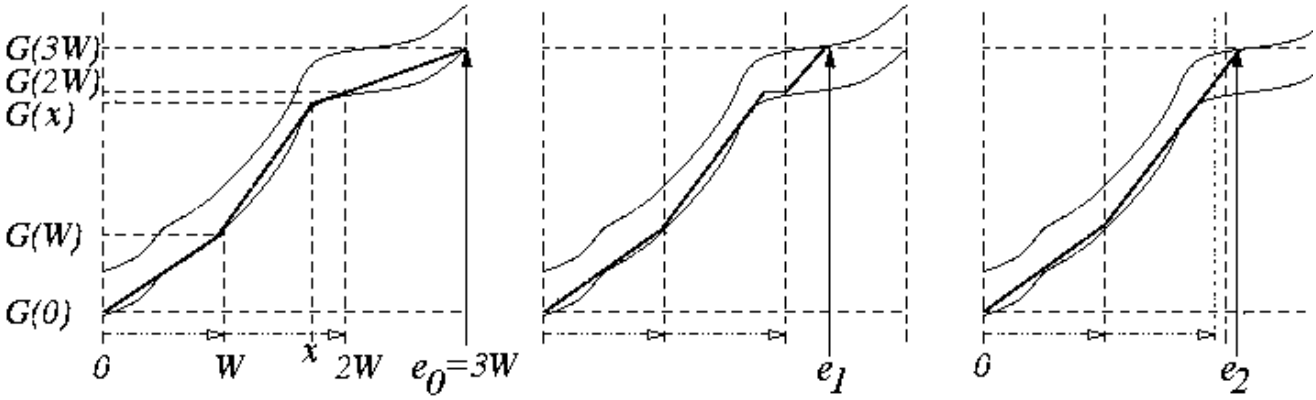


Fig. 2. The window-based online media stream with window size *W* and initial delay *D*. Notice that, the frames outside the current window are already scheduled or not generated. It demonstrates the differences between the pre-recorded media stream and the online generated media stream.



(a) (b)

Fig. 3. (a) The obtained schedule of SLWIN(*W*) requires a large bandwidth. (b) A better schedule is shown for comparison.



(a) (b) (c)

Fig. 4. (a) The previous approach releases the network channel at time $e_0 = 3W$. (b) When the aggressive work-ahead scheme is applied, the network channel can be released at time e_1 where $e_1 < e_0$. (c) When the window-sliding size can be dynamically changed, network channel can be released at time e_2 where $e_2 < e_1 < e_0$.

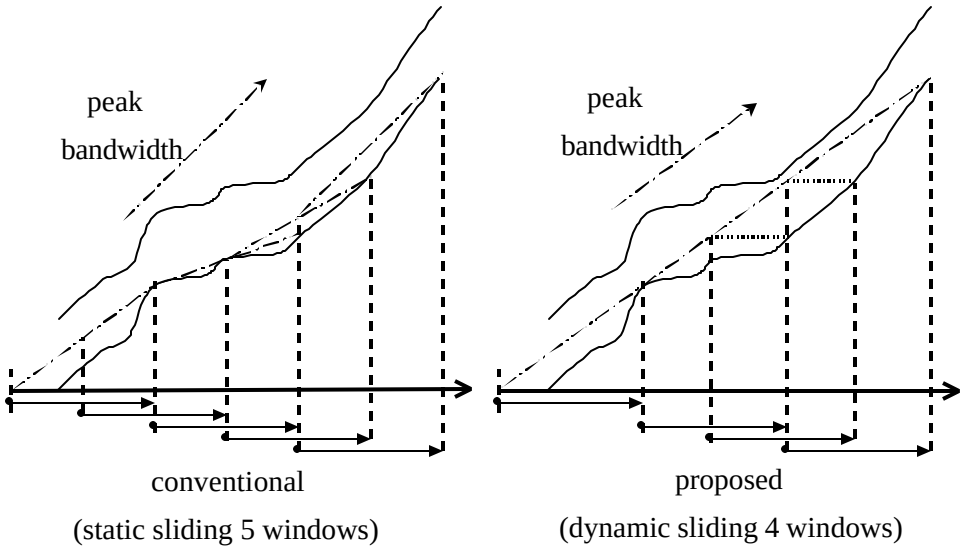


Fig. 5. Our ADWS method can dynamically decide the suitable window-sliding size to shape the bursty traffic. It is different from the previous method with a constant window-sliding size.

