Optimal Real-time Admission Control Algorithms for the Video-On-Demand (VOD) Service

Frank Yeong-Sung Lin
Department of Information Management
National Taiwan University
Taipei, Taiwan, R.O.C.

Abstract—**In order to meet the Quality-Of-Service (QOS) requirements of the VOD (Video-On-Demand) service, and, on the other hand, to maximize the system throughput (revenue), it is essential that the admission control algorithm be carefully designed. In this paper, two new types of admission control schemes for the VOD service are proposed. They are the Enhanced Strict Admission Control (ESAC) and the Probabilistic Admission Control (PAC). In the ESAC schemes, we propose to use more statistics (of small amount and easily pre-calculated) than the peak frame size of the stored video information to strictly guarantee the QOS requirement and to achieve potentially much higher throughput. In the PAC schemes, we propose to use similar statistics as used in the ESAC schemes to achieve even higher throughput at the cost of some small and controllable likelihood of lost/overdue data. The admission control problems are formulated as feasibility problems where different systems of simultaneous equations are considered. For each admission control scheme, if the corresponding system of simultaneous equations has a feasible solution, then admit the call request; otherwise, reject the call. Special structures of the systems are identified so as to facilitate the development of optimal real-time admission control algorithms. Efficient optimal algorithms are also proposed to calculate the minimal buffer requirement for a given performance objective.**

I. INTRODUCTION

The Video-On-Demand (VOD) service [10], [20] has become feasible due to the availability of a number of enabling technologies, such as MPEG (Moving Picture Experts Group) [3], ATM (Asynchronous Transfer Mode) [4], [5], ADSL (Asymmetric Digital Subscribers Line) [6], [14] and HFC (Hybrid Fiber-Coax) with the maturity of two physical and medium access control protocol standards, i.e. IEEE 802.14 [7] and MCNS (Multimedia Cable Network System) [8]. The VOD service can provide viewers with the same quality and control as those of the VCR playback of rental programs but with a lot more flexibility and convenience. Therefore, many information service providers, network service providers, and Customer Premises Equipment (CPE) vendors are actively in the process of realizing the VOD service.

A number of possible system architectures to support the VOD service have been proposed [2], [10], [11], [12], [13], [17], [20], [21], [24]. It is commonly agreed that a high-speed backbone and broadband distribution/community networks will be used to transport video streams. More importantly, it is likely that video information is stored digitally, in a compressed form to save storage space and to reduce transport cost. This compression/decompression process typically requires transporting variable-bit-rate video streams, which is assumed the case in this paper and would make the development of efficient and effective admission control schemes more difficult.

A key challenge involved in providing the VOD service is the need to store a huge amount of video data in an archive such that each video stream can be accessed and transmitted to the display device in real time. Various video server architectures [1], [16], [18], [19], [22] have been proposed, e.g., the disk array architecture, the juke box and disk array architecture, and the distributed architecture.

In a video server with the disk array architecture, video information is stored in a high-speed disk array. When a video stream is to be played, it will be first brought into a buffer (e.g. RAM) and then sent to the viewer through a communication facility. In a video server with the juke box and disk array architecture, two types of storage are used. The entirety of the video information is stored in "juke boxes" – large and slower devices such as optical disks. Whereas, frequently accessed video information (usually a small portion of the information stored in the juke box) is stored in high-speed disk arrays to reduce the access time. When a miss on the disk array occurs, two cases are possible. In the first case, the requested video information is first brought from the juke box into the disk array and then sent to the buffer for transmission. In the second case, the requested video information is brought directly into the buffer and transmitted to the viewer. In a video server with a distributed architecture, video information are distributed among a number of servers which cooperatively support the service. In this paper, we consider the admission control problem under the disk array architecture. The result can be extended for other video server architectures.

For continuous retrieval of video data, it is essential that video information be available at the display device by the time of its playback. We refer to this as the *continuity requirement*. In order to satisfy the continuity requirement for each active video stream, admission control schemes should be carefully designed. Two types of admission control schemes have been proposed. They are strict admission control schemes [1], [16], [23] and predictive admission control schemes [2], [15]. The objective of strict admission control schemes is to guarantee that the continuity requirement be strictly satisfied by allocating system resources based on the worst-case scenario (i.e. considering the peak frame size). Whereas, predictive admission con-

Publisher Item Identifier S 0018-9316(98)09550-X

trol schemes attempt to reliably predict the future behavior of the system based upon its measured past behavior. Strict admission control schemes have the advantage of simplicity at the cost of low throughput. This can easily be understood by the example that the peak to mean ratio of coded frame sizes is 12 for the Star Wars film using the MPEG coding scheme [9]. If the worst-case engineering approach is taken, the system resources may be overly underutilized. To improve the throughput of the strict admission control schemes, predictive admission control schemes allow a given bound on the data loss/overdue probability and use measurements to estimate the expected behavior of the system. However, to collect real-time measurements incurs system overhead. In addition, the dynamics of video traffic typically make it insufficient to make admission control decisions based upon snapshot load conditions (it is not guaranteed that the data loss/overdue probability for each video stream be within the given bound).

