On Prioritizing the Delivery of Short Videos Clips

Chi-Kwong LI*, Tsun-Ping J. TO* and Chi-Kin LEUNG**

*Dept. of Electronic and Information Engineering, ** Centre for Multimedia Signal Processing.
The Hong Kong Polytechnic University, Hunghom, Hong Kong SAR, China.

Abstract-- In Interactive Video-on-Demand (IVOD) applications, service availability and response times are more visible to the user than the underlying data throughput. In this paper, a resource model for incoming request is proposed. Based on the resource model, the performance of several admission policies for short video requests is evaluated in terms of probability of admission, queuing delay, and throughput. Simulation results show that prioritizing requests can increase the number of admitted requests and reduce the queuing time. The technique proposed in the paper is independent of the underlying disk scheduling technique, hence it can be employed to improve the user-perceived performance of VOD servers under various system loading conditions.

Index Terms -- Interactive Video-on-demand, admission priority, request buffering.

I. INTRODUCTION

In a video-on-demand (VOD) server, four kinds of resources, namely the disks, memory buffers, processor time, and network bandwidth are all involved in data retrievals. The use of these resources will be scheduled by a scheduler in order to support timely retrieval of data [1]. Considerable amounts of work reported in the literature [2-7] concentrate on disk scheduling to achieve high data throughput as the main performance objective. Relatively few studies have been conducted on user-perceived performance issues, which are more important in interactive environments. VOD services need not be limited to long programs. Short video clips can be more prevalent in digital library applications involving interactive hypermedia databases. In these Interactive Video-On-Demand (IVOD) applications, maximizing the percentage of admitted requests and minimizing response time are more important user-perceived objectives.

Arrivals and completions of interactive video requests can be bursty or sparse at different times. Instead of rejecting arriving requests immediately, an IVOD server may buffer the requests in a request queue so that pending requests can be admitted on completion of existing ones. For a successfully serviced client request, the response time is the time between request generation and start of playout. The response time is composed of round-trip network latencies, queuing time at the server prior to admission, and start-up delay at the server. When a server is lightly loaded, queuing time is negligible. When a server is heavily loaded, arriving requests will wait in a request queue until the completion of some previously admitted requests. The response time will depend on the amount of time a request spent in the queue.

Since the playout rate and the length of interactive requests can differ by large values, priority schemes based on these parameters can be imposed to implement an admission policy to sequence the admission of requests. In this paper, we study the effects of admission policies. We shall propose a new admission policy that can significantly reduce the response time while the throughput is maintained. Simulation studies are conducted to demonstrate the overall performance advantages of the proposed admission policy.

II. MODEL AND PROBLEM

A typical VOD system consists of clients and servers on a network. Each client may generate real-time stream requests for video or audio data. The client requests are sent to the VOD servers via a data network, as shown in Fig. 1 below.

![Figure 1 - System architecture of VOD](image)

The resources in our VOD server model consist of one or more disk drives, a pool of memory buffers and at least one CPU to execute the scheduler. The scheduler controls the sequence and sizes of disk accesses, manages the memory buffers, and invokes periodic functions to transmit data to the clients. The VOD scheduler performs an admission test to ensure that non-starvation guarantees can be extended to the ongoing services before new requests can be admitted. When new requests are admitted or when ongoing services have been completed, the scheduler must decide how resources are to be reallocated.

The VOD server services clients in turns by storing client requests into memory buffers. As more clients are admitted, the time spent by a server in completing one round of service, referred to as the service period, increases. The underlying disk scheduling technique employed to sequence the order of request admission will impose a limit on the server throughput. Such limit is enforced by the admission control mechanism of the...
server. On the admission of new requests, the amount of read ahead and the associated buffers allocated to each client need to be adjusted [1,9]. Once admitted, a request for a motion video segment becomes a continuous delivery of video data and is often referred to as a video stream, or stream.

At low request arrival rates, user requests are promptly entertained because the server has sufficient capacity. At high arrival rates, the admission control mechanism of a server will not allow the server to accommodate all arriving requests. When a new request cannot be admitted upon arrival, it will not be admitted until at least one existing stream completes. In an IVOD environment where stream lengths are short and stream departures are frequent, there is a high probability that within a certain time the server will have reclaimed significant amounts of resources from departing streams. Therefore, instead of rejecting arriving requests immediately upon their arrivals, an IVOD server can maintain a request queue to buffer arriving requests. This arrangement can enable the server to allocate bandwidth returned by completed streams more efficiently.

