Enhanced Delivery of On-Demand Video Streams to Heterogeneous Receivers

Bashar Qudah and Nabil J. Sarhan
The Department of Electrical and Computer Engineering
Wayne State University
Detroit, MI, 48202 USA
e-mail: {bqudah,nabil}@wayne.edu

Abstract

The number of video streams that can be serviced concurrently is highly constrained by the required real-time and high-rate transfers of multimedia data. Resource sharing techniques can be used to address this problem. We study how heterogeneous clients can be supported while delivering data in a client-pull fashion using enhanced resource sharing. We propose three *hybrid solutions* for supporting heterogeneous download bandwidth. The first solution simply combines existing resource sharing techniques and deals with clients as two bandwidth classes. The other two solutions, however, classify clients into multiple bandwidth classes and service them accordingly by capturing the proposed ideas of *Adaptive Stream Merging* or *Enhanced Adaptive Stream Merging*, respectively. Moreover, we consider heterogeneity in the available client buffer space. Furthermore, we study how request scheduling can be adapted to the heterogeneous environment so as to exploit the variations in client bandwidth and buffer space. We evaluate the effectiveness of the proposed solutions and analyze various scheduling policies through extensive simulation.

0.1 Introduction

Recently, video streaming services have grown dramatically in popularity. Unfortunately, the number of video streams that can be delivered concurrently is highly constrained by required real-time and high-rate transfers of multimedia data. Resource sharing techniques [5, 9, 10, 13, 14, 21, 16, 20] address this challenge by utilizing the multicast facility. *Batching* [1, 9, 26], the simplest of these techniques, services all waiting requests for a video using one stream. Stream merging techniques further reduce the delivery costs by aggregating requests into larger groups that share the same multicast streams. These techniques include *Patching* [6, 14], *Transition Patching* [5, 7], and *Earliest Reachable Merge Target* (ERMT) [10]. Whereas Batching and stream merging techniques deliver data in a *client-pull* fashion, periodic broadcasting techniques [18, 15, 17] employ the *server-push* approach. In particular, they divide each supported video into multiple segments and broadcast them periodically on dedicated channels. Thus, they significantly reduce the server bandwidth requirements but can be used only for the most popular videos, and they require clients to wait until the next broadcast times of the first segments. Moreover, server channels may become underutilized when videos are requested infrequently. This paper considers the client-pull approach.

Efficient support for clients with heterogeneous resources, such as download bandwidth and available buffer space, is required so as to provide successful video streaming services over the Internet. Measurements of Internet client bandwidth [4] show that the bandwidth varies significantly not only from one technology (e.g., Cable, DSL, and Dial-up) to another, or from one Internet Service Provider (ISP) to another, but also from one client to another within the same ISP. Client physical location, and even the connection time, can play a significant role in the actual bandwidth. The available client buffer space also varies significantly, considering the wide variety of devices that support video streaming.

Resource sharing has been studied extensively, but to our knowledge, no study has addressed the client heterogeneity issue in systems employing the client-pull delivery approach. Batching requires each client to have one download channel at the video playback rate, whereas stream merging techniques generally require two. *Bandwidth Skimming* [11] allows the number of download channels to be non-integer through special video encoding or complex data delivery, but still assumes receivers homogeneity. Client heterogeneity has not been addressed except by a few, recent studies, such as HeRo [3], BroadCatch [25], and OHPB [24], which use the server-push approach. However, clients with bandwidth less than double the video playback rate can still face significant startup delay times compared to the video length with all three techniques.

This paper studies how heterogeneous receivers can be supported while delivering video streams in a client-pull fashion. We propose three solutions: *Simple Hybrid Solution* (SHS), *Adaptive Hybrid Solution* (AHS), and *Enhanced Hybrid Solution* (EHS). SHS simply combines Batching with either Patching (SHS-P) or ERMT (SHS-E). Batching is used for clients with bandwidth less than $2r$ (where r is video playback rate in bits per second) and Patching/ERMT is used for the rest. In contrast, AHS and EHS classify clients into multiple bandwidth classes and service them accordingly. AHS employs a new stream type, called *adaptive stream*, and EHS employs an *enhanced adaptive stream* type to serve clients with bandwidth capacities between r and $2r$. AHS and EHS employ adaptive streams or enhanced adaptive streams in conjunction with Batching and Patching or ERMT, leading to

four possible schemes: AHS-P, AHS-E, EHS-P, and EHS-E. We also study how to address the variations in client bandwidth during a session. In addition, we study the support for heterogeneity in the available client buffer space. Moreover, we introduce the concept of a *virtual queue* and discuss how scheduling policies can be adapted to capture the variations in client bandwidth and buffer space. Furthermore, we outline a comprehensive solution for client heterogeneity that utilizes layered video coding.

We evaluate the effectiveness of the proposed schemes and analyze various scheduling policies through extensive simulation. We also study the impacts of many parameters, including client bandwidth and buffer space distributions, server capacity, request arrival rate, customer waiting tolerance, number of videos, and video length. We consider two service models: *True Video-on-Demand* (TVOD) and *Near Video-on-Demand* (NVOD). In TVOD, we compare the server bandwidth required to service all requests immediately. In NVOD, however, we fix server resources and consider primarily five performance metrics: customer defection (i.e., turn-away) probability, average waiting time, unfairness against unpopular videos, unfairness against clients with low bandwidth, and unfairness against clients with low buffer space.

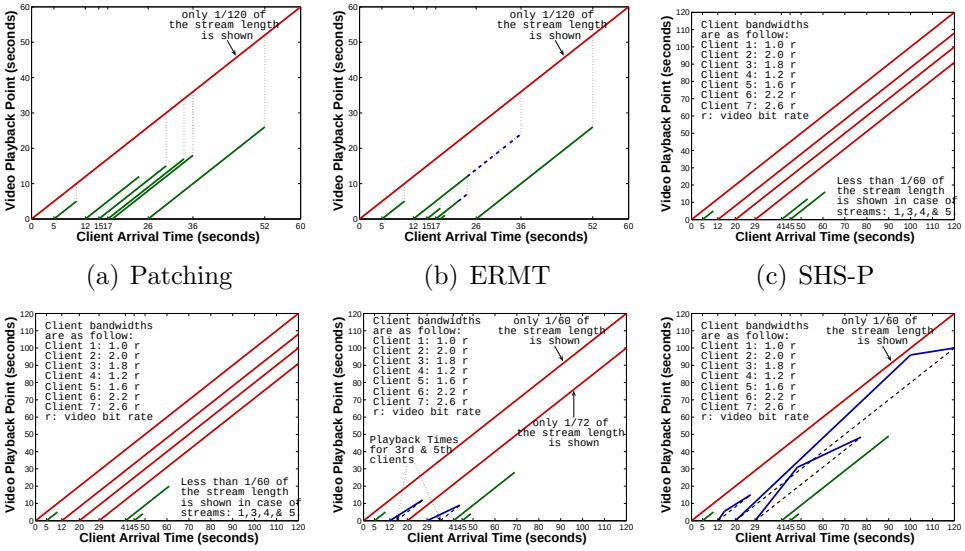
The rest of this paper is organized as follows. Section 0.2 provides background information. Sections 0.3 and 0.4 discuss the proposed support for heterogeneous client download bandwidth and available buffer space, respectively. Subsequently, Section 0.5 discusses request scheduling in the heterogeneous environment. Section 0.6 briefly discusses a comprehensive delivery solution. Finally, Section 0.7 discusses the performance evaluation methodology and main results.