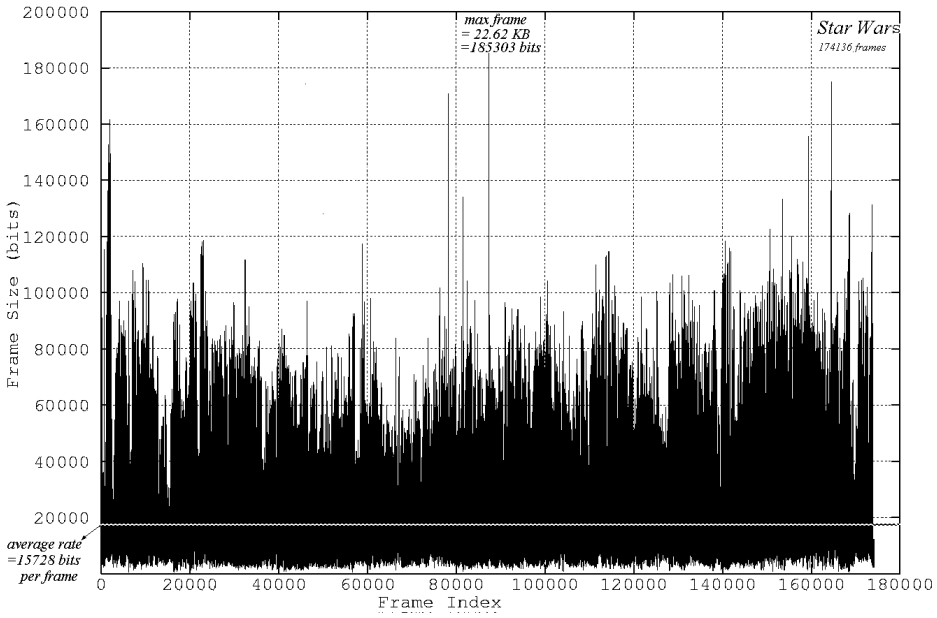
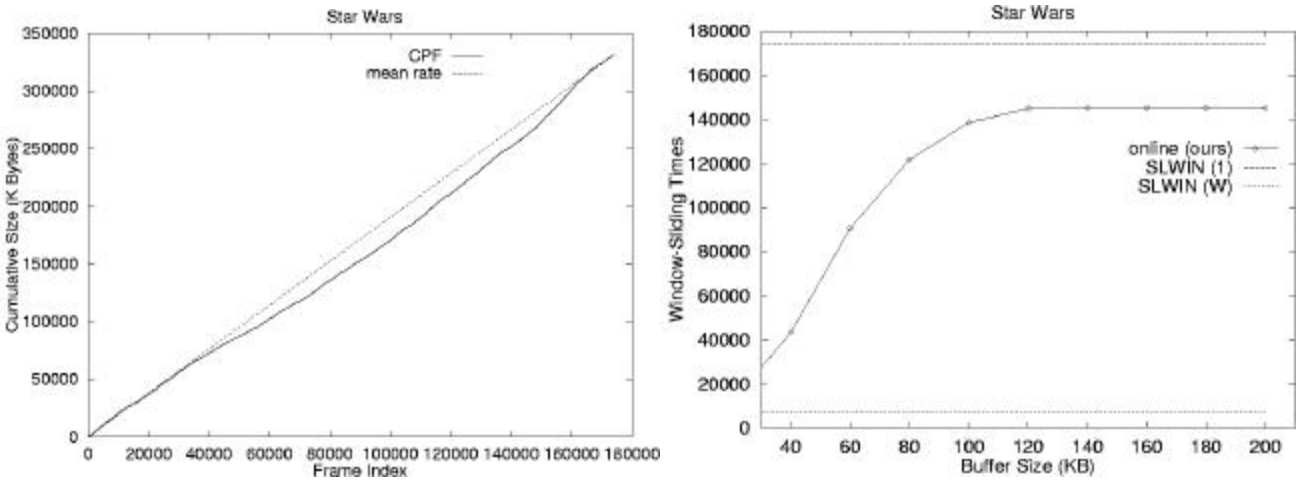
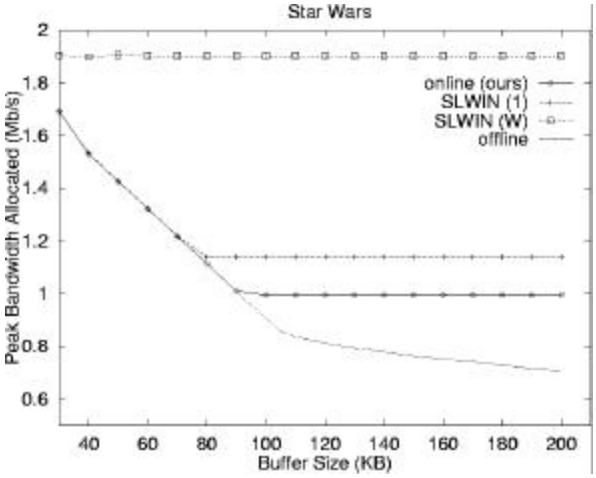


