Aggregated Smoothing: Considering All Streams Simultaneously for Transmission of Variable-Bit-Rate Encoded Video Objects

Sooyong Kang and Heon Y. Yeom

Abstract: **Transmission of continuous media streams has been a challenging problem of multimedia service. Lots of works have been done trying to figure out the best solution for this problem, and some works presented the optimal solution for transmitting the stored video using smoothing schemes applied to each individual stream. But those smoothing schemes considered only one stream, not the whole streams being serviced, to apply themselves, which could only achieve local optimum not the global optimum. Most of all, they did not exploit statistical multiplexing gain that can be obtained before smoothing. In this paper, we propose a new smoothing scheme that deals with not an individual stream but the whole streams being serviced simultaneously to achieve the optimal network bandwidth utilization and maximize the number of streams that can be serviced simultaneously. We formally proved that the proposed scheme not only provides deterministic QoS for each client but also maximizes number of clients that can be serviced simultaneously and hence achieves maximum utilization of transmission bandwidth.**

Index Terms: **Variable bit rate, traffic smoothing, aggregated smoothing, multimedia systems.**

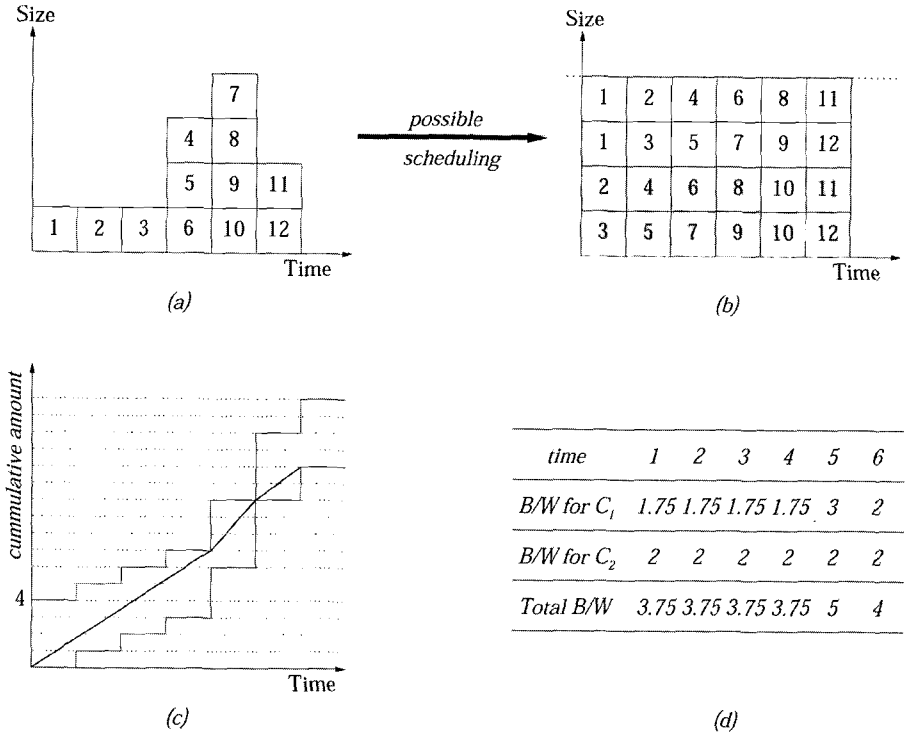
I. INTRODUCTION

The VBR(Variable Bit Rate) nature of the video data presents a challenge in designing VOD systems since the disk bandwidth as well as network bandwidth should be guaranteed as well as maximizing the utilization. Smoothing of VBR streams is one of the widely used techniques to enhance network bandwidth utilization of the VOD(Video-On-Demand) system. Smoothing is used to reduce the bit rate variability of the video streams. It is performed by transmitting all or part of high bit rate frames in advance to reduce the peak data rate of the stream, and allocating network bandwidth to the stream according to the reduced peak data rate. Hence it is possible to achieve high bandwidth utilization as well as providing deterministic service. Various research have been conducted based on this approach [1]–[5]. And the performance of each scheme was evaluated in [6]. Another approach to enhance network bandwidth utilization is to permit some level of QoS(Quality-of-Service) degradation by giving up deterministic service guarantees. This approach exploits the indirect smoothing effect that comes out from the reduced per-stream bit rate variability when lots of VBR streams are multiplexed, and the effect is known as *'Statistical Multiplexing Gain.'* Hence, the schemes based on this approach presented a probabilistic bound on the QoS degradation [?]–[10]. These schemes exploit the fact that the video services are less error sensitive than data services and so the probablistic QoS guarantee is sufficiently reasonable to consider. However, if the service requires payment, like Pay-Per-View system, users may expect deterministic QoS rather than probabilistic QoS. There is also a scheme combining the two approaches [11]. After reducing peak data rate through a proper prefetching scheme, this approach multiplex those smoothed streams statistically to raise network bandwidth utilization. However, to the best of our knowledge, there has been no work on smoothing a set of multiplexed streams on the whole. Two of the recent smoothing schemes for a stored video are 'optimal smoothing' from [3] and 'on-off smoothing' from [4]. Optimal smoothing minimizes the variability of allocated bandwidth to a stream without start delay, while on-off smoothing permits only two transmission state 'on' and 'off' with allocating a fixed bandwidth to a stream. Hence, while the optimal smoothing requires some level of variability to the bandwidth allocation after smoothing, on-off smoothing enables a fixed bandwidth allocation to a stream which makes it easy to provide a deterministic service without wasting serious amount of network bandwidth. But as the on-off smoothing engages start delay, the real playback is delayed to prefetch necessary amount of data in advance. The combined scheme in [11] is a simple statistical multiplexing of optimally smoothed streams. As they described, the statistical multiplexing gain that can be obtained by statistically multiplexing optimally smoothed streams is the lower bound of the statistical multiplexing gain that can be obtained by multiplexing the original streams. And it can be understood that the optimal smoothing scheme removes a great portion of statistical multiplexing gain that can be obtained without the optimal smoothing scheme. Similarly, on-off smoothing removes almost all of the statistical multiplexing gain on behalf of deterministic service guarantee. Therefore, we can roughly conclude that smoothing of individual stream reduces statistical multiplexing gain, which leaves room for performance enhancement when statistical multiplexing gain can be fully obtained while providing deterministic service. The problem of those smoothing schemes is that they are based on *'individual smoothing'* - smoothing each individual stream by itself without considering the other streams transmitted with it. Hence it can not take into consideration the interaction of bit rates of each stream transmitted simultaneously. Hence, in spite that an individual smoothing scheme can be optimized like optimal smoothing, the optimized scheme