When the request queue becomes full and a new request arrive, the server may either reject the new request, or admitted it by ejecting a pending request from the request queue. Rejection delay refers to the response time experienced by users whose requests are unfortunately ejected after having waited in the request queue for some time.

The lifetime of a request that has been admitted into the system can be modeled as four distinct phases, namely the queuing phase, the start-up phase, the play phase and the departure phase, as shown in Fig. 2. The queuing phase denotes the period from arrival to successful admission. Its length is referred to as the admission delay. Upon admission, a request enters the start-up phase and waits for service. The play phase begins when delivery starts until the stream is completely read. The departure phase denotes the period in which service involves delivery of buffered data.

Figure 2 - Lifetime of an admitted request

While the performance of a VOD server is often measured by its data-rate throughput, the admission ratio, which measures the percentage of requests that are admitted, is actually more significant from a user’s point of view. In interactive applications, both the admission ratio and the admission delay are experienced directly by the users, and hence should be treated as important performance metrics. We identify the following performance objectives for an IVOD system:

- Maximizing the percentage of requests admitted,
- Minimizing the admission delay,
- Minimizing the rejection delay, and
- Maintaining the data throughput.

III. RELATED WORKS

Many techniques have been proposed for disk scheduling in VOD servers. They are mostly after-admission disk service sequencing techniques. The round-robin algorithm [2] is simple to implement, but can only achieve moderate stream throughputs because disk overheads are not minimized. The disk seek overhead are minimized by seek-reducing approaches such as the SCAN, or elevator seek [3], approach. However, as the order of service is reversed by SCAN in alternating service rounds, the buffer requirement is necessarily doubled [1]. To some extent this problem is alleviated by more sophisticated seek-reducing schemes such as GSS [4] and PS S [5]. Deadline-driven scheduling approaches [6,7] have also been proposed previously for real-time retrieval of VOD data.

Under the assumptions that videos are long and a good fraction of client requests are for popular videos, the batching technique [8] lets multiple clients share a single data stream from the server. To improve the client capacity, a batching server needs to align the starting times of clients. The clients’ waiting time will increase with the batching factor and is quite unfavorable to interactive short videos. With the interval caching technique [9], the data brought in by a stream is cached for reuse by a closely following stream. As concurrent accesses to the same video clip can be infrequent, the interval caching policy may not be effective unless abundant memory is available. Performance in terms of the number of requests admitted, and the issue of long response time has not been fully addressed.

IV. PROCESSING OF PENDING REQUESTS

When pending requests cannot be accommodated by a finite queue, some requests need to be rejected. If a server maintains an excessively large queue, the time a request can wait in the queue can easily be longer than what users can tolerate. Therefore, the request queue need not be very long. If new requests are immediately rejected from entering a full queue, new admissible requests will not be considered at all for admission. In this paper, we choose to reject the oldest request from the full queue when a new request arrives. This rejection policy is also a fair one in the sense that each request will be given a fair chance to be considered for admission. A model to characterize the workload that is presented by a request to the VOD server will be established next. With this model, the performance tradeoffs among different admission policies are also explained.

A. The Bandwidth-Time-Product workload model

As an IVOD server switches services from one stream to the next, disk seek and rotation latencies are incurred. We refer to such loss of disk bandwidth as service overhead. An admission policy that favors requests with higher rates will admit fewer requests but deliver a higher data throughput. Therefore, a trade-off between admission ratio and the data throughput exists.

Service to a newly admitted stream can only start after the needed buffers are reclaimed from departing stream(s). Furthermore, before service can be given to the new stream, service may need to be given to existing
stream. Therefore, when disk bandwidth no longer used by a completed stream is granted to a new stream, the reserved disk bandwidth is not utilized. We refer to such loss of disk bandwidth as **transient overhead**. As the lengths of videos are typically short in interactive applications, stream admissions and departures can become quite frequent. If the admission policy favors longer streams, the frequency of transients can be reduced and the throughput will be improved at the expense of longer admission delay. Therefore, a trade-off between admission delay and data throughput exists.