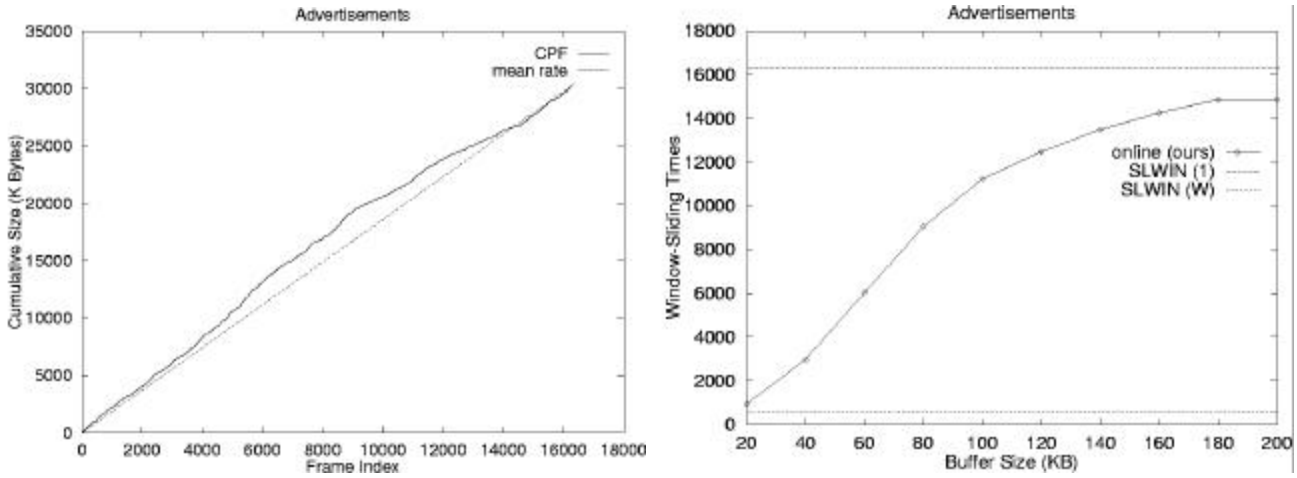
Fig. 6. A plot of the frame number versus frame size with an overlaying average.





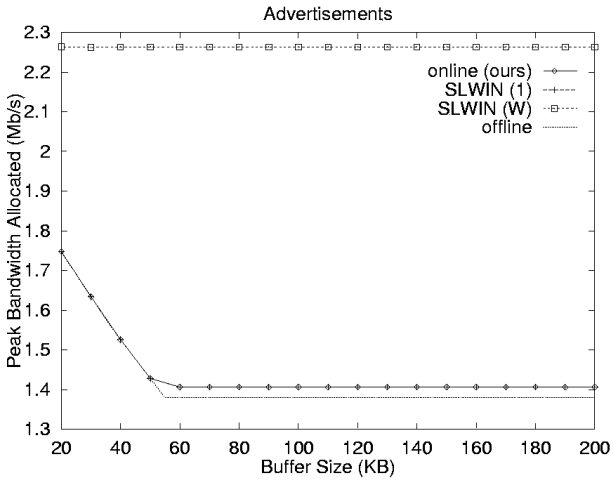
(c)

Fig. 7. (a) *Star Wars* movie stream. Comparisons of our method with the SLWIN(1), the SLWIN(*W*) and the offline scheduler on the basis of (b) window-sliding number and (c) required network bandwidth.