In this paper, we consider two new types of admission control schemes, i.e. the Enhanced Strict Admission Control (ESAC) algorithms and the Probabilistic Admission Control (PAC) algorithms. The ESAC algorithms intend to improve the throughput of the strict admission control schemes by employing more and easily pre-calculated information than the peak frame size from the stored video information. Whereas, the PAC algorithms intend to reduce the on-line overhead incurred in the predictive admission control algorithms by using statistics available from analyzing (also off-line) the stored video information. In addition, unlike predictive admission control algorithms, the PAC algorithms guarantee that the data loss/overdue probability for each video stream be within a pre-specified bound.

In this paper we also consider the buffer management problem, where for a given system load the minimum amount of buffer requirement is calculated to serve such load. The buffer management algorithm can be applied either in the system design phase to calculate the minimum buffer requirement or on a real-time basis for dynamic buffer management.

The basic approach taken in this paper is to formulate each of the admission control problems as the feasibility problem of a discrete system, and to formulate the buffer management problems as combinatorial optimization problems. Special properties of such systems are identified. Those properties facilitate efficient optimal algorithms so that real-time applications can be supported.

The remainder of this paper is organized as follows. In Sections 2 and 3, the ESAC algorithms and the PAC algorithms are presented, respectively. Section 4 summarizes this paper.

II. THE ESAC ALGORITHMS

Assume that the disk scheduler services all video streams in a round-robin fashion. Refer to a *cycle* as the time of such a round. Let B be the size of the available buffer in terms of blocks. Let K be the block transfer rate from the disk array to the buffer. Let I be the set of the new and the existing video streams, where the admission control of the new video stream(s) is to be determined. Let A be the worst-case disk array access overhead (including seek time and latency) for all the streams in I in one cycle. For demonstration purposes, it is assumed that contiguous block allocation policies are adopted. Under such an assumption, A can easily be calculated from the worst-case seek time, the worst-case latency and the number of streams considered, which is independent of the number of blocks fetched for each stream in a cycle. In the cases where restricted contiguous block allocation policies or random block allocation policies are adopted, the problem formulation and algorithms could be modified accordingly.

Let T_{ij} be j if in each cycle j blocks of data for stream i is fetched from the disk array to the buffer and 0 otherwise. Let J_i be the set of possible numbers of blocks fetched in one cycle for stream $i \in I$. In the appendix, a method is proposed to help better select J_i. For each video stream i, pre-calculate d_{ij} which is the worst-case (minimum) playback duration for stream i when exactly j blocks are fetched from the disk in each cycle. Note that d_{ij} can be calculated easily from the stored video data by using the timestamps stored with the data and the following two possible methods. If the number of blocks fetched in each cycle for each admitted stream is kept the same for the entire duration of the call, then a "fixed-displacement window" with width j can be used to calculate d_{ij}. On the other hand, if the number of blocks fetched in one cycle for each admitted call may change with time (upon call admission and/or termination), then a "running window" with width j can be used to calculate d_{ij}. It is clear that for given i and j, the value of d_{ij} calculated by using the "fixed-displacement window" is no less than that calculated by using the "running window". Note also that d_{ij} needs to be calculated only once (off-line) and can be stored at the beginning of each video stream for admission control purposes. One example is give below to illustrate how d_{ij} is calculated. Consider a video stream i which is stored in a sequence a data blocks of a fixed size. The playback duration (in units of time) for each of the data blocks is assumed to be 1, 2, 3, 3, 2, 1, 1, 2, 3, 3, 2, 1, 1, 2, 3, 3, 2, 1, ..., respectively and in that order. When $j = 1$, $d_{ij} = \min\{1,2,3,3,2,1,1,2,3,3,2,1,...\} = 1$ for both window schemes. When $j = 2$, for the fixed-displacement window scheme, $d_{ij} = \min\{3,6,3,3,6,3,...\} = 3$, while for the running window scheme, $d_{ij} = \min\{3,5,6,5,3,2,3,5,6,5,3,...\} = 2$. It can also be easily verified that when $j = 3$, d_{ij} equals 6 and 4 for the fixed-displacement and the running window schemes, respectively.