Manuscript received July 18,2001; approved for publication by Jaiyong Lee, Division III Editor, July 15, 2003.

S. Kang is with the Department of Computer Science Education, Hanyang University, Seoul, Korea, email: sykang@hanyang.ac.kr.

H. Y. Yeom is with the School of Computer Science and Engineering, Seoul National University, Seoul, Korea, email: yeom@snu.ac.kr.



time	1	2	3	4	5	6
B/W for C_1	1.75	1.75	1.75	1.75	3	2
B/W for C_2	2	2	2	2	2	2
Total B/W	3.75	3.75	3.75	3.75	5	4

Fig. 1. Problem of individual smoothing: (a) Original stream(client needs at least 4 buffer unit), (b) possible schedule with client buffer size 4 and 6, (c) optimal smoothing with client buffer size 4(if buffer size is 5 or 6, straight line with slope 2 can be made), (d) bandwidth requirement for optimal smoothing when total available bandwidth is 4.

can not be regarded as a globally optimized smoothing scheme that maximizes the number of streams serviced simultaneously with given resources.

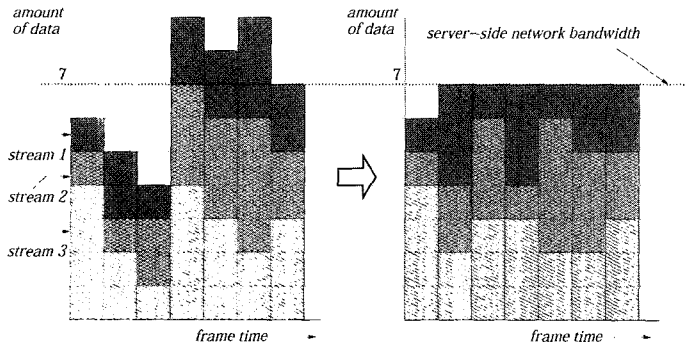
In this paper, we propose a globally optimized smoothing scheme, *'Aggregated smoothing,'* which maximizes the number of active streams while providing deterministic service to each clients. The proposed scheme minimizes the overall client buffer requirement through maintaining the original statistical multiplexing gain. Aggregated smoothing scheme smoothes all the multiplexed streams being transmitted together on the whole.

The organization of the rest of this paper is as follows. In section II, we investigate the detailed problem of individual smoothing schemes using the optimal smoothing and the on-off smoothing as examples. Section III describes the concept of aggregated smoothing and in section IV we present an admission control scheme that extracts the proper transmission schedule of all clients. In addition, the deterministic service guarantee is proved formally. Section V deals with the optimality of the aggregated smooting scheme. Finally, the concluding remarks are presented in section VI.

II. INDIVIDUAL SMOOTHING

Individual smoothing schemes such as optimal smoothing[3] and on-off smoothing[4] do not consider the characteristics of other streams or clients being serviced at the same time. They are applied to each individual stream considering only the frame size of each unit time and client buffer size. It results that, for optimal smoothing, an extra multiplexing scheme is required to transmit optimally smoothed streams simultaneously, and for on-off smoothing, since the inter-stream interaction is totally excluded from the scheme the bandwidth utilization decreases rapidly as client buffer size decreases. These problems originated from the fact that the two smoothing schemes smooth each individual stream independently from others. These individual smoothing can not exploit the advantage of statistical multiplexing arising from the transmission of multiple streams simultaneously. Since they do not obtain the affirmative effect of varying the assigned bandwidth according to the state of client buffer and frame size of each stream, they can not achieve the global optimum.

Decoding equipment of each client can have different sized buffers. In an individual smoothing, each stream uses its bandwidth allocated in terms of time exclusively, which disables optimizing overall buffer utilization. As a result, a new request that can be serviced through a proper scheduling might be rejected by an admission control algorithm for individual smoothing. Fig. 1 shows such an example. First, Fig. 1(a) shows the shape of an original object. When two clients C_1 and C_2 with buffer size 4 and 6, respectively, request the object at the same time and the total available bandwidth is 4, one of the two requests can not be accepted if the optimal smoothing is applied. As we can see from Fig. 1(c), the optimal smoothing scheme generates assigned bandwidth of 1.75, 1.75, 1.75, 1.75, 3, and 2 in each time for the client C_1. For client C_2, the optimal smooth-



Before and after aggregated smoothing. In this case, it is assumed that all the clients have the same buffer size(= 5)

Fig. 2. Aggregated smoothing.

Table 1. Buffer requirements in case of Fig. 1(b).