(a)

(b)



(c)

Fig. 8. (a) *Advertisements* video stream. Comparisons of our method with the SLWIN(1), the SLWIN(*W*) and the offline scheduler on the basis of (b) window-sliding number and (c) required network bandwidth.

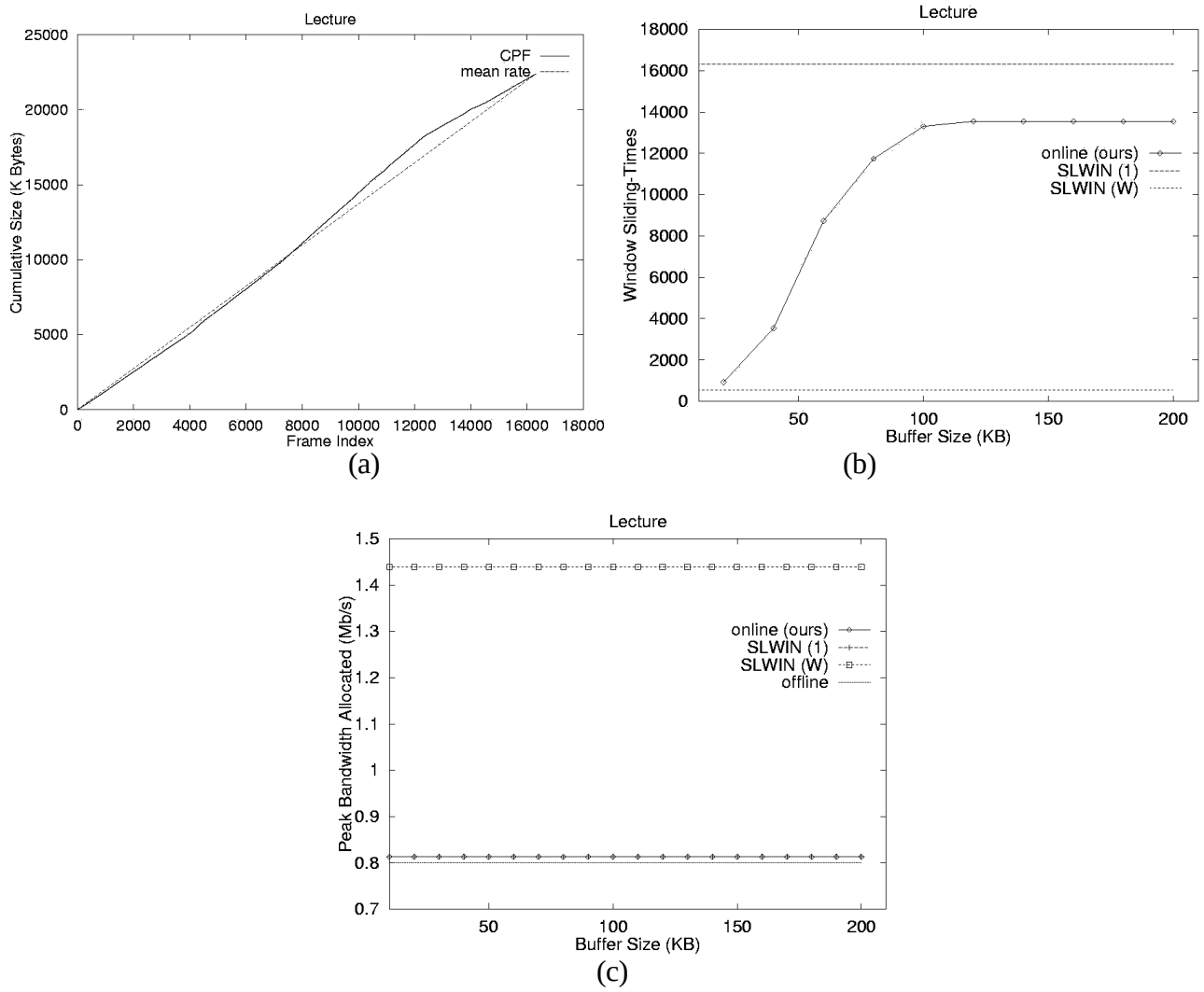


Fig. 9. (a) *Lecture* video stream. Comparisons of our method with the SLWIN(1), the SLWIN(*W*) and the offline scheduler on the basis of (b) window-sliding number and (c) required network bandwidth.