Let g_i be the continuous function of playback duration with respect to the number of blocks fetched in each cycle time for stream i, constructed by using d_{ij} and linear interpolation (to connect each pair of points (j,d_{ij}) and $(j+1,d_{i(j+1)})$ with a segment). The construction of g_i as a continuous function is principally for the purpose of illustrating a number of properties to be introduced later, where linear programming relaxation of the integer decision variables are considered. For the running window scheme, it is clear that each g_i is monotonically increasing. A more strong result is given below.

Property 1: For the running window scheme, $g_i(a+b) \geq g_i(a) + g_i(b)\ \forall a, b \in N,\ i \in I$.

Proof: Consider any $(a+b)$ contiguous data blocks for stream i, whose playback duration is denoted by $t_1, t_2, t_3, ..., t_{a+b}$ without loss of generality. Then,

$$\begin{aligned}\sum_{k=1}^{a+b} t_k &= \sum_{k=1}^{a} t_k + \sum_{k=a+1}^{a+b} t_k \\ &\geq g_i(a) + g_i(b).\end{aligned}$$

The above relation holds for any choice of $(a+b)$ contiguous data blocks for stream i, and therefore also holds for the minimum among all possible choices, which is equal to $g_i(a+b)$. This completes the proof.

This property can be generalized to the case where the domain is the set of non-negative real numbers. The result implies that it is more effective (from the stand point of the average playback duration per data block) to fetch more data blocks in each cycle time for each stream, and will be used to show another useful property for algorithm development.

We formulate the admission control problem as the feasibility problem of System P1. If System P1 has a feasible solution, then admit the new call(s); otherwise, reject the call(s).

System P1:

$$\begin{aligned} A + \sum_{k\in I}\sum_{j\in J_k} \frac{T_{kj}}{K} &\leq g_i(\textstyle\sum_{j\in J_i} T_{ij}) \quad \forall i \in I & (1)\\ \sum_{i\in I}\sum_{j\in J_i} T_{ij} &\leq B & (2)\\ \sum_{j\in J_i} \frac{T_{ij}}{j} &= 1 \quad \forall i \in I & (3)\\ T_{ij} &= 0 \text{ or } j \quad \forall i \in I, j \in J_i. & (4)\end{aligned}$$

The left-hand-side of Constraint (1) is the worst-case cycle time. Constraint (1) requires that no starvation occur (the continuity requirement be satisfied) for every stream. Constraint (2) requires that the total amount of buffer allocated to the admitted call(s) not exceed a given value B. Constraints (3) and (4) require that for each stream $i \in I$ exactly one particular number of blocks (chosen from J_i) are fetched from the disk array in one cycle.

Despite of the integrality constraint (4), the following polynomial-time algorithm can be applied to determine the feasibility of System P1.

Algorithm A:

1. Set H to be a lower bound on the minimum buffer requirement for System P1 to be feasible (e.g. 0 or a possibly higher value by using the result provided in the appendix).
2. If H is greater than B, then report "System P1 is infeasible" and stop.
3. Fix the term $\sum_{i\in I}\sum_{j\in J_i} T_{ij}$ on the left-hand-side of Constraint (1) to be H.
4. Find for each $i \in I$ the minimum (single) $j \in J_i$ such that $A + H/K \leq g_i(\sum_{j\in J_i} T_{ij})$. Denote the solution by $\{T'_{ij}\}$.
5. Calculate $H' = \sum_{i\in I}\sum_{j\in J_i} T'_{ij}$.
6. If $H' \leq H$, then report "System P1 is feasible" and stop; otherwise, increase H by 1 and go to Step 2.

This algorithm exhaustively selects all possible values of B that satisfy Constraint (2) and examines whether Constraints (1), (3) and (4) can be satisfied at the same time. The above version of Algorithm A starts from a lower bound on the minimum buffer requirement and stops when a feasible buffer requirement is reached (if B is sufficiently large to make System P1 feasible). This provides an additional function to determine the minimum buffer requirement to service the entire offered load.

Although Algorithm A is a polynomial-time algorithm, for large B many iterations may be required to determine the feasibility of System P1, which may exclude Algorithm A from real-time applications. The following special properties of System P1 are identified, which facilitate a much more efficient heuristic procedure than Algorithm A to determine the feasibility of System P1.