$time$	1	2	3	4	5	6
C_1	1	2	3	4	4	2
C_2	3	4	5	6	4	2

ing scheme generates constant bandwidth allocation with allocated bandwidth 2. Hence, for the two clients, C_1 and C_2, the total required bandwidth(5) exceeds the available bandwidth(4). However, with proper allocation of bandwidth to each client, both the requests can be accepted as can be seen in Fig. 1(b). The buffer requirements for client C_1 and C_2 in each time is presented in Table 1. As we can see from the example, despite that the individual smoothing can be optimal for a stream, it is not optimal on the whole. Thus, to get to the global optimum, it is needed to deal with all streams collectively.

III. AGGREGATED SMOOTHING

Since aggregated smoothing regards all the multiplexed streams as one huge stream to smooth the variation of data rate, it transmits data independently of the buffer-bandwidth relation[4] of each stream. What it does is that it is trying to vary the bandwidth allocated to each stream dynamically according to the clients' state of their buffers. Since the total bandwidth available to all streams currently being serviced is fixed on the link(or device) bandwidth, if the aggregate of bandwidth requirements from each stream do not exceed the link bandwidth it does nothing but transmit all the streams with their own data rates. On accepting a new stream by which the total required bandwidth exceeds link bandwidth, it is needed to determine when to prefetch how much data from which stream(s). In this section, we present an underlying principle of aggregated smoothing. Table 2 defines important notations used in this article.

Fig. 2 shows an example of aggregated smoothing. As we can see in the figure, each stream is transmitted with their own data rate when the aggregate of frame sizes does not exceed the link bandwidth and there is no data to be prefetched at the time. But when the total required bandwidth exceeds the link bandwidth, it is needed to prefetch surplus data in advance. Hence, smoothing by prefetching is to be done at this time and it should determine 1) which stream(s) to be prefetched and 2) how much data of the selected stream(s) to be prefetched.

Table 2. Notations.

t	frame time
t_l	the time that the last stream finishes
$f_i(t)$	frame size of stream i at time t
$P_i(t)$	amount of data to be prefetched for stream i at time t
$P(t)$	total amount of data that should be prefetched at time t
$B_i(t)$	available buffer size of client i at time t
B_i	buffer size of client i
$D_i(t)$	amount of data belonging to stream i that should be transmitted at time t
$MPA_i(t)$	maximum prefetchable amount of client i at time t
$MPA(t)$	total prefetchable amount at time t, $(= \sum_i MPA_i(t))$
τ_t	the real transmission time of data that should be transmitted at or before time t
$\wp$	the amount of data already being prefetched currently
$\mathcal{B}$	server-side network bandwidth

Table 3. $P(t)$ value in case of Fig. 2.

t	1	2	3	4	5	6	7
$P(t)$	0	2	5	3	2	0	0

Table 4. Remaining buffer size of each client in case of Fig. 2.

t	1	2	3	4	5	6	7
$stream1$	4	2	3	1	2	2	3
$stream2$	4	3	1	2	1	1	3
$stream3$	1	3	2	0	2	3	2

A. Determining Prefetch Amount

There are two important things that should be considered when determining the prefetch amount. First, sufficient amount of data should be prefetched to keep the timeliness of data transmission for deterministic service guarantee. Second, minimal amount of data should be prefetched not to enlarge the required client buffer size by unnecessary prefetching. The former is related to the quality of service and the latter to the optimality of the smoothing scheme. In this section we only consider the former, and the latter will be dealt with in Section V. To provide deterministic service guarantee, data belonging to the frame that should be displayed at time t must be actually transmitted on or before t assuming that there are no physical time delay concerning transmission and decoding. Thus, if the total amount of data that should be transmitted at time t exceeds link capacity, the surplus data should be prefetched before t. Two different cases exist that the amount of data to be transmitted at time t exceeds the link bandwidth. First, the sum of frame sizes of each stream at time t exceeds the link bandwidth. Second, while the sum of frame sizes of each stream does not exceed the link bandwidth, the sum of the current sum and the data of future frames that should be prefetched at the time exceeds the link bandiwdth. Let $S_0, S_1, \cdots, S_{n-1}$ be the streams being serviced currently and let $f_i(t)$ be the frame size of S_i at time t. Then the above two cases can be generalized as $\sum_{i=0}^{n-1} f_i(t) + P(t) > \mathcal{B}$, where $P(t)$ is the total prefetch amount at time t. $P(t)$ is determined by the amount of data at time later than t and to determine the value, aggregated smoothing regards the whole stream as one huge stream, S, and calculate prefetch amount at each time.

Algorithm(Prefetch Amount Calculation)

$f_S(t)(=\sum_{i=0}^{n-1} f_i(t))$: data amount of stream S at time t;
B : link bandwidth
$surplus$: amount of surplus data;
$P(t) = 0$ for all $0 \leq t \leq t_l$;
$surplus = 0$;
for($t = t_l; t \geq 0; t = t - 1$) {
  if($f_S(t) > B$) {
    $surplus = surplus + f_S(t) - B$;
    $P(t-1) = surplus$;
  }
  else {
    $surplus = \max\{surplus - (B - f_S(t)), 0\}$;
    $P(t-1) = surplus$;
  }
}

Fig 3. Prefetch amount calculation algorithm.

Fig. 3 shows the algorithm calculating the total prefetch amount at each time. The algorithm is executed at the admisson control time and the result can be used unchanged until another new client is accepted and the algorithm generates new result to replace the old one. Table 3 shows the $P(t)$ values for the case shown in Fig. 2.

In this case, since the maximim prefetch amount is 5, there is no way to provide deterministic service if the sum of client buffer sizes is less than 5. To prevent this kind of QoS degradation, client buffer sizes at the admission control time need to be considered.