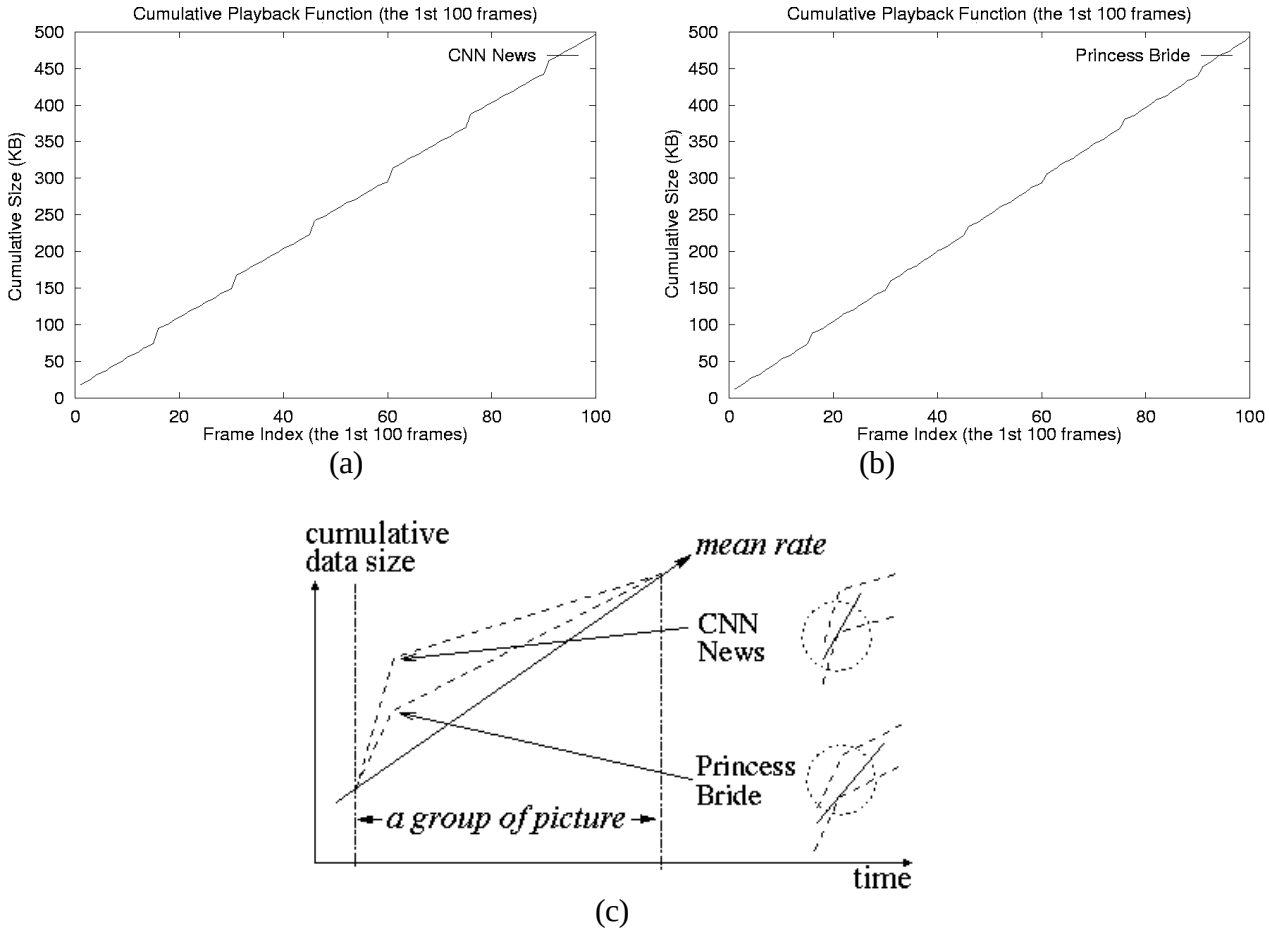


Fig. 10. (a) *CNN News* video stream has large MFS and STD. (b) *Princess Bride* video stream has small MFS and STD. (c) *CNN News* video stream with large MFS and STD would require a high peak bandwidth for jitter-free transmission.