0.2 Background Information

0.2.1 Resource Sharing

Let us discuss now the main resource sharing techniques that can be combined to support heterogeneous clients: Batching, Patching, and ERMT. While Batching services all waiting requests for a video using one full multicast video stream and requires only one client download channel at the video playback rate. Patching expands the multicast tree dynamically to include new requests. A new request joins the latest *regular* (i.e., full) stream for the object and receives the missing portion as a *patch*. Hence, it requires two download channels (each at the video playback rate) and additional client buffer space. When the playback of the patch is completed, the client continues the playback of the remaining portion using the data received from the multicast stream and already buffered locally. To avoid the continuously increasing patch lengths, regular streams are retransmitted when the required patch length for a new request exceeds a pre-specified value, called *regular window* (W_{reg}). Figure 1(a) further explains the concept of Patching in servicing a 2-hour video.

ERMT is a near optimal hierarchical stream merging technique. It also requires two download channels (each at the playback rate), but it makes each stream sharable and thus leads to a dynamic merge tree. A new client joins the closest reachable stream (*target*) and receives the missing portion by a new stream (*merger*). Reachable means the merge can occur before the target terminates. After the merger stream finishes and merges into the



(a) Patching (b) ERMT (c) SHS-P

(d) SHS-E (e) AHS-E (f) EHS-E

Figure 1: Stream Merging Techniques

target, the later can get extended to satisfy the playback requirement of the new client(s), and this extension can affect its own merge target. For example, in Figure 1(b), the third stream length got extended twice; once when the fourth stream merged into it, and once when the fifth did.

0.2.2 Request Scheduling

To facilitate scheduling, a waiting request queue is maintained for each video. All requests in a selected queue can be serviced using only one stream, whenever available resources become available. The main scheduling policies include *First Come First Serve* (FCFS) [9], *Maximum Queue Length* (MQL) [9], *Maximum Factored Queue Length* (MFQL) [1], and *Minimum Cost First* (MCF) [23]. FCFS selects the queue with the oldest request, whereas MQL selects the longest queue, and MFQL selects the queue with the *largest factored length*. The factored length is the queue length divided by the square root of the relative access frequency of its corresponding video. In contrast, MCF is a recently proposed policy that captures the variations in stream lengths by selecting the queue requiring the least cost. We study in this paper the preferred implementation of MCF, called *MCF-P*, which considers the cost per request.

0.3 Supporting Heterogeneous Download Bandwidth

We discuss next three proposed solutions for supporting heterogeneous client download bandwidth. Subsequently, we discuss how to control optimally regular streams in the proposed schemes and how to handle variations in the client download bandwidth during the session's lifetime. We assume that the video playback rate is r bits per second.

0.3.1 Simple Hybrid Solution (SHS)

Batching and all existing stream merging techniques assume client bandwidth homogeneity. Batching requires the client download bandwidth to be at least r (corresponding to one channel at the video playback rate), whereas stream merging techniques generally require at least $2r$. A simple and straightforward solution to support heterogeneous clients is to combine resource sharing techniques. We consider two alternatives: *Patching with Batching* (SHS-P) and *ERMT with Batching* (SHS-E). ERMT is the most efficient and Patching is the simplest among all stream merging techniques that require client bandwidth of $2r$ [19].

Figures 1(c) and 1(d) further explain how SHS-P and SHS-E work. Clients (or group of clients) with bandwidth capacities of $2r$ or greater ($2nd$, $6th$, and $7th$ in the figures) are serviced with Patching or ERMT, while those with less bandwidth ($1st$, $3rd$, $4th$, and $5th$) are serviced using Batching. Batching streams can eliminate the need for regular streams in Patching and reduces the average stream lengths in both Patching and ERMT, which results in a lower cost of service than a solution using two separate systems, each supporting a different class of clients.

0.3.2 Adaptive Hybrid Solution (AHS)

SHS classifies clients into two bandwidth classes: clients with bandwidth equal to or higher than r but lower than $2r$, and clients with bandwidth capacities equal to or higher than $2r$. SHS services the clients in the first class using Batching and the clients in the second class with Patching or ERMT, depending on the scheme used. Hence, it does not capture important opportunities for resource sharing in the first class. For the second class, utilizing more than double the playback rate (i.e., $2r$) is not expected to lead to worthwhile performance benefits. The results in [12] show that utilizing more than two channels (each at the playback rate) at each client leads to only low additional performance benefits when optimal stream merging (or ERMT) is used. Figure 2 shows the lower bound for server bandwidth requirement in serving a single video in a TVOD fashion to homogeneous clients using stream merging. In the heterogeneous environment, not every client has more than $2r$ in bandwidth and thus the potential gain is even less significant. Let us now discuss how to utilize the extra client bandwidth in the first class.

We propose here the idea of *Adaptive Stream Merging*. A new stream type, called *adaptive stream*, or succinctly *a-stream*, is introduced to service those clients with bandwidth higher than r but lower than $2r$. The client uses r of bandwidth to listen to the latest batching stream for the video and uses its remaining bandwidth to receive the missed portion as an a-stream at a slower rate than the video playback. The a-stream can be multicast when more than one request is waiting for the video. This stream replaces the costly batching stream used for such a client or a group of clients in the previous solution. Adaptive streams are adaptive to both the client available resources and server load. To achieve the adaptability to client resources, the delivery rate of an a-stream is determined based on the client with the lowest bandwidth in the group that has been selected for service. The adaptability to server load is achieved by triggering a-streams only when global performance optimization can be attained.

Let us now discuss in more detail the adaptability to server load. Obviously, since the

missed data are received at a slower rate than the video playback, clients need to buffer data for some time before the playback can start. The additional waiting time due to buffering, T_{buf}, depends on the size of data to be delivered and the delivery rate. Let us assume that a group of clients are admitted to the system P seconds after the last batching stream, the lowest bandwidth in the group is b, $r < b < 2r$, and $B = \frac{b}{r}$. Then, T_{buf} is given by

$$T_{buf} = (\frac{2 - B}{B - 1}).P. \quad (1)$$

Since T_{buf} is predictable, clients can be encouraged to wait by informing them with the expected time of playback (which is roughly equal to T_{buf}).

The required buffering has two conflicting effects. In particular, it leads to increasing data sharing among streams and thus servicing more clients and allowing resources to become available sooner for servicing new requests. It, however, increases the waiting time for some clients and may in turn cause them to defect. The overall server performance in terms of client defection rate and average waiting time depends on the combined effect. Hence, Adaptive Stream Merging triggers an a-stream only when the longest total client waiting time (including the required buffering time) will not exceed a certain *threshold*, which is equal to the mean client waiting tolerance times a certain factor. This factor can take any positive real value, and it varies dynamically based on the variation in client defection rate so as to achieve the lowest possible defection rate. If the triggering condition is not met, a batching stream is delivered instead. As will be shown later, the dynamic approach reduces not only the defection probability but also the average waiting time. Figure 4 further illustrates the impact of the waiting threshold on system performance under the default workload discussed in Section 0.7 and summarized in Table 1. The available server capacity is assumed to be $1000r$.