Property 2: If the running window scheme is adopted and there exists a feasible solution to the linear programming relaxation of System P1, denoted by $\{\tilde{T}_{ij}\}$, such that $\sum_{i\in I}\sum_{j\in J_i} \tilde{T}_{ij} = \tilde{B} \leq B$, then there exists a feasible solution to the following system:

System P2:

$$(2),\ (3),\ 0 \leq T_{ij} \leq j\ \forall i \in I,\ j \in J_i \text{ and}$$
$$A + \frac{B}{K} \leq g_i(\sum_{j\in J_i} T_{ij}) \qquad \forall i \in I. \tag{5}$$

Proof: From the assumption, there exists a feasible solution to the following system:

System P3:

$$(2),\ (3),\ 0 \leq T_{ij} \leq j\ \forall i \in I,\ j \in J_i \text{ and}$$
$$A + \frac{\tilde{B}}{K} \leq g_i(\sum_{j\in J_i} T_{ij}) \qquad \forall i \in I. \tag{6}$$

Also from the assumption

$$A + \frac{\tilde{B}}{K} \leq g_i(\sum_{j\in J_i} \tilde{T}_{ij}) \qquad \forall i \in I. \tag{7}$$

Since $\tilde{B} \leq B$ from the assumption, Equation (7) implies that

$$\frac{\tilde{B}}{B}A + \frac{\tilde{B}}{K} \leq g_i(\sum_{j\in J_i} \tilde{T}_{ij}) \qquad \forall i \in I. \tag{8}$$

Multiplying both sides of Equation (8) by $B/\tilde{B}$ yields

$$A + \frac{B}{K} \leq \frac{B}{\tilde{B}} g_i(\sum_{j \in J_i} \tilde{T}_{ij}) \qquad \forall i \in I. \tag{9}$$

Since $B/\tilde{B} \geq 1$ and from the continuous version of Property 1, the right-hand-side of Equation (9) is less than or equal to $g_i(\frac{B}{\tilde{B}} \sum_{j \in J_i} \tilde{T}_{ij})$. Consequently,

$$A + \frac{B}{K} \leq g_i(\frac{B}{\tilde{B}} \sum_{j \in J_i} \tilde{T}_{ij}) \qquad \forall i \in I. \tag{10}$$

Equation (10) implies that the solution where for each stream i the buffer requirement is $\frac{B}{\tilde{B}} \sum_{j \in J_i} \tilde{T}_{ij}$ satisfies Constraint (5). In addition, the corresponding overall buffer requirement of this solution is equal to B, which makes Constraint (2) satisfied. Therefore, this solution is also feasible to System P2. Consequently, if the linear programming relaxation of System P1 is feasible then System P2 is feasible. This completes the proof.

It is also clear that if System P2 is feasible, then the linear programming relaxation of System P1 is feasible (Constraint (5) is more restricted than the corresponding constraint in the linear programming relaxation of System P1). Based upon this fact and Property 2, the following property is identified.

Property 3: When the running window scheme is adopted, the feasibility of the linear programming relaxation of System P1 can be determined by examining the feasibility of System P2. If System P2 is feasible then the linear programming relaxation of System P1 is feasible; otherwise, the linear programming relaxation of System P1 and also System P1 itself are infeasible.

The feasibility of System P2 can be determined in an efficient way. First, since for every $i \in I$, g_i is a monotonically increasing function (for the running window scheme), one can easily use standard line search techniques to calculate the minimum number of data blocks required for each stream i such that Constraint (5) is satisfied. If such calculated total buffer requirement is no greater than B then System P2 is feasible; otherwise, System P2 is infeasible. It is clear that this procedure exactly determines the feasibility of System P2. The following heuristic admission control algorithm is then developed accordingly.

Algorithm ESAC-HEU:

1. Find for each $i \in I$ the minimum (single) $j \in J_i$ such that $A + B/K \leq g_i(\sum_{j \in J_i} T_{ij})$. Denote the solution by $\{T'_{ij}\}$.
2. Calculate $B' = \sum_{i \in I} \sum_{j \in J_i} T'_{ij}$.
3. If $B' \leq B$, then admit the call(s); otherwise, reject the call(s).

This approach has the following advantages. First, it is simple. Compared with Algorithm A, Algorithm ESAC-HEU requires 1 iteration rather than in the worst case B iterations. Second, when a new call arrives, only one single inequality (in Step 1) needs to be examined so as to determine whether sufficient buffer is available to admit this call. Third, the buffer allocation for existing streams does not need to be changed when a new call is admitted. In addition, when a call terminates, the buffer allocated to this stream is simply freed. The number of blocks fetched in each cycle and the buffer allocation for each of the remaining streams do not need to be changed (still feasible). This property allows the fixed-displacement window scheme to calculate $\{d_{ij}\}$, which as explained previously would potentially result in better performance than the running window scheme that shall be adopted when the number of data blocks fetched in each cycle time may change. However, when the fixed-displacement window scheme is adopted, sequential search rather than more efficient line search techniques shall be used to calculate the minimum buffer requirement for each stream in Step 1 of Algorithm ESAC-HEU. This is because g_i may not be monotonically increasing when the fixed-displacement window scheme is used.