B. Determining Streams to be Prefetched

To actually prefetch already determined amount of data($P(t)$), target streams to prefetch and the amount of data to be prefetched in each stream are to be determined. Simple schemes easy to think can be to select streams in order of their data rate at the time and prefetch data in that order or to select all streams(n) as target streams and prefetch $P(t)/n$ data from each stream. But these schemes can not utilize client buffer efficiently. The former scheme can result in buffer overflow since it shows a biased selection of target streams. One example is presented in Fig. 4. For the streams of which the frame size in each time appears in (a), prefetching the highest data rate stream first generates the transmission schedule as (b). The scheme always selects stream A as a target stream. In this case, as we can see from (d), the maximum buffer requirement for prefetching of stream A is 7 while the corresponding client has buffer sized 5, which results in client buffer overflow. But if prefetching is done according to the transmission schedule in (c), deterministic service can be provided since buffer requirements for all clients are less than or equal to 5(e). The latter scheme can also result in client buffer overflow like in the case of the former scheme because of the ignorance of client buffer size. That is, assigning the same prefetch amount to all the clients regardless of their buffer size can induce the buffer overflow for a client with relatively small sized buffer while the buffer of a client with relatively large sized buffer still remains partially unused. An example of this phenomenon can also be found easily. The reason why the above two simple schemes cause client buffer overflows that are avoidable with a proper prefetch schedule is that they do not consider the remaining buffer size of each client at the prefetching time. Therefore, to solve this problem, remaining buffer size of each client should be considered at target stream selection time. Namely, avoid prefetching streams of which client buffers have little space to store future data and prefetch relatively large amount of data of a stream of which client buffer has a lot of space unused, to prevent not inevitable buffer overflows. Hence, a stream of which the unused client buffer size is the largest should be the first target of prefetching and the change of the buffer state should be reflected without delay to the prefetch stream selection process. Table 4 presents the remaining buffer size of each client in Fig. 2. All the clients are assumed to have buffer sized 5. The right side of Fig. 2 is actually the result of smoothing using the remaining buffer size information appeared in table 4. In case that multiple streams of which remaining buffer sizes are the same exist, any stream among them can be selected as a target stream.

t	1	2	3	4
stream A	2	2	5	5
stream B	1	1	1	2
stream C	1	1	2	1
total	4	4	8	8

(a)

t	1	2	3	4
stream A	4	4	3	3
stream B	1	1	1	2
stream C	1	1	2	1
total	6	6	6	6

(b)

t	1	2	3	4
stream A	2	4	3	5
stream B	2	1	1	1
stream C	2	1	2	0
total	6	6	6	6

(c)

t	1	2	3	4
stream A	4	6	7	5
stream B	1	1	1	2
stream C	1	1	2	1
total	6	8	10	8

(d)

t	1	2	3	4
stream A	2	4	5	5
stream B	2	2	2	2
stream C	2	2	3	1
total	6	8	10	8

(e)

Fig. 4. Problem of prefetching highest rate first($B_i = 5$, for all i): (a) Original streams, (b) prefetching highest rate first, (c) possible solution, (d) buffer requirement for prefetching in case of (b), (e) buffer requirements for prefetching in case of (c).

C. Transmission Schedule

Let $\mathcal{T}(t)$ be a set of $D_i(t), i = 0, 1, \cdots, n-1$ when there are n streams being serviced. Transmisson schedule, $\mathcal{T}$ is a sequence of $\mathcal{T}(t)$s in order of time t from $t = 0$ to $t = t_l$. Admission control algorithm generates transmission schedule when a new request is accepted. According to the transmission schdule the real transmission of data in each stream is done and the transmission schedule guarantees deterministic service. The formal proof of the deterministic service guarantee by the transmission schedule is presented in the next section. The transmission schedule can be used unchanged until a new client is accepted for service and a new transmission schedule is generated by the admission control algorithm. Accepting a new client requires replacement of old transmission schedule with a newly generated transmission schedule.

IV. ADMISSION CONTROL

In this section, we present an admission control algorithm and prove that the algorithm guarantees deterministic service.

A. Admission Control Algorithm

Two major considerations in admission control are that 1)all the streams including the new stream should be transmitted without any delay within the given link bandwidth and 2)prefetching for smoothing should not invoke client buffer overflow to provide deterministic service. The admission control scheme presented in this section considers those two considerations simultaneously. The algorithm calculates the total prefetch amount that is needed for delayless transmission of data and then tries to make a transmission schedule that provides deterministic service considering the prefetch amount. If there is no such transmission schedule, the new request is rejected. As all the clients have limited amount of buffers, there is an upper bound of data amount that can be stored in the client buffer. Hence, distribution of total prefetch amount to each client should be done taking client buffer size into consideration and the admission control algorithm should trace the maximum prefetchable amount of data of each client. If total prefetch amount exceeds the sum of all the maximum prefetchable amount of each client, the new request should not be accepted to provide deterministic service to the existing clients. Let $MPA_i(t)$ be the maximum prefetchable amount of client C_i at time t. Then $MPA_i(t)$ can be defined as follows.

$$MPA_i(t) = \min\{B_i - f_i(t), P_i(t+1) + f_i(t+1)\}$$

In the above formula, $B_i - f_i(t)$ means the remaining buffer size and $P_i(t+1) + f_i(t+1)$ means the amount of data of which transmission deadline is $t+1$. Actually, the amount of data that can be prefetched at time t can exceed $P_i(t+1)+f_i(t+1)$ since data of which transmission deadline is larger than $t+1$ also can be prefetched as long as the total amount does not exceed the buffer size. However, prefetching data of which transmission deadline is larger than $t+1$ is meaningless because it is not indispensable[1]. Additionally, it not only occupies client buffer unnecessarily in advance but also increases bandwidth allocated to the client at time t. Hence, we assume that even if $P_i(t+1) + f_i(t+1)$ is less than $B_i - f_i(t)$, more data than $P_i(t+1) + f_i(t+1)$ will not be prefetched. As we can see from the defining formula of $MPA_i(t)$, $MPA_i(t)$ depends on $P_i(t+1)$, and $P_i(t+1)$ depends on the scheme of distributing total prefetch amount to clients.