To further reduce the additional waiting time caused by buffering, the clients can perform early snooping. Basically, each client can start snooping on the latest batching stream for the video as soon as it issues the request as opposed to waiting till it gets admitted for service. In this case, P in Equation 1 becomes the individual client arrival time rather than the time of admission for service.

The proposed *Adaptive Hybrid solution* (AHS) applies Adaptive Stream Merging with early snooping for clients with bandwidth between but not equal to r and $2r$. For other clients, it operates like SHS. Specifically, it applies Batching for clients with r bandwidth and Patching or ERMT for client with bandwidth of $2r$ or higher. This leads to two alternative schemes: AHS-P (when Patching is used) and AHS-E (when ERMT is used).

Figure 1(e) further explains how AHS-E works. We assume here that the buffering time cannot exceed 30 seconds. The $3rd$ and the $5th$ clients are serviced using a-streams instead of batching streams, which reduces the cost from 28829 seconds in case of SHS-E to 14459 seconds with AHS-E. The buffering times are 3 seconds for the $3rd$ client and 6 seconds for the $5th$. Servicing the $4th$ client using an a-stream would have resulted in 80 seconds of buffering time, which exceeds the assumed threshold, and thus, it is serviced by Batching.

0.3.3 Enhanced Hybrid Solution (EHS)

The Adaptive Hybrid Solution (AHS) requires additional delay due to buffering time, which may not be suitable for True Video-on-Demand (TVOD). Thus, we propose the concept of *enhanced adaptive streams*, or succinctly *ea-streams*, which are a generalization of the adaptive streams in AHS. Like a-streams, they are used for clients with bandwidth between but not equal to r and $2r$. These two stream types, however, vary significantly. The basic idea of ea-streams can be explained as follows. Initially, the server delivers the requested video at a rate higher than the playback rate, using the full client bandwidth. Then, the client starts to listen to the latest regular stream while using the remaining bandwidth to receive a slow patch stream at a rate lower than the playback rate. Therefore, an ea-stream has two phases: fast transmission at a rate higher than playback rate, followed by a slow transmission at a rate lower than the playback rate. These two phases take X and Y seconds, respectively. The initial transmission of the video at a higher rate than the playback rate eliminates the need for any extra delay in addition to the waiting time for server resource to become available and the buffering time for smoothing out the variation in the end-to-end packet delay (delay jitter).

Figure 3(a) further clarifies the concept of ea-streams. In this example, a client with bandwidth b, where $r < b < 2r$ (i.e., $B = \frac{b}{r}$), arrived P seconds after the latest regular stream (the first stream in the figure). The second stream is the ea-stream. The dashed line depicts the actual playback line. The sign @ denotes the transmission rate in the unit of the playback rate. X and Y must be controlled so as to achieve zero delay caused by buffering (i.e., $T_{buf} = 0$).

The question now arises as to how to provide the client with a continuous playback with minimum cost and without any additional waiting time due to buffering (i.e., $T_{buf} = 0$). Basically, two conditions must be satisfied. First, the last playback point (frame) $P + X$ in the ea-stream must arrive at the scheduled time of playback. Specifically, it is required for playback exactly after $P + X$ seconds of the video. This time must be equal to the required time to receive that point of the video: $X + Y$ seconds. Second, the data that is delivered by the ea-stream must be equal to the missed data from the last regular stream, which is equal to $P + X$. These two conditions translate to the following two equations, respectively:

$$P + X = X + Y \tag{2}$$

and

$$X.B + Y.(B - 1) = X + P. \tag{3}$$

It follows from these two equations that

$$Y = P \tag{4}$$

and

$$X = \frac{(2 - B)}{(B - 1)}P. \tag{5}$$

Hence, the cost for delivering the ea-stream in seconds of data can be determined by

$$Cost = P + X = \frac{P}{B - 1}. \tag{6}$$

Equation 6 is valid only if the cost is less than the video length. Otherwise, a new regular stream must be delivered instead.

A generalized version of ea-streams can allow for some *controlled* buffering waiting time (T_{buf}) to reduce the size of the delivered data, which may be suitable for Near Video-on-Demand (NVOD). Let us assume that a NVOD service is provided, T_{max} is the maximum allowed client waiting time, and the client has already waited T_{org} seconds before being admitted for service. Subsequently, $T_{buf_{max}} = max(T_{max} - T_{org}, 0)$. If an a-stream (as opposed to ea-stream) can result in a lower T_{buf} than $T_{buf_{max}}$, then it should be used, and thus $X = 0$, $Y = \frac{P}{B-1}$, and the cost is simply P. Otherwise, $T_{buf} = T_{buf_{max}}$ and the following two equations must be satisfied, as illustrated in Figure 3(b):

$$X + Y - T_{buf} = X + P \tag{7}$$

and

$$X.B + Y.(B - 1) = X + P. \tag{8}$$

Subsequently, Y and X can be found as follows:

$$Y = P + T_{buf} \tag{9}$$

and

$$X = \frac{(2 - B)}{(B - 1)}.P - T_{buf}. \tag{10}$$

Finally, the cost is reduced by T_{buf} and can be found by

$$Cost = P + X = \frac{P}{B - 1} - T_{buf}. \tag{11}$$

Again, the last equation is valid only if the cost is less than the video length. Otherwise, a new regular stream must be delivered instead.

The use of ea-streams can be combined in a hybrid solution with Batching and Patching or ERMT to support clients with different bandwidth classes. We refer to this solution as *Enhanced Hybrid Solution* (EHS). It has two variants: EHS-P and EHS-E, depending on whether Patching or ERMT is used for clients with $2r$ or higher bandwidth. Figure 1(f) shows the prior example with EHS-E. The cost here is reduced to 7421.34 seconds of video data.

0.3.4 Optimal Control of Regular Streams

Equations 6 and 11 highlight the importance of minimizing the average value of time skewness P since the last full stream. Let us assume that for a certain D-second video, the request arrival rate is λ and the average time between two consecutive full streams is W_{reg}. Thus, the number of requests arriving per video length can be given by $N = \lambda D$ and the number of full streams per video length is $\frac{D}{W_{reg}}$. When $T_{buf} = 0$, the cost per ea-stream is $\frac{P}{B-1}$, and thus the total cost for servicing a heterogeneous group of clients arriving within a video duration can be given by

$$Cost = \sum_{i=1}^{N} (\frac{P_i}{B_i - 1}) + \frac{D}{W_{reg}}.D. \tag{12}$$

For simplicity, let us assume a homogeneous group of clients, each with bandwidth B^* (in multiples of the video playback rate). Since the average value of P_i for a long period of time is $\frac{W_{reg}}{2}$, the total cost per video duration is given by

$$Cost = \sum_{i=1}^{N}(\frac{P_i}{B^* - 1}) + \frac{D^2}{W_{reg}} \approx \frac{\lambda D W_{reg}}{2(B^* - 1)} + \frac{D^2}{W_{reg}}. \quad (13)$$