Under some circumstances, it is desired to minimize the buffer requirement. As discussed earlier, a variation of Algorithm A can be applied for this purpose. However, a significant number of iterations may be needed to find the minimum buffer required. The following property provides a more efficient way to calculate the minimum buffer requirement and forms a major part of a far more efficient optimal admission control algorithm than Algorithm A.

Property 4: Assume System P1 is infeasible for a given value of B, denoted by $\hat{B}$. Let B^* be the minimum value of B such that System P1 is feasible. Solve the following problem.

Problem P4:

$$\min \sum_{i \in I} \sum_{j \in J_i} T_{ij} \tag{11}$$

subject to:

(3), (4) and

$$A + \frac{\hat{B}}{K} \leq g_i(\sum_{j \in J_i} T_{ij}) \qquad \forall i \in I. \tag{12}$$

Denote the optimal solution to Problem P4 by $\{T'_{ij}\}$. Let B' be $\sum_{i \in I} \sum_{j \in J_i} T'_{ij}$. Then, $\hat{B} < B' \leq B^*$.

Proof: We first prove that $B' \leq B^*$. Let $\{T^*_{ij}\}$ be an optimal solution associated with the minimal buffer size B^* ($\sum_{i \in I} \sum_{j \in J_i} T^*_{ij} = B^*$). Then the following equation holds.

$$A + \frac{B^*}{K} \leq g_i(\sum_{j \in J_i} T^*_{ij}) \qquad \forall i \in I. \tag{13}$$

And for each $i \in I$, it is clear that the T_{ij} with the smallest index j such that (13) is satisfied is set to j and that all the other T_{ij}'s are set to zero. Since $\hat{B} < B^*$ from the assumption, for each $i \in I$, if T_{ij} with the smallest index j such that (13) is satisfied is set to j and all the other T_{ij}'s are set to zero, then the calculated buffer requirement is no greater than B^*. We next prove that $\hat{B} < B'$. Assume that $\hat{B} \geq B'$. Then $\hat{B}$ is feasible, which contradicts the assumption. This completes the proof.

Property 4 implies that if an infeasible value of B is chosen, then the substitution of this value into Problem P4 will lead to a larger total buffer requirement (or the same, only if B is optimal). This total buffer requirement is either infeasible or optimal. Due to such strict monotonicity before the minimum buffer requirement is calculated, the buffer requirement will converge from below to the optimal value using repeated substitution.

Based upon Property 4, an optimal algorithm to find the minimum buffer requirement is developed and presented below.

Algorithm ESAC-BM:

1. Choose a lower bound on the minimum buffer requirement as the initial value for B (e.g. using the result in the appendix).
2. Find for each $i \in I$ the minimum (single) $j \in J_i$ such that $A + B/K \leq g_i(\sum_{j \in J_i} T_{ij})$. Denote the solution by $\{T'_{ij}\}$.
3. Calculate $B' = \sum_{i \in I} \sum_{j \in J_i} T'_{ij}$.
4. If $B' = B$, then stop; otherwise, increase the value of B to B' and go to Step 2.

From the strict monotonicity implied by Property 4, the buffer size strictly increases iteration by iteration if not optimal, and will converge to an optimal solution. Compared with Algorithm A, to calculate the minimum buffer requirement B^*, Algorithm ESAC-BM typically requires much fewer iterations than Algorithm A due to a hopping search strategy rather than a sequential one.

The above result immediately leads to the following optimal real-time admission control algorithm.

Algorithm ESAC-OPT:

1. Apply Algorithm ESAC-BM to calculate the minimum buffer requirement B^*.
2. If B^* is no greater than the size of the available buffer (B), then admit the call(s); otherwise, reject the call(s).

Note that to improve the performance, Step 1 of Algorithm ESAC-OPT can be modified in such a way that if $B' > B$ at any intermediate stage of applying Algorithm ESAC-BM, then reject the call(s) and stop.

Algorithm ESAC-OPT is optimal in the sense that if and only if System P1 is feasible, the call(s) will be admitted. As Algorithm ESAC-BM is the major part of Algorithm ESAC-OPT, Algorithm ESAC-OPT is more efficient than Algorithm A as discussed previously.