A video request $V_i$ can be characterized by its length (in time) $l_i$ and its rate (or bandwidth) $r_i$. In the absence of overheads, its net resource requirement is the **Bandwidth-Time-Product (BTP)**, or $r_i l_i$, which is numerically equal to the data size of the request. The BTP can be represented by a rectangle in a two-dimensional space of bandwidth $b$ and time $t$. The service overhead $SOH_i$ increases the resource requirement of $V_i$ in the bandwidth dimension. The transient overhead $TOH_i$ increases the resource requirement of $V_i$ in the time dimension. The total resource requirement of request $V_i$ is represented by its **BTP workload**, shown as the $(SOH_i + r_i)(TOH_i + l_i)$ sized rectangle in Fig. 3.

![Figure 3 - BTP workload model of a video request $V_i$](image)

B. **Admission policies**

In this sub-section we shall examine the effects of four admission policies on the performance of a VOD scheduler.

1) **First-Come-First-Served (FCFS)**

   In this policy, the server will consider admitting only the first (or the oldest) pending request in the request queue. When the request queue is full, all new requests will be rejected upon arrival. Rejection delay is shortest when the server is overloaded. Unfortunately, admissible low-rate requests can easily be blocked by the presence of any inadmissible high-rate requests arrived earlier, resulting in considerably long admission delays.

2) **First-Fit (FF)**

   The oldest request in the queue with a bandwidth requirement no higher than the remaining available bandwidth will be admitted first. Therefore, high-rate requests will less likely be accommodated when the server is over-loaded. As the system unintentionally favors low-rate requests, more streams are admitted and the admission ratio is likely higher than FCFS.

3) **Shortest-Length-First (SLF)**

   Regardless of their rates, requests with shorter lengths will be considered for admission first. As pending requests in the queue can be cleared faster, the admission ratio will be improved and the admission delay will be reduced.

4) **Highest-Aspect-Ratio First (R/L)**

   In order to prioritize requests with fair weights given to rate and length parameters, the rate $r_i$ and length $l_i$ can be combined as the **Aspect Ratio (AR)** with $AR_i = r_i / l_i$. Fig. 4 illustrates how some requests are ranked by AR values. Admission prioritized by the AR parameter will favor high rate requests among those with equal length; and short rate requests among those with equal rate. In ranking requests that differ in both rate and length, the AR ranking policy is unbiased because neither parameter will dominate the other.

![Figure 4 - Ranking of requests by AR values](image)

V. **EXPERIMENTAL EVALUATION**

In this section we shall evaluate the effects of admission policies on the performance of an IVOD server. Each of our experiments spans the system life cycle from startup to shutdown over a period of 4000 seconds. The arrival rate of requests is varied to reveal performance of the server under a wide range of loads.

A. **Simulation model & experimental setup**

We model three types of resources in an IVOD server: the disk, the memory buffers and the CPU. A discrete-event simulation program, written in Simscript, is used to simulate the system. Each active entity in the system is modeled by one process, which includes a disk scheduler, a buffer manager, an admission controller, a stream request generator and the admitted streams. A media server is modeled by one process, which includes a disk scheduler, a memory buffer manager, and a CPU. The sizes of the disk, the memory buffers and the CPU are set to be the same. The sizes of the disk, the memory buffers and the CPU are set to be the same.

We model three types of resources in an IVOD server: the disk, the memory buffers and the CPU. A discrete-event simulation program, written in Simscript, is used to simulate the system. Each active entity in the system is modeled by one process, which includes a disk scheduler, a buffer manager, an admission controller, a stream request generator and the admitted streams. A media server is modeled by one process, which includes a disk scheduler, a memory buffer manager, and a CPU. The sizes of the disk, the memory buffers and the CPU are set to be the same. The sizes of the disk, the memory buffers and the CPU are set to be the same.

In our experiments, we model the disk as a sequence of disk sectors. Each disk sector is allocated to a disk controller, and the controller is responsible for managing the disk space. The disk controller is modeled by one process, which includes a disk scheduler, a buffer manager, and a CPU. The sizes of the disk, the memory buffers and the CPU are set to be the same. The sizes of the disk, the memory buffers and the CPU are set to be the same.

In our experiments, we model the disk as a sequence of disk sectors. Each disk sector is allocated to a disk controller, and the controller is responsible for managing the disk space. The disk controller is modeled by one process, which includes a disk scheduler, a buffer manager, and a CPU. The sizes of the disk, the memory buffers and the CPU are set to be the same.