$W_{reg_{opt}}$ can be found as follows:

$$\frac{\partial}{\partial W_{reg}}(\frac{\lambda D W_{reg}}{2(B^* - 1)} + \frac{D^2}{W_{reg}}) = 0. \quad \Rightarrow \quad W_{reg_{opt}} = \sqrt{\frac{2D(B^* - 1)}{\lambda}}. \quad (14)$$

By substituting $W_{reg_{opt}}$ in Equation 13,

$$Cost_{opt} \approx \frac{D\sqrt{\lambda D}}{\sqrt{2(B^* - 1)}} + \frac{D\sqrt{\lambda D}}{\sqrt{2(B^* - 1)}}. \quad (15)$$

Therefore, when the delivery cost is minimized, the total cost of ea-streams is equal to the total cost of full streams. This conclusion is valid for both a-streams and ea-streams in both homogeneous and heterogeneous client environments and with both TVOD and NVOD service models. Motivated by this conclusion, the server can simply trigger full streams dynamically by computing the total cost of a-streams or ea-streams for each video since the last full stream and initiating a full stream for servicing the new requests for the video if the computed value exceeds the cost of a full stream.

0.3.5 Handling Variations in Client Bandwidth over Time

The variation in client bandwidth during the service time makes streaming challenging, especially when resource sharing is employed. Thus, the server should be conservative in estimating the client bandwidth to avoid some pauses in the playback. Compared with SHS and AHS, EHS can deal much better with this problem because of the fast delivery period of ea-streams. These streams can adapt easily to drops in the average bandwidth below the estimated value during the fast delivery period, as long as the average remains higher than b. Before the end of the fast delivery period, X and Y can be recomputed using the client bandwidth history, and thus the fast delivery period can be extended if necessary. This also helps in having a more accurate estimate for the more sensitive, slow delivery period. X can be computed conservatively to deal with further unexpected drops in the average bandwidth during the slow delivery period. To cover for a maximum drop of δ_B, where $0 \leq \delta_B \leq B - 1$, the new fast delivery period $\overline{X}$ can be calculated using

$$\overline{X} = X + \frac{\delta_B}{B - 1}.(P + T_{Buf}). \quad (16)$$

The additional cost is $\frac{\delta_B}{B-1}(P + W_{Buf})$. Consequently, the new value of Y ranges from 0 to $P + T_{Buf}$, depending on both δ_B and the actual drop in bandwidth. If the bandwidth drops beyond δ_B during the slow delivery period, the server can switch the service back to the fast

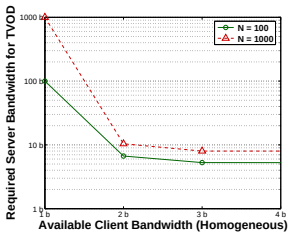
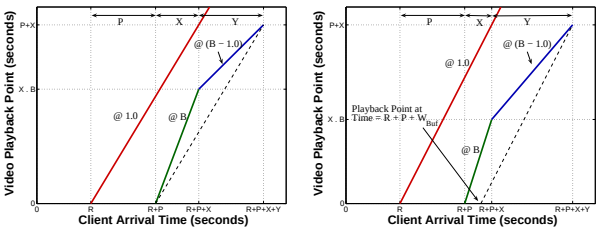


Figure 2: Lower Bound of Server Bandwidth Requirement



(a) Without "Forced Waiting Time" (b) With "Forced Waiting Time"

Figure 3: Enhanced Adaptive Streams

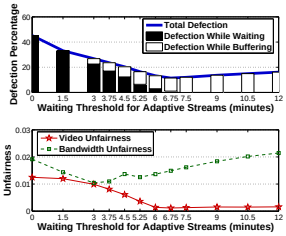
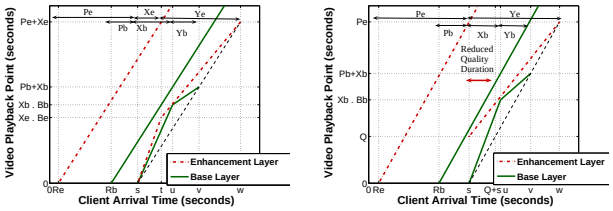


Figure 4: Effect of AHS Waiting Threshold [MQL, $B = 1000$]



(a) TVOD with Full Video Quality (b) TVOD with Short Period of Reduced Quality

Figure 5: Options for Supporting Layered Video Coding

delivery mode (when the server resources become available for the increased delivery rate). If the client bandwidth is significantly unstable, the fast delivery is recommended to continue until the merge with a full stream. A drop in bandwidth below b for batching clients (or any other clients) cannot be solved unless a layered video coding is used and the video quality is reduced for sometime to fit the client available bandwidth.

0.4 Dealing with Heterogeneous Available Client Buffer Space

Available client buffer space is another concern, especially with the wide variety of devices capable of streaming videos. It comes, however, next in importance after the download bandwidth (as long as some buffer space is available in each client). This is due to two reasons. First, the download bandwidth is generally the bottleneck and is decided by more than the device features and specs. Second, while scalable resource sharing techniques (such as ERMT and Patching) require strict download bandwidth at almost every client, they are much more forgiving with buffer space. For example, using Patching, a client with bandwidth *slightly* less than the required $2r$ can only be serviced using regular streams. Therefore, such a client can either defect or wait until the next regular stream (for approximately W_{reg}), or alternatively Patching can change its behavior by initiating a regular stream earlier. With the last option, Patching may effectively lose its advantage over Batching if the clients with such bandwidth are common. The situation is different when it comes to buffer space. In particular, a client with buffer space of $r.S$ in bits that is slightly less than the required $b.W_{reg}$ buffer space can be serviced as long as it arrives within S seconds from the last regular

stream. If it does not arrive within that window, it needs to wait for the next regular stream for approximately $W_{reg} - S$ seconds, or the next regular stream has to start approximately $W_{reg} - S$ seconds earlier. In general, the second option is preferred over forcing clients to wait for extended periods, given that the server resources permit such an earlier service. For all stream merging techniques, including those proposed in this paper, the required client buffer space is $(P + T_{buf}).r$, where $T_{buf} = 0$ for all stream types other than a-streams and ea-streams with forced waiting, and it is also optional in the latter and can be reduced as much as possible to fit the client capability.

0.5 Request Scheduling for Heterogeneous Download Bandwidth and Buffer Space

Supporting client heterogeneity complicates the scheduling issue. In this case, a video waiting queue may contain requests for clients varying in download bandwidth and buffer space. So, there are different subgroups of requests that can be serviced concurrently. For example, assume that three requests R1, R2, and R3 for video v are waiting for service. The corresponding clients have 1, 1.5, and 2 channels, respectively. Assuming enough buffer space in all three, the server can service all three requests by Batching or only R2 and R3 by the a-streams or ea-streams, or only R3 by Patching or ERMT. Thus, we introduce the concept of a *virtual queue*. A virtual queue is a subgroup of the waiting requests for a certain video. In particular, virtual queue $q_{v,i,j}$ has all clients in the v^{th} video queue (q_v) with bandwidth classes $\geq i$ and buffer space classes $\geq j$. A bandwidth class is a group of clients with bandwidth capacities within a specified range. Similarly, a buffer space class is a group of clients with buffer space within a specified range. For example, bandwidth *Class 0* can be for clients with bandwidth within $[r, 1.05r[$, *Class 1* for clients with bandwidth within $[1.05r, 1.10r[$, and so on. The same applies for buffer space classes. *Class 0* can be for clients with buffer space within $[0, 30r[$, *Class 1* for clients with buffer space within $[30r, 60r[$, and so on. In this example, the bandwidth step is $0.05r$ and called *bandwidth class width*, and the *buffer space class width* is $30r$.