III. THE PAC ALGORITHMS

As explained previously, the objective of the PAC approach is (i) to reduce the on-line overhead and (ii) to increase the system throughput at the cost of a small and controllable (and strictly satisfied) overdue probability. Let t_{ij} be the playback duration of j blocks of data for video stream i. Note that t_{ij} is a random variable. Let F_{ij} be the cumulative distribution function of t_{ij}. F_{ij} can be pre-calculated (off-line and only once) from the stored video data (timestamps) and the following two methods as introduced previously. If the number of blocks fetched in each cycle for each admitted stream is kept the same for the entire duration of the call, then the fixed-displacement window' with width j can be used to calculate F_{ij}. On the other hand, if the number of blocks fetched in each cycle for each admitted call may change with time (upon call admission and/or termination), then the running window with width j can be used. For the latter case one may, for example, apply the following method to obtain a conservative estimate of F_{ij}. Assume that stream i contains M blocks of data. Then, for each time threshold t, using the running window scheme, $M - j + 1$ windows will be examined and the number of windows in which the playback duration is less than or equal to t can be calculated. Denote this number by k. Then, $\frac{k}{\lfloor M/j \rfloor}$ is a conservative estimate of $F_{ij}(t)$. F_{ij} then can be stored at the beginning of video stream i for admission control purposes. Let S_i be the allowable overdue probability (tolerable degree of starvation) for stream i. Other notation has been introduced in the previous section.

Like the approach used to develop the ESAC algorithms, using the information of the playback duration distributions, we formulate the PAC admission control problem as the feasibility problem of System P5 shown below. If the system has a solution, then admit the call(s); otherwise, reject the call(s).

System P5:

$$\sum_{j \in J_i} \frac{T_{ij}}{j} F_{ij}(A + \sum_{k \in I} \sum_{j \in J_k} \frac{T_{kj}}{K}) \leq S_i \quad \forall i \in I \tag{14}$$

$$\sum_{i \in I} \sum_{j \in J_i} T_{ij} \leq B \tag{15}$$

$$\sum_{j \in J_i} \frac{T_{ij}}{j} = 1 \quad \forall i \in I \tag{16}$$

$$T_{ij} = 0 \text{ or } j \;\; \forall i \in I, j \in J_i. \tag{17}$$

The left-hand-side of Constraint (14) is the overdue probability for stream i. Constraint (14) requires that for each stream i the overdue probability be no greater than a given bound. Constraint (15) requires that the total amount of buffer allocated not exceed a given value B. Constraints (16) and (17) require that for each stream $i \in I$ exactly one particular number of blocks (chosen from J_i) are fetched from the disk array in one cycle.

To determine the feasibility of System P5, the following polynomial-time algorithm similar to Algorithm A can be applied.

Algorithm B:

1. Set H to be a lower bound on the minimum buffer requirement for System P5 to be feasible.

2. If H is greater than B, then report "System P5 is infeasible" and stop.
3. Fix the term $\sum_{i\in I}\sum_{j\in J_k} T_{kj}$ on the left-hand-side of Constraint (14) to be H.
4. Find for each $i \in I$ the minimum (single) $j \in J_i$ such that $\sum_{j\in J_i} \frac{T_{ij}}{j} F_{ij}(A + H/K) \le S_i$. Denote the solution by $\{T'_{ij}\}$.
5. Calculate $H' = \sum_{i\in I}\sum_{j\in J_i} T'_{ij}$.
6. If $H' \le H$, then report "System P5 is feasible" and stop; otherwise, increase H by 1 and go to Step 2.

Algorithm B is similar to Algorithm A and is in general not suitable for real-time applications. As in the case for System P1, the following efficient heuristic algorithm is therefore developed to determine the feasibility of System P5.

Algorithm PAC-HEU:

1. Find for each $i \in I$ the minimum (single) $j \in J_i$ such that $\sum_{j\in J_i} \frac{T_{ij}}{j} F_{ij}(A + B/K) \le S_i$. Denote the solution by $\{T'_{ij}\}$.
2. Calculate $B' = \sum_{i\in I}\sum_{j\in J_i} T'_{ij}$.
3. If $B' \le B$, then admit the call(s); otherwise, reject the call(s).

As discussed in the previous section, one might want to minimize the buffer requirement. The following property, which is similar to Property 4, provides an efficient way to calculate the optimal value of B.

Property 5: Assume System P5 is infeasible under a given value of B, denoted by $\hat{B}$. Let B^* be the optimal value of B. Solve the following problem.