In our experiments, we model the disk as a sequence of disk sectors. Each disk sector is allocated to a disk controller, and the controller is responsible for managing the disk space. The disk controller is modeled by one process, which includes a disk scheduler, a buffer manager, and a CPU. The sizes of the disk, the memory buffers and the CPU are set to be the same.

In our experiments, we model the disk as a sequence of disk sectors. Each disk sector is allocated to a disk controller, and the controller is responsible for managing the disk space. The disk controller is modeled by one process, which includes a disk scheduler, a buffer manager, and a CPU. The sizes of the disk, the memory buffers and the CPU are set to be the same.

In our experiments, we model the disk as a sequence of disk sectors. Each disk sector is allocated to a disk controller, and the controller is responsible for managing the disk space. The disk controller is modeled by one process, which includes a disk scheduler, a buffer manager, and a CPU. The sizes of the disk, the memory buffers and the CPU are set to be the same.

In our experiments, we model the disk as a sequence of disk sectors. Each disk sector is allocated to a disk controller, and the controller is responsible for managing the disk space. The disk controller is modeled by one process, which includes a disk scheduler, a buffer manager, and a CPU. The sizes of the disk, the memory buffers and the CPU are set to be the same.

In our experiments, we model the disk as a sequence of disk sectors. Each disk sector is allocated to a disk controller, and the controller is responsible for managing the disk space. The disk controller is modeled by one process, which includes a disk scheduler, a buffer manager, and a CPU. The sizes of the disk, the memory buffers and the CPU are set to be the same.

In our experiments, we model the disk as a sequence of disk sectors. Each disk sector is allocated to a disk controller, and the controller is responsible for managing the disk space. The disk controller is modeled by one process, which includes a disk scheduler, a buffer manager, and a CPU. The sizes of the disk, the memory buffers and the CPU are set to be the same.

In our experiments, we model the disk as a sequence of disk sectors. Each disk sector is allocated to a disk controller, and the controller is responsible for managing the disk space. The disk controller is modeled by one process, which includes a disk scheduler, a buffer manager, and a CPU. The sizes of the disk, the memory buffers and the CPU are set to be the same.
requests have a mean value of 30 seconds, whereas the rates of the requests have a mean value of 180 Kbytes/sec (the MPEG1 rate). Client requests are generated as Poisson arrivals. Size of the request queue is 32. A disk scheduler [7] called GS_EDF was employed as the underlying disk scheduler.

B. Results

![Admission ratio vs arrival rate](image)

**Figure 5 - Admission ratio vs arrival rate**

![Admission delay vs arrival rate](image)

**Figure 6 - Admission delay vs arrival rate**

![Rejection delay vs arrival rate](image)

**Figure 7 - Rejection delay Vs arrival rate**

Our experiments are conducted with request arrival rates ranging from 0.5 to 4 arrivals per second, for a buffer capacity of 8 Mbytes. The admissions policies FCFS, FF, SLF and R/L are evaluated. The two performance metrics, namely admission ratio and admission delay, are shown in Fig. 5 and Fig. 6, respectively. The performance of FCFS is poor because its admission ration is lowest and admission delay is highest. Performance of SLF and R/L is significantly better than that of FF. From Fig. 7 we can find that R/L gives a rejection delay significantly shorter than that of SLF, particularly at the peaks. Therefore, compared to the commonly employed FCFS and FF policies, our R/L admission policy gives better performance in almost every aspect, namely admission ratio, admission delay and rejection.

VI. CONCLUSION

An IVOD server is subject to fluctuating arrivals of requests over time. When an IVOD server is overloaded with arriving requests, the quality of service perceived by a user are: the chance of receiving service, the promptness of service and response. Since an admission policy is independent of the underlying disk scheduling technique, it can be dynamically selected by the server to suit various load conditions. Therefore, an admission policy will be effective in improving response time of interactive video servers that operate under fluctuating load. In this paper, we study the performance of several admission policies that prioritize the admission of incoming requests. We also propose a new admission policy called R/L. Based on the simulation results, it is shown that the R/L admission policy is able to give better performance in terms of admission ratio, admission delay, and rejection when compared to the other policies.

VII. REFERENCES