Scheduling can be performed by dividing the requests in each video queue to virtual queues. Instead of selecting a video queue as in the homogeneous case, a scheduling policy should select a virtual queue in the set of all possible virtual queues (among all videos). Consequently, the implementations of the existing scheduling policies may need to be adjusted so as to incorporate the new virtual queue concept. Because of their nature, MQL and FCFS will select for service *all* requests in a particular video queue and will not utilize the variations in client bandwidth and buffer space. MFQL and MCF, however, can exploit these variations by operating at the virtual queue level, and thus not all waiting requests for a video are necessarily selected. Our implementation of MFQL for the heterogeneous environment divides the virtual queue length by the square root of the relative access frequency of clients with similar or lower bandwidth and buffer space classes requesting the same video. Hence, it tries to reduce the bias against both lower classes and unpopular videos, in the same way the homogeneous MFQL tries to do for unpopular videos.

0.6 Towards a More Comprehensive Delivery Solution

The proposed resource sharing solutions do not assume any particular video coding. To better support client heterogeneity, however, they can be used with layered video coding. This coding encodes the video into a base layer and one or more enhancement layers. The base layer provides low, but acceptable and usable quality, whereas each enhancement layer provides further incremental refinement to the quality. The combination of the proposed resource sharing solutions and layered coding can provide a comprehensive and efficient solution.

The proposed resource sharing solutions can operate simply on each layer independently. This approach allows supporting clients with a wider range of bandwidth, including those with fewer than one channel. In addition, using layered coding can provide other more aggressive options in servicing clients with bandwidth capacities higher than the playback rate, such as (i) using adaptive or enhanced adaptive streams to increase data share-ability, (ii) reducing video quality for a very short period of time, and (iii) some combination of the two. Figure 5 clarifies the first two options in the case of a TVOD service. The investigation of this comprehensive solution is left for future work.

0.7 Performance Evaluation Methodology and Results

We have developed a simulator for a video streaming server that supports various resource sharing techniques and scheduling policies. The simulator has been validated by reproducing several graphs in previous studies. The simulation stops after a steady state analysis with 95% confidence interval is reached.

We analyze the effectiveness of the proposed schemes and analyze various scheduling policies through extensive simulation. We consider two service models: TVOD and NVOD. In NVOD, we fix the server bandwidth and consider primarily five performance metrics: customer defection probability, average waiting time, unfairness against unpopular videos, unfairness against clients with low bandwidth, and unfairness against clients with low buffer space. The defection probability is the most important and can be defined as the probability that customers leave the system without being serviced because of waiting times exceeding their tolerance. The average waiting time comes next in importance. It includes the client waiting time for admission and any possible buffering time. Video unfairness quantifies the bias against unpopular videos and is equal to $\sqrt{\sum_{i=1}^{M}(d-d_i)^2/(M-1)}$, where M is the number of videos, d_i is the i_{th} video defection rate, and d is the mean defection rate for all videos. We introduce bandwidth/buffer space class unfairness to measure the bias against clients with low bandwidth/buffer space classes. It can be calculated using the previous equation by redefining M as the number of classes, d_i as the i_{th} class defection rate, and d as the mean defection rate for all classes. In TVOD, we compare the server bandwidth required to service all requests immediately.

0.7.1 Workload Characteristics

Like most prior studies, we assume that the arrival of the requests follows a Poisson Process with an average arrival rate λ and the accesses to videos are highly localized and follow a Zipf-like distribution with skewness parameter $\theta_v = 0.271$. Generally, we characterize the waiting tolerance of customers by an exponential distribution with mean μ_{tol}. However, with AHS, we utilize the *time-of-service guarantee* (TSG) model [2, 22, 26] whenever the actual playback time is known and guaranteed. With this model, if the current waiting time plus the additional buffering time is less than μ_{tol}, the customer waits; otherwise, the waiting tolerance follows an exponential distribution with mean μ_{tol}. The default value of μ_{tol} is 1.5 minutes. EHS is implemented without requiring any additional waiting time due to buffering (i.e., $T_{buf} = 0$) and thus TSG is not used. Unless otherwise indicated, each client is assumed to have 60 minutes of buffer space to ensure high performance.

Table 1 summarizes the main workload characteristics and shows the default values. Unless otherwise indicated, we consider a server with 120 videos, each of which is 120-minute long. We examine the server at different loads by fixing the request arrival rate at 40 requests per minute and varying the server bandwidth (server capacity) generally from $300r$ to $3000r$. Since interactive operations can be supported using contingency channels [8], the relative performance of various techniques and policies does not depend on these operations as long as the fraction of server channels used for these operations is kept the same (which is typically the case). To isolate the impact of these operations, the reported server capacity in this study includes only non-contingency channels.

Because of the lack of a workload characterization study of client bandwidth, we generally use five bandwidth distributions: two Zipf-like distributions with skewness parameters θ_B = 0.0 and 0.5, and three truncated Normal distributions with mean values μ_B = 1.5, 2.0, and 2.5 and a standard deviation of 0.5, all in the unit of one channel at the playback rate r. The first Zipf distribution represents the case when there is a high ratio of clients with bandwidth in the region just above r, and the others cover other possible scenarios. The client bandwidth capacities range between r and $11r$ in all five distributions. Similarly, we characterize the buffer space distribution by five distributions: Uniform, Zipf (with parameter $\theta_s = 0$), and three Normal distributions with μ_S = 10, 30, and 60 minutes of video. The standard deviation of the Normal distributions is 15 minutes.

0.7.2 Result Presentation and Analysis

Comparing Resource Sharing Schemes

Let us start by comparing various resource sharing schemes that work for heterogeneous receivers (Batching, SHS-P, SHS-E, AHS-P, AHS-E, EHS-P, and EHS-E) in terms of defection rate, average waiting time, and video and bandwidth class unfairness. Figures 6 and 7 depict the results under Normal and Zipf-Like bandwidth distributions, respectively. At this stage, scheduling is performed using MQL, and no forced waiting time is done with EHS (i.e., $T_{buf} = 0$).