Problem P6:

$$\min \sum_{i\in I}\sum_{j\in J_i} T_{ij} \tag{18}$$

subject to:

(16), (17) and

$$\sum_{j\in J_i} \frac{T_{ij}}{j} F_{ij}(A + \frac{\hat{B}}{K}) \le S_i \qquad \forall i \in I. \tag{19}$$

Denote the optimal solution by $\{T'_{ij}\}$. Let B' be $\sum_{i\in I}\sum_{j\in J_i} T'_{ij}$. Then, $\hat{B} < B' \le B^*$.

The proof for Property 5 is similar to that for Property 4 and is therefore omitted. Based upon Property 5, an efficient optimal algorithm to find the minimum buffer requirement is developed and presented below.

Algorithm PAC-BM:

1. Choose a lower bound on the minimum buffer requirement as the initial value for B.
2. Find for each $i \in I$ the minimum (single) j such that $\sum_{j\in J_i} \frac{T_{ij}}{j} F_{ij}(A + \frac{B}{K}) \le S_i$. Denote the solution by $\{T'_{ij}\}$.
3. Calculate $B' = \sum_{i\in I}\sum_{j\in J_i} T'_{ij}$.
4. If $B' = B$, then stop; otherwise, increase the value of B to B' and go to Step 2.

The above result also immediately leads to the following optimal real-time admission control algorithm.

Algorithm PAC-OPT:

1. Apply Algorithm PAC-BM to calculate the minimum buffer requirement B^*.
2. If $B^* \le B$, then admit the call(s); otherwise, reject the call(s).

Two properties of the PAC schemes are given below. First, when the tolerance of overdue probability S_i becomes larger, the system becomes less restricted and potentially higher system throughput/revenue can be achieved. Second, when every S_i is set to 0 such that the continuity requirement shall be strictly satisfied, then the PAC schemes become the ESAC schemes.

IV. SUMMARY

The VOD service has become popular. In order to provide satisfactory service to the subscribers and to maximize the throughput of the system or the revenue of the service provider, an efficient and effective admission control scheme is essential.

In this paper, we intend to improve the performance of two existing admission control schemes, i.e. the strict admission control and the predictive admission control. In the strict admission control scheme, the admission control decision is made base upon the peak frame size so that the continuity requirement is strictly satisfied. However, this scheme in many cases is overly conservative, specially when the frame sizes have a wide distribution. In the predictive admission control scheme, for the purpose of achieving higher throughput/revenue than the strict admission control scheme, the strict continuity requirement is relaxed and a pre-specified maximum data overdue probability is tolerable. However, this scheme requires real-time traffic measurement collection and analysis, and more importantly, cannot guarantee that the pre-specified data overdue probability threshold be not exceeded.

Two new admission control schemes are thus proposed in this paper to improve the performance and to alleviate the drawbacks of the aforementioned existing schemes. They are the Enhanced Strict Admission Control (ESAC) scheme and the Probabilistic Admission Control (PAC) scheme. In the ESAC scheme, we propose to use slightly more statistics than the peak frame size of the stored video information to achieve potentially much higher throughput than the strict admission control scheme, where the continuity requirement is still strictly guaranteed. The statistics data required for the implementation of the ESAC scheme are of small amount and can be easily pre-calculated (off-line and only once). In the PAC scheme, we propose to use similar statistics as used in the ESAC scheme to achieve even higher throughput. Compared with the predictive admission control scheme, the PAC scheme does not require real-time traffic measurement collection and analysis, and can guarantee that the maximum allowable data overdue probability be not exceeded.

The admission control problems are formulated as feasibility problems of the corresponding systems of simultaneous equations. If the system has a feasible solution, then admit the call(s); otherwise, reject the call(s). To determine the feasibility of the systems, we first identify special properties of the systems, and then develop both heuristic and optimal real-time admission control algorithms. One part of the optimal admission control algorithm can also be used to calculate the minimum buffer requirement for a given load. This optimal buffer management algorithm can either be applied in the system design phase or on a real-time basis for dynamic memory allocation due to its high efficiency.

References

[1] D.P. Anderson, Y. Osawa and R. Govindan. A File System for Continuous Media. *ACM Transactions on Computer Systems*, Vol. 11, No. 2, May 1993.

[2] D. Clark, S. Shenker and L. Zhang. Supporting Real-Time Applications in an Integrated Services Network: Architecture and Mechanism. *Proceedings of the ACM SIGCOMM*, pages 14-26, 1992.

[3] D. Le Gall. A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, Vol. 34, No. 4, pages 47-58, April 1991.

[4] R. Handel, M.N. Huber, and S. Schruder. *ATM Networks: Concepts, Protocols, Applications*, Addison-Wesley, 2nd Edition, 1994.