To accept a new request, $MPA(t) \geq P(t)$ should be satisfied for all t. While $P(t)$ can not be changed since it depends only on link bandwidth($\mathcal{B}$) and frame size of each stream($f_i(t)$), $MPA(t)$ can be varied by changing $P_i(t)$s. Therefore, the existence of $P_i(t)$s, $i = 0, 1, \cdots, n-1$ that satisfy $MPA(t) \geq P(t)$ for all time t should be examined to determine the acceptance of new request. In addition, letting $t = 0$ be the time when a new stream begins to be transmitted, $P(0) + \sum_{i=0}^{n-1} f_i(0) - \mathcal{B}$ should be less or equal than the total amount of data already prefetched to each client($\wp$) at the time since additional prefetching is not possible at the time. Then we can formalize the definition of admission control as follows.

Definition 1 (Admission Control) : Let $\mathcal{P}(t)$ be a set of prefetch amounts for each client at time t, i.e., $P_0(t), P_1(t), \cdots, P_{n-1}(t)$ such that $\sum_{i=0}^{n-1} P_i(t) = P(t)$. And let $\mathcal{P}$ be a sequence, $\mathcal{P}(0), \mathcal{P}(1), \cdots, \mathcal{P}(t_l - 1)$. Then, admission control for a new client C_n is defined as testing the following two constraints.

- Existence of sequence $\mathcal{P}$ that satisfies, for $1 \leq t \leq t_l$,

$$\forall \mathcal{P}(t) \; of \; \mathcal{P}, \sum_{i=0}^{n-1} \min\{B_i - f_i(t), P_i(t) + f_i(t)\} \geq P(t-1).$$

- No additional data that should have been prefetched before time 0, i.e.,

$$P(0) + \sum_{i=0}^{n-1} f_i(0) - \mathcal{B} \leq \wp.$$

A $\mathcal{P}(t)$ that satisfies the inequality of the first constraint in the above definition is called 'proper $\mathcal{P}(t)$'. Calculating prefetch amount, $P(t)$, of each time can be done with the algorithm in the previous section, and the second constraint of above definition can be examined easily using $P(0)$. To examine the first constraint, we can simply test all the possible $\mathcal{P}$'s. However, as the number of all possible $\mathcal{P}$'s is roughly ${}_{n+P(0)-1}C_{P(0)} \times {}_{n+P(1)-1}C_{P(1)} \times \cdots \times {}_{n+P(t_l-1)-1}C_{P(t_l-1)}$, which is almost impossible to test on-line. Hence, instead of searching for the $\mathcal{P}$ that satisfies the constraint by examining all the cases, we generate $\mathcal{P}(t)$ that satisfies $MPA(t) \geq P(t)$ in reverse order from time $t_l - 1$ to 0 to find $\mathcal{P}$. Fig. 5 shows the admission control algorithm that contains this process. The admission control algorithm can be partitioned into two parts. The first part (the first while loop in the for loop) generates a $\mathcal{P}(t)$ that is 'proper' at time t. When the first part could not make a proper $\mathcal{P}(t)$, the second part redistributes future prefetch amount to make a proper $\mathcal{P}(t)$. Hence, the second part of the algorithm may change $\mathcal{P}(t+1), \mathcal{P}(t+2), \cdots, \mathcal{P}(t_l-1)$, keeping them still proper. In the first part, the algorithm selects a stream that has the largest remaining buffer size as a target stream for prefetching and prefetches one data unit. Changes to the remaining buffer size caused by the prefetching is reflected immediately. And the algorithm selects a target stream again until either all the prefetch amount are distributed($P = 0$) or there is no way to distribute the prefetch amount without client buffer overflow and delay($\mathcal{O} = \emptyset$) while keeping the past $\mathcal{P}(t)$s unchanged. The reason all the prefetch amount can not be distributed to clients is that $MPA(t)$ is less than $P(t)$, and in this case, the algorithm tries to increase $MPA(t)$ by manipulating $P_i(t'), t' \geq t+1$(the second part). Increasing $MPA(t)$ equal to the prefetch amount means that there is a sequence of $\mathcal{P}(t)$s from $t_l - 1$ to the current time that satisfies $MPA(t) \geq P(t)$ for all t from $t_l - 1$ to the current time. Kernel of the second part is the process of redistributing total prefetch amount by bandwidth reallocation. For the clients that have remaining buffer space but do not have data to prefetch, $MPA_i(t) = P_i(t+1) + f_i(t+1)$ is satisfied. In this case, by decreasing the futurebandwidth allocated to them, which results in the increment of data to be prefetched at current time, they increase $P_i(t+1)$s and finally increase $MPA_i(t)$. On the contrary, for the clients that have data to prefetch but don't have remaining buffer, $MPA_i(t) = B_i - f_i(t)$ is satisfied.

[1] Data with transmission deadline t does not mean that it belongs to the frame that will be played at time t. If data belonging to the latter case is determined to be prefetched at time $t-2$, its transmission deadline is regarded as $t-2$.