The results demonstrate the advantage of utilizing ea-streams in EHS and a-streams in AHS, with a noticeable lead for the former, regardless of the workload and bandwidth distribution. For example, under the Normal distribution, while Batching requires $4800r$ in

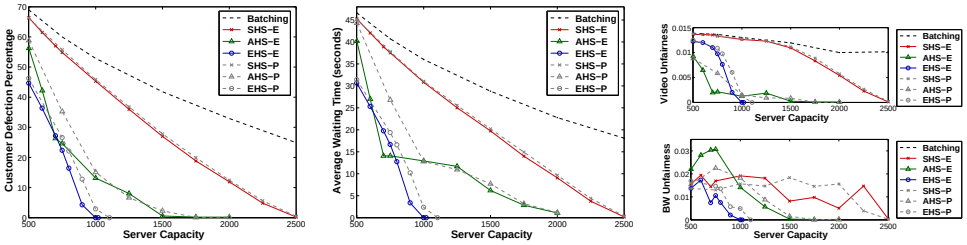
Table 1: Summary of Workload Characteristics

Parameter	**Model/Value(s)**
Request Arrival Model	Poisson Process
Request Arrival Rate (λ)	20 to 360, default: **40** Requests/min.
Server Capacity	$300r$ to $3000r$
Video Access	Zipf-Like ($\theta_v = 0.271$)
Number of Videos	60 to 1920, default: **120**
Video Length (D)	5 to 180, default: **120** min.
Waiting Tolerance Model	Exponential with mean $\mu_{tol} = 0.5$, **1.5** (default), and 2.5 min. Time-of-Service Guarantee with threshold μ_{tol}
Waiting Tolerance Mean (μ_{tol})	0.5, **1.5** (default), 2.5 min
Client Bandwidth Model	Zipf-Like with $\theta_B = 0$ and 0.5 Normal with mean $\mu_B = 1.5r$, $2.0r$, and $2.5r$, $\sigma_B = 0.5r$
Client Bandwidth Range	r to $11r$
Bandwidth class width	$0.05r$
Client Buffer Space Model	Uniform Zipf (with $\theta_s = 0$) Normal with mean $\mu_S = 10$, 30, and 60 min, $\sigma_S = 15$ min
Buffer Space Range	5 to 120 min
Buffer space class width	10 sec

server bandwidth to achieve TVOD (i.e., 0 defections and 0 waiting time), SHS-E requires $2500r$, and EHS-E requires less than $1000r$. AHS-E can achieve 0 defections with $1700r$ server bandwidth, but it can never completely eliminate the waiting times due to the required, initial buffering time. Moreover, with $1000r$ server bandwidth, when EHS-E achieves TVOD, AHS-E, SHS-E, and Batching cause defections of approximately 13%, 45%, and 53%, respectively. Of course, when it comes to the less important metrics, video and bandwidth class unfairness in particular, the results are slightly different. Batching treats all clients as homogeneous receivers, but among the other three solutions, EHS is the fairest towards lower bandwidth classes. For video unfairness, EHS and AHS are the fairest among all schemes (including Batching) towards unpopular videos. The better fairness of AHS in some cases towards unpopular videos is because the defections here are of two types: defections caused by waiting times till admission and defections caused by waiting times after admission due to buffering. The unfair nature of MQL makes the first type dominated by unpopular video clients and the second dominated by popular video clients with low bandwidth. This in a way balances the gap in defection rates among videos.

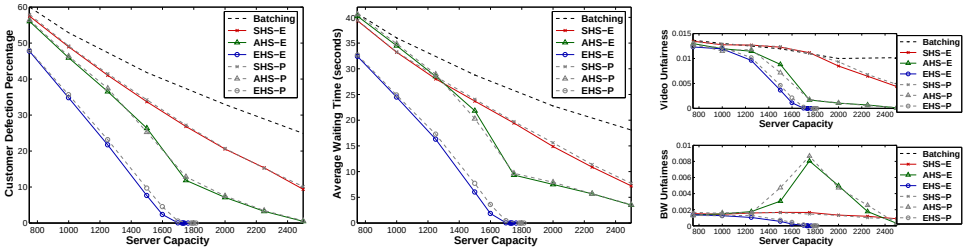
There are two interesting observations. (1) The three proposed solutions vary significantly in performance, whereas the two variants of each perform close to one other. In other words, using ERMT instead of Patching for serving clients with bandwidth capacities of $2r$ or higher is of less significance than changing the solution (and thus the method of service for clients with bandwidth between r and $2r$, noninclusive). (2) Although some additional buffering time is imposed by a-streams in AHS, a significant gain is achieved even in the average waiting time, compared to SHS and Batching.

From this point on, only the ERMT variants of the three solutions are considered: EHS-



(a) Defection (b) Waiting (c) Unfairness

Figure 6: Comparing Different Schemes [Normal Bandwidth Dist. with $\mu_B = 2.0$, MQL]



(a) Defection (b) Waiting (c) Unfairness

Figure 7: Comparing Different Schemes [Zipf Bandwidth Dist. with $\theta_B = 0.0$, MQL]

E, AHS-E, and SHS-E. EHS-P, AHS-P, and SHS-P are dropped because they perform close to, but slightly worse than EHS-E, AHS-E, and SHS-E, respectively. Batching is dropped because of its poor performance.

Comparing Scheduling Policies

Let us now discuss the impact of scheduling. Figures 8, 9, and 10 compare scheduling policies when applying EHS-E, AHS-E, and SHS-E, respectively. The results are shown under two bandwidth distributions: Zipf with $\theta_B = 0.0$ and Normal with $\mu_B = 2.0$. FCFS and MFQL in general perform worse than MCF-P and MQL in terms of defection rate and waiting time. Figure 11 compares scheduling policies in the two main metrics (defection rate and waiting time) under the remaining bandwidth distributions. To keep the figure less crowded, FCFS and MFQL are excluded.

With EHS and AHS, MCF-P achieves the best results in the two most important metrics, compared with all considered policies, regardless of the bandwidth distribution. It also has better video fairness than its closest competitor (MQL). MCF-P, however, does not perform well in bandwidth class unfairness since it considers the cost. Overall, MCF-P is generally the best choice with EHS and AHS. The results are quite different with SHS-E. The best two candidates are still MQL and MCF-P, based on the two most important metrics. While MQL causes fewer defections than MCF-P when the majority of clients have low bandwidth (as in Zipf with $\theta_B = 0$ and Normal with $\mu_B = 1.5$), MCF-P performs better when clients have higher bandwidth capacities on the average. The high defection rate of MCF-P compared with MQL in the first case is due to its bias against Batching clients, who require the highest cost of service. When SHS is used, Batching clients include every client with bandwidth lower than $2r$. MCF-P, however, performs better than MQL in the average waiting time.

Next, we use the best scheduling policy for each resource sharing scheme. For SHS-E, MQL is used in case of Normal($\mu_B = 1.5$), and MCF-P is used under the other distributions. For EHS-E and AHS-E, MCF-P is always used. MCF-P is chosen for SHS-E under Zipf($\theta_B = 0.0$) because the small increase in defection rate compared with MQL is justified by a more significant reduction in waiting time.

Comparing Schemes with Best Scheduling

Figure 12 compares EHS-E, AHS-E, and SHS-E under five different bandwidth distributions. As expected, the performance gap increases when the distribution offers more clients that can benefit from ea-streams or a-streams. For example, when the average client bandwidth is $2.5r$, the majority of clients do have more than the $2r$ required by ERMT, while when the average client bandwidth is $1.5r$, the majority can be serviced by ea-streams or a-streams. However, EHS-E keeps its lead (followed by AHS-E) under all bandwidth distributions. In practical systems, the average client bandwidth is not expected to be very large compared to the video playback rate (r) because having a large portion of clients with very high bandwidth usually motivates a higher video quality.