[5] U. Black. *ATM: Foundation for Broadband Networks*, Prentice Hall, 1995.

[6] M. Humphrey, J. Freeman and Paradyne Corporation. How xDSL Supports Broadband Services to the Home. *IEEE Network Magazine*, Vol. 11, No. 1, pages 14-23, January/February 1997.

[7] Institute of Electrical and Electronics Engineers, Inc. *IEEE Project 802.14/a Draft 3 Revison 3*, April 1998.

[8] Cable Television Laboratories, Inc. *Data-Over-Cable Service Interface Specifications - Radio Frequency Interface Specification*, October 1997.

[9] M.W. Garrett. Contributions Towards Real-Time Services on Packet-Switched Networks. Ph.D. Thesis, Columbia University, March 1993.

[10] A.D. Gleman, H. Kobrinski, L.S. Smoot, S.B. Weinstein, M. Fortier and D. Lemay. A Store-And-Forward Architecture for Video On Demand Service. *Proceedings of the IEEE ICC'91*, 1991.

[11] A.D. Gelman, S. Halfin, Walter Willinger. On Buffer Requirements for Store-And-Forward Video On Demand Service Circuits. *Proceedings of the IEEE GLOBECOM'91*, 1991.

[12] A.D. Gelman and L.S. Smoot. An Architecture for Interactive Applications. *Proceedings of the IEEE ICC'93*, 1993

[13] L. De Giovanni, A.M. Langellotti, L.M. Patitucci and L. Petrini. Dimensioning of Hierarchical Storage for Video on Demand Services. *Proceedings of the IEEE ICC'94*, 1994.

[14] T.R. Hsing, C.T. Chen and J.A. Bellisio. Video Communications and Services in the Copper Loop. *IEEE Communications Magazine*, January 1993.

[15] S. Jamin, S. Shenker, L. Zhang and D.D. Clark. An Admission Control Algorithm for Predictive Real-Time Service. *Third International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 349-356, November 1992.

[16] P. Lougher and D. Shepherd. The Design and Implementation of a Continuous Media Storage Server. *Third International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 69-80, November 1992.

[17] R. Ramarao, V. Ramamoorthy. Design of On-Demand Video Delivery Systems: The Spatio-Temporal Storage Allocation Problem. *Proceedings of the IEEE ICC'91*, 1991.

[18] P.V. Rangan and H.M. Vin. Designing File Systems for Digital Video and Audio. *Proceedings of the Thirteenth ACM Symposium on Operating Systems Principles*, pages 81-94, 1991.

[19] P.V. Rangan, H.M. Vin and S. Ramanathan. Designing an On-Demand Multimedia Service. *IEEE Communications Magazine*, Vol. 30, No. 7, pages 56-65, July 1992.

[20] W.D. Sincoskie. Video on Demand: Is it Feasible? *Proceedings of the IEEE GLOBECOM'90*, 1990.

[21] W.D. Sincoskie. System Architecture for a Large Scale Video on Demand Service. *Computer Networks and ISDN Systems*, Vol. 22, pages 155-162, 1991.

[22] L. Vaitzblit. The Design and Implementation of a High Bandwidth File Service for Continuous Media. MIT Thesis, September 1991.

[23] H.M. Vin and P.V. Rangan. Admission Control Algorithms for Multimedia-On-Demand Servers. *Third International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 56-68, November 1992.

[24] Y.C. Lai, Y.D. Lin and H.Z. Lai. A Hierarchical Network Storage Architecture for Video-on-Demand Services. *IEEE Transactions on Broadcasting*, Vol. 43, No. 2, pages 145-154, June 1997.

Appendix

In this appendix, a method to calculate J_i for each $i \in I$ is proposed. The objective is to find for each stream i the minimum set J_i without changing the feasibility of the problem.

The following property provides a lower bound on the smallest element in each J_i.

Property A1: Find the minimum positive integer j such that

$$A + \frac{j}{K} \leq d_{ij}. \tag{20}$$

The result is a lower bound on the smallest element in J_i.

Proof: (20) is the result of minimizing the buffer requirement subject to (1) where I is a singleton. It is clear that if I is enlarged to contain any other element(s), the same or larger j is required to satisfy (1). This completes the proof.

Property A1 can also be used to calculate a lower bound on the minimum buffer requirement by summing up such calculated lower bounds for individual streams. In addition, with the lower bound on the buffer requirement of each stream i from the above property, denoted by B_i, an upper bound on the buffer requirement for each stream i can also be calculated.

Property A2: For each stream $i \in I$, $B - \sum_{k \in I, k \neq i} B_k$ is an upper bound on its buffer requirement.