Algorithm(Admission Control)

C_i : client i, $R_i(t)$: remaining buffer size of client i at time t
$\mathcal{G}_1, \mathcal{G}_2, \mathcal{O}$: group of clients
$\mathcal{G}_1 = \mathcal{G}_2 = \mathcal{O} = \emptyset$;
Insert all clients into group $\mathcal{O}$;
$P_i(t_l) = 0$ for all corresponding i's
calculate $P(t), t = 0, 1, 2, \cdots, t_l - 1$;
if($P(0) + \sum_{i=0}^{n-1} f_i(0) - \mathcal{B} > \wp$) reject;
for($t = t_l - 1; t \geq 0; t = t - 1$) {
  $P = P(t)$;
  $R_i(t) = B_i - f_i(t)$;
  if($P > \sum_i R_i(t)$) reject;
  $D_i(t) = 0$ for all i's;
  $D_i(t+1) = P_i(t+1) + f_i(t+1)$ for all i's;
  while($P > 0$ and $\mathcal{O} \neq \emptyset$) {
    select C_i that has the largest $R_i(t)$ from group $\mathcal{O}$;
    if($D_i(t+1) > 0$ and $R_i(t) > 0$) {
      $P_i(t) = P_i(t) + 1$; $D_i(t+1) = D_i(t+1) - 1$; $D_i(t) = D_i(t) + 1$;
      $R_i(t) = R_i(t) - 1$; $P = P - 1$;
    }
    else if($R_i(t) > 0$)
      {insert C_i into group $\mathcal{G}_1$; delete C_i from group $\mathcal{O}$;}
    else if($D_i(t+1) > 0$)
      {insert C_i into group $\mathcal{G}_2$; delete C_i from group $\mathcal{O}$;}
    else delete C_i from group $\mathcal{O}$;
  }
  if($P > 0$) { /* not a proper $\mathcal{P}(t)$ */
    $t' = t + 1$;
    while($\sum_i f_i(t') > \mathcal{B}$ or $P(t') > 0$) {
      while($P > 0$) {
        select C_i that has the largest $D_i(t'+1)$ from $\mathcal{G}_1$;
        pick a client, C_j, that has the largest $P_j(t+1)$ from $\mathcal{G}_2$;
        if($D_i(t'+1) > 0$ and $P_j(t+1) > 0$) {
          $D_j(t'+1) = D_j(t'+1) + 1$; $D_j(t') = D_j(t') - 1$;
          $P_j(t+1) = P_j(t+1) - 1$;
          $D_i(t'+1) = D_i(t'+1) - 1$; $P_i(t) = P_i(t) + 1$;
          $R_i(t) = R_i(t) - 1$; $P = P - 1$;
          if($R_i(t) = 0$) delete C_i from group $\mathcal{G}_1$;
        }
        else if($P_j(t+1) = 0$) reject;
        else if($D_i(t'+1) = 0$) break;
      }
      $t' = t + 1$;
    }
    if($P > 0$) reject;
  }
}
accept;

Fig 5. Admission control algorithm.

In this case, by increasing the future bandwidth allocated to them, which results in the decreasing of data to be prefetched at current time, they decrease $P_i(t+1)$s as long as $MPA_i(t)$s are kept unchanged as $B_i - f_i(t)$. The increment of $P_i(t+1)$s of the former clients and the decreasing of $P_i(t+1)$s of the latter clients should be the same to keep the $P(t+1)$ unchanged. The group of the former clients are denoted by $\mathcal{G}_1$ and the group of the latter clients are denoted by $\mathcal{G}_2$. There are four different cases that a request is rejected. First, if the second constraint of Definition 1 is not safisfied, the request is rejected. Second, if the total prefetch amount at a time is larger than the sum of remaining buffer sizes the request is rejected. The third case is that if there is no way to reduce the prefetch amount of any client in group $\mathcal{G}_2$ when there still remains data that should be distributed for prefetch to clients. This is the case that the sum of frame sizes of clients in group $\mathcal{G}_2$ at the time is larger than the sum of link bandwidth and their buffer sizes. Finally, if the algorithm could not find any proper $\mathcal{P}(t)$ until it reaches a time when sum of frame sizes of all the clients does not exceed the link bandwidth and there is no data to prefetch, the request is rejected since bandwidth reallocation at that time is meaningless. At the time the admission control algorithm is finished, $P_i(t)$, $i = 0, 1, \cdots, n-1$, $t = 0, 1, \cdots, t_l - 1$ means the amount of data that should be prefetched to client C_i at time t and $D_i(t)$ means the total amount of data that should be transmitted to client C_i at time t. Hence, transmission schedule $\mathcal{T}$ is determined as a sequence $\mathcal{T}(0), \mathcal{T}(1), \cdots, \mathcal{T}(t_l)$, where $\mathcal{T}(t) = \{D_0(t), D_2(t), \cdots, D_{n-1}(t)\}$. The theoretical time complexity of the proposed admission control algorithm is $O(L^2 \cdot N \log N)$, where N is the number of streams currently being serviced and L is the length of stream(exactly t_l, assuming current time being 0). But as most of the rejection occurs at the first and the second rejection points of the algorithm and the outer while loop of the second part of the algorithm finishes within a few iteration generally, the actual time complexity will be about $O(LN \log N)$. In the mean time, the transmission schedule generated by the admission control algorithm removes any other computation for smoothing at the real transmission time. The data unit used in the aggregated smoothing scheme including admission control algorithm is the network transmission unit such as cell in ATM network or packet in ethernet environment. (In the ethernet environment, the proposed scheme can not completely remove the uncertainty of the network such as packet delay or delay jitter.)