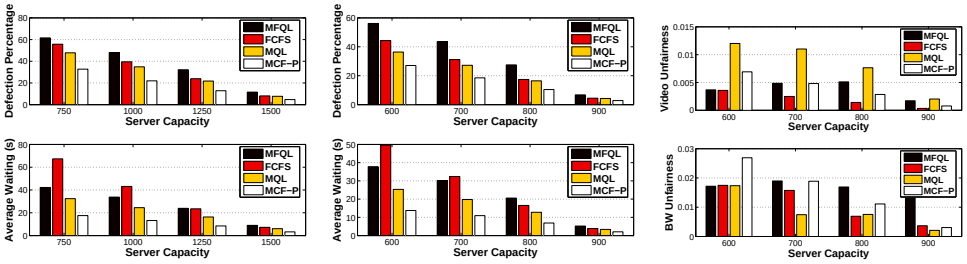
Impact of Forced Waiting with EHS-E

As discussed in Section 0.3, EHS-E can be implemented with some additional buffering time (T_{buf}). Figure 13 shows the results for two values of forced waiting: 1.5 and 3.0 minutes. Although a small buffering time might slightly increase server throughput, it has a much larger negative impact on the waiting time. Large buffering times, however, negatively impact both throughput and waiting time. Thus, it is preferred not to introduce any forced waiting time with EHS. Interestingly, the average waiting time is smaller for the larger value of the buffering time. This is because only serviced customers are considered in the average waiting time. Forced buffering time in the case of ea-streams affects every client almost equally (unlike a-streams). Therefore, a relatively large buffering time, compared to the average waiting tolerance, can result in the defection of the majority of clients serviced by ea-streams.

Impact of Available Client Buffer Space

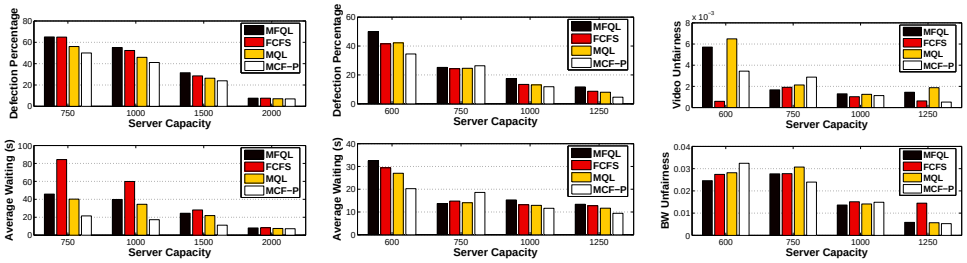
Let us now study the impact of available client buffer space on the relative performance of the proposed resource sharing schemes. As demonstrated in Figure 14, limited buffer space and its heterogeneity do not change the relative performance of various schemes. It narrows, however, the performance gaps among them, especially between the two alternative implementations of each solution (such as EHS-E and EHS-P).

Let us now examine in more detail the impact of the available client buffer space on the performance of EHS-E under different download bandwidth distributions. As expected, Figure 15 demonstrates that system performance increases with the available buffer space, but the change is not dramatic even with a significant change in the available buffer space. For example, increasing the average available buffer space by a factor of 6 (under Normal Distribution and Server Capacity of 500) reduces the defection probability by only 16% from 43% to 36%, and the average waiting time by only 18% from 22 to 18 seconds. As expected,



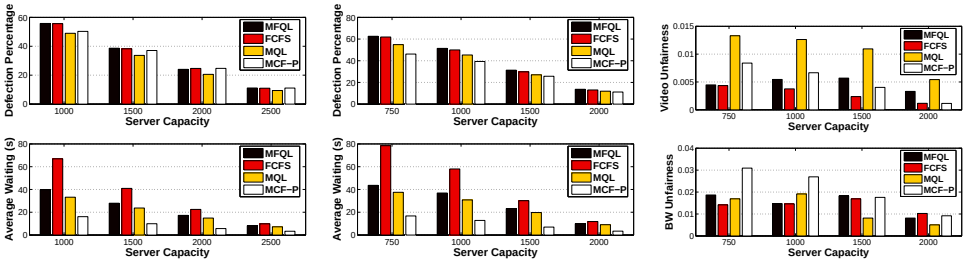
(a) Zipf ($\theta_B = 0.0$) (b) Normal ($\mu_B = 2.0$) (c) Normal ($\mu_B = 2.0$)

Figure 8: Comparing Scheduling Policies for EHS-E



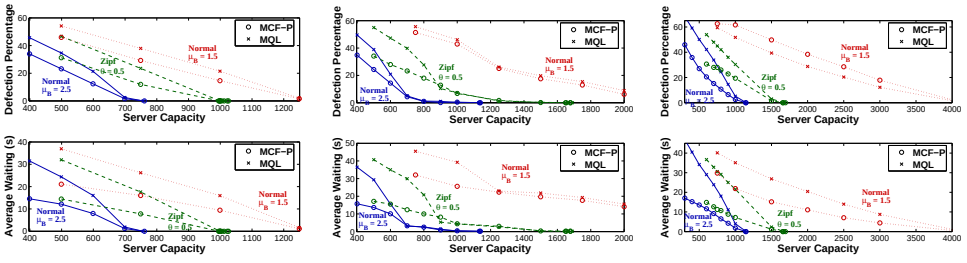
(a) Zipf ($\theta_B = 0.0$) (b) Normal ($\mu_B = 2.0$) (c) Normal ($\mu_B = 2.0$)

Figure 9: Comparing Scheduling Policies for AHS-E



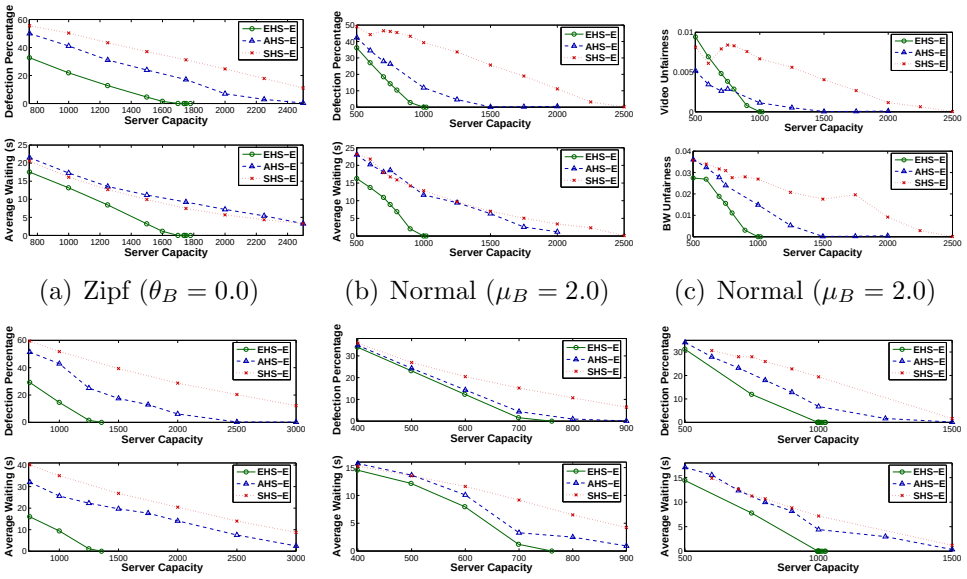
(a) Zipf ($\theta_B = 0.0$) (b) Normal ($\mu_B = 2.0$) (c) Normal ($\mu_B = 2.0$)