B. Deterministic Service Guarantee

Lemma 1 (Nonexistence of Buffer Overflow) : Transmission schedule $\mathcal{T}$ generated by the admission control algorithm does not cause buffer overflow at the real transmission time.

Proof: In the first part of the admission control algorithm, $P_i(t)$ increases only if $R_i(t) > 0$. And in the second part of the algorithm, a stream is deleted from group $\mathcal{G}_1$ as soon as $R_i(t)$ becomes 0. Hence, the prefetch amount of stream i at time t, $P_i(t)$ never exceeds B_i. □

Lemma 2 (In-time Transmission) : Transmission schedule $\mathcal{T}$ generated by the admission control algorithm does not permit any delay of data.

Proof: We will show that $\tau_t \leq t$ for all $t (0 \leq t \leq t_l)$. At time t_l, the amount of data that should be transmitted is

$$D(t_l) = \sum_i D_i(t_l) = \begin{cases} \mathcal{B} & \text{if } \sum_i f_i(t_l) \geq \mathcal{B} \\ \sum_i f_i(t_l) & \text{otherwise} \end{cases}$$

If $D(t_l) = \sum_i f_i(t_l)$, it is trivial that $\tau_{t_l} \leq t_l$. If $D(t_l) = \mathcal{B}, \tau_{t_l} \leq t_l$ when $\tau_{t_l - 1} \leq t_l - 1$.
At time $t_l - 1$, the amount of data that should be transmitted is

$$\begin{aligned} D(t_l - 1) &= \sum_i D_i(t_l - 1) \\ &= \begin{cases} \mathcal{B}, & \text{if } \sum_i f_i(t_l - 1) + P(t_l - 1) \geq \mathcal{B} \\ \sum_i f_i(t_l - 1) + P(t_l - 1), & \text{otherwise} \end{cases} \end{aligned}$$

If $D(t_l - 1) = \sum_i f_i(t_l - 1) + P(t_l - 1)$, it is trivial that $\tau_{t_l - 1} \leq t_l - 1$. If $D(t_l - 1) = \mathcal{B}, \tau_{t_l - 1} \leq t_l - 1$ when $\tau_{t_l - 2} \leq t_l - 2$.

$\vdots$

At the current time(0), the amount of data that should be transmitted is

$$D(0) = \sum_i D_i(0) = \sum_i f_i(0) + P(0) - \wp$$

Since the admission control algorithm guarantees $P(0) + \sum_i f_i(0) - \mathcal{B} \leq \wp$, $D(0) \leq \mathcal{B}$ is satisfied. Hence, $\tau_0 = 0$.
Therefore, $\tau_1 \leq 1 \Rightarrow \tau_2 \leq 2 \Rightarrow \cdots \Rightarrow \tau_{t_l-1} \leq t_l - 1 \Rightarrow \tau_{t_l} \leq t_l$.
Finally, $\tau_t \leq t$ for all t. □

Theorem 1 (Deterministic Service) Transmission schedule $\mathcal{T}$ generated by the admission control algorithm guarantees deterministic service to all the clients serviced.

Proof: By Lemma 1 and 2, $D_i(t)$ guarantees no buffer overflow and no data delay. Hence, deterministic service is guaranteed. □

V. OPTIMALITY OF AGGREGATED SMOOTHING

In this section, we prove that the proposed aggregated smoothing scheme minimizes client buffer requirement and maximizes the number of clients that can be serviced simultaneously with the given link bandwidth.

Lemma 3 (Indispensable Prefetching) : Aggregated smoothing scheme prefetches future data when it is indispensable and prefetches only indispensible amount of data.

Proof: (1. Time) Aggregated smoothing scheme prefetches data at time $t-1$ if $\sum_i f_i(t) + P(t) > \mathcal{B}$. If we do not prefetch surplus data before time t, $D(t) = \sum_i D_i(t) = \sum_i f_i(t) + P(t) > \mathcal{B}$. Hence, $\tau_t > t$, which violates deterministic service constraint. Therefore, prefetching must be done at or before time $t-1$.
(2. Amount) Aggregated smoothing scheme prefetches exactly $\sum_i f_i(t) + P(t) - \mathcal{B}$ of data. If we prefetch less than that amount of data, $D(t) = \sum_i D_i(t) > (\sum_i f_i(t) + P(t)) - (\sum_i f_i(t) + P(t) - \mathcal{B}) = \mathcal{B}$. Hence, $\tau_t > t$, which violates deterministic service constraint. Therefore, at least $\sum_i f_i(t) + P(t) - \mathcal{B}$ of data should be prefetched. □

Theorem 2 (Minimal Buffer Requirement) : Aggregated smoothing scheme requires minimal amount of overall client buffer.

Proof: Straightforward from Lemma 3. □

Theorem 3 (Maximizing Number of Streams) : Aggregated smoothing scheme maximizes the number of clients serviced simultaneously for given resources.

Proof: Assume that a service request from client C_n is rejected by the proposed admission control algorithm. It is enough to prove that no other deterministic admission control algorithm accepts the request.
Case 1: Rejected at the first rejection point in the algorithm
Suppose that C_n is accepted by any other admission control algorithm. Since the proposed admission control algorithm prefetches only indispensable amount of data, the total prefetch amount at time 0 generated by the other algorithm, $P'(0)$, is no less than $P(0)$. Hence, the amount of data that should be transmitted at time 0 by the other algorithm, $D'(0) = P'(0) + \sum_i f_i(0) - \wp \geq P(0) + \sum_i f_i(0) - \wp > \mathcal{B}$. Therefore, $\tau_0 > 0$, which implies that the other algorithm violates deterministic service constraint. It means that the other admission control algorithm is not a deterministic admission control algorithm.
Case 2: Rejected at the second rejection point in the algorithm
Similar to the Case 1. Since $P'(t) \geq P(t)$ and $P(t) > \sum_i R_i(t)$, $P'(t) > \sum_i R_i(t)$. Hence, the other algorithm invokes client buffer overflow which means that the algorithm does not provide deterministic service.
Case 3: Rejected at the third rejection point in the algorithm
In this case, $\sum_{i\in\mathcal{B}} f_i(t+1) > \sum_{i\in\mathcal{B}} B_i + \mathcal{B}$. Hence, no admission control algorithm can transmit $\sum_{i\in\mathcal{B}} f_i(t+1)$ of data without delay.
Case 4: Rejected at the fourth rejection point in the algorithm
In this case, $P(t') = 0$ and $\sum_{t=t'-1}^{t+1} \sum_{i\in\mathcal{B}} f_i(t) > (t+1-t') \times \mathcal{B} + \sum_{i\in\mathcal{B}} B_i$. Similarly to case 3, no admission control algorithm can transmit $\sum_{t=t'-1}^{t+1} \sum_{i\in\mathcal{B}} f_i(t+1)$ of data without delay.
Hence, if a request is rejected by the proposed admission control algorithm, no other deterministic admission control algorithm can accept it. Therefore aggregated smoothing algorithm maximizes the number of streams serviced simultaneously. □

VI. CONCLUSION AND FUTURE WORKS

Previous works concerning smoothing of VBR streams focused on the smoothing of individual stream. Hence they did not take into consideration the effect of multiplexing multiple VBR streams. Multiplexing multiple VBR streams cuts down per stream variability, which helps to increase link utilization and number of clients serviced simultaneously. However smoothing using statistical multiplexing gain does not provide deterministic guarantee. In this paper, we presented a new smoothing scheme integrating per stream smoothing by prefetching and statistical multiplexing. The presented scheme not only provides deterministic service to all the clients serviced but also maximizes link utilization. The scheme regards all the multiplexed streams as one huge stream and deterministically smoothes the stream considering each client's buffer occupancy. We have proved that the proposed scheme minimizes client buffer requirement and maximizes the number of streams serviced simultaneously. This scheme can be used for VOD service not only in the hierarchical ATM network environment but also in the best-effort network such as ethernet. Currently, we are looking for more simple admission control algorithm maintaining optimality of the proposed algorithm.

REFERENCES

[1] J. M. McManus and K. W. Ross, "Video on demand over ATM: Constant-rate transmission and transport," *IEEE J. Select. Areas Commun.*, pp. 1087–1098, Aug. 1996.
[2] E. Knightly and H. Zhang, "Traffic characterization and switch utilization using deterministic bounding interval dependent traffic models," in *Proc. INFOCOM'95*, (Boston, MA), Apr. 1995, pp. 1137–1145.
[3] J. Salehi, Z.-L. Zhang, J. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," in *Proc. ACM SIGMETRICS'96*, (Philadelphia, PA), 1996, pp. 222–231.
[4] S. Kang and H. Y. Yeom, "Transmission of video streams with constant bandwidth allocation," *Computer Commun.*, vol. 22, no. 2, pp. 173–180, 1999.

[5] J. Zhang and J. Y. Hui, "Traffic characteristics and smoothness criteria in VBR video traffic smoothing," in *Proc. ICMCS'97*, (Ottawa, CA) , 1997, pp. 3–11.
[6] W.Feng and J. Rexford, "Performance evaluation of smoothing algorithms for transmitting prerecorded variable-bit-rate video," *IEEE Trans. Multimedia*, vol. 1, pp. 302–313, Sept. 1999.
[7] E. Knightly and H. Zhang, "Providing end-to-end statistical performance guarantees with bounding interval dependent stochastic models," in *Proc. ACM SIGMETRICS'94*, (Nashville, TN), May 1994, pp. 211–220.
[8] R. Izmailov and E. Ayanoglu, "Priority statistical multiplexing of mixed vbr video and cbr traffic in B-ISDNATM with a threshold algorithm," in *Proc. INFOCOM'93*, (San Francisco, CA), Mar. 1993, pp. 910–918.
[9] D. Reininger *et al.*, "Statistical multiplexing of VBR MPEG compressed video on ATM networks," in *Proc. INFOCOM'93*, (San Francisco, CA), Mar. 1993.
[10] H. M. Vin *et al.*, "A statistical admission control algorithm for multimedia servers," in *Proc. ACM Multimedia*, (San Francisco), Oct. 1994, pp. 33–40.
[11] J. Salehi *et al.*, "Smoothing, statistical multiplexing and call admission control for stored video," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 1148–1166, Aug. 1997.

Sooyong Kang is a full-time instructor in the Department of Computer Science Education, Hanyang University, Korea. He received his B.S. degree in mathematics and M.S. and Ph.D. degrees in computer science from Seoul National University in 1996, 1998, and 2002, respectively. His research interests include multimedia systems, especially multimedia data transmission system and multimedia storage system. Also he is interested in the security area including secure communication on the internet, intrusion detection system and attack detection system.

Heon Y. Yeom is an associate professor in the school of computer science and engineering, Seoul National University, Korea, since 1992. He received his B.S. from Seoul National University in computer science and statistics in 1984 and M.S. and Ph.D. from Texas A&M University in computer science in 1986 and 1992, respectively. From 1999 to 2002, he was also the CTO of SIGn Corporation, where he led the Firewall/VPN system implementation team. His research interests include multimedia systems, distributed systems, fault tolerant systems and security systems.