Figure 10: Comparing Scheduling Policies for SHS-E



(a) EHS-E (b) AHS-E (c) SHS-E

Figure 11: Comparing Scheduling Policies with Different Bandwidth Distributions



(a) Zipf ($\theta_B = 0.0$) (b) Normal ($\mu_B = 2.0$) (c) Normal ($\mu_B = 2.0$)

(d) Normal ($\mu_B = 1.5$) (e) Normal ($\mu_B = 2.5$) (f) Zipf ($\theta_B = 0.5$)

Figure 12: Comparing the Schemes with Best Scheduling

video unfairness is not very sensitive to buffer space availability. The impact on bandwidth class unfairness is moderate and is inversely proportional to the buffer space. This is due to forcing more full streams which indirectly helps the clients with lower bandwidths.

Comparing the Impacts of Server Capacity, Client Bandwidth, and Client Buffer Space

After evaluating in isolation the impacts of different client and server resources on system performance and requirements, let us now compare the impacts of the three main resources: server capacity, client bandwidth, and client buffer space. Comparing the impacts of these resources is very helpful in taking informative decisions to face any degradation in system performance. Figure 16 plots the results under different combinations of buffer space availability levels for these three main resources. The main results can be summarized as follows:

(1) The gain is maximized when we invest in the resource that is the bottleneck. For example, as indicated in Table 2, doubling the average client buffer space from 10 minutes worth of data to 20 reduces the server capacity required for providing TVOD service approximately by 3% when the average download bandwidth is $1.5r$, by 6% when the average download bandwidth is $2.0r$, and by 9% when the average download bandwidth is $2.5r$. Thus, as the download bandwidth increases, the buffer space becomes more significant. The reverse also holds true. For example, as the average download bandwidth increases by 33%, the required server capacity to achieve TVOD service is reduced approximately by 16% when the average buffer space is 10 minutes, by 21% when the average buffer space is 30 minutes, and by 24% when the average buffer space is 60 minutes. In NVOD, increasing the average client buffer space from 10 minutes of data to 20 (while fixing the download bandwidth at $2.0r$) eliminates only 5% of the defections when the server capacity is 500 channels, 12% of the defections when the server capacity is at 750, and 47% of the defections when the

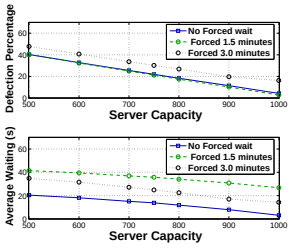
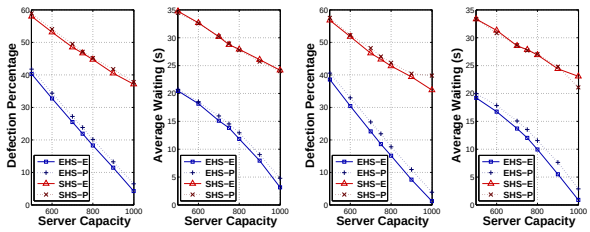
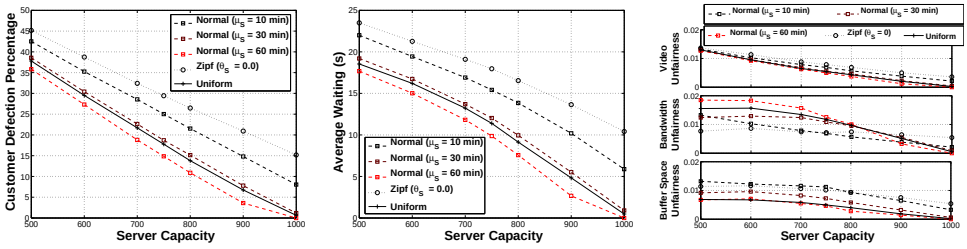


Figure 13: Impact of Forced Waiting with EHS-E [Normal Bandwidth Dist. with $\mu_B = 2.0$, Normal Buffer Space Dist. with $\mu_S = 20$ min, MCF-P]



(a) $\mu_S = 20$ min (b) $\mu_S = 30$ min

Figure 14: Impact of Buffer Space on Various Schemes [Normal Bandwidth Dist. with $\mu_B = 2.0$, Normal Buffer Space Dist., MCF-P]



(a) Defection (b) Waiting (c) Unfairness

Figure 15: Impact of Buffer Space Distribution on EHS-E [Normal Bandwidth Dist. with $\mu_B = 2.0$, MCF-P]

server capacity is 1000 channels. Again, as the average download bandwidth increases from $1.5r$ to $2.0r$ (while fixing the buffer space the size at 10 minutes), 21% of the defections are eliminated when the server capacity is 500 channels, 48% of the defections are eliminated when the server capacity is 750 channels, and all defections are eliminated when the server capacity is 1000 channels.

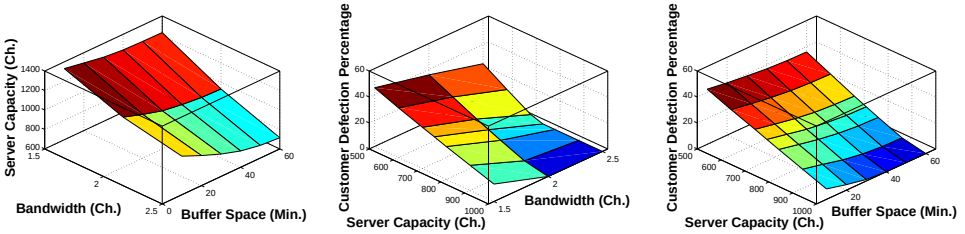
(2) Server and client bandwidth capacities, in general, have larger impacts on performance than the available client buffer space, as long as some buffer space is available in most clients. For example, when the average download bandwidth is $1.5r$, the availability of no buffer space leads to requiring 4800 server channels to achieve TVOD, while a 10-minute buffer space reduces the requirement to approximately 1331. Doubling that average buffer space, however, reduces this requirement to 1295 (i.e., by only 3%). Even hex-tripling that average buffer space reduces the requirement to 1230 (only 7.5%). In contrast, increasing the average download bandwidth from 1.5 to 2.0 (approximately 33%) can reduce the required server capacity to 1116 (approximately 11%). Similarly in NVOD, when the server capacity is 500 channels, increasing the buffer space from 10 minutes to 60 can reduce the defection rate approximately from 42.5% to 36%, while increasing the server capacity to 600 channels results in a defection rate of 35%.

(3) Server and client bandwidth capacities both have comparable effects on system performance. For example increasing the average download bandwidth from 1.5 to 2.5 (67%) reduces the defection approximately from 45% to 25% when the server capacity is 500 chan-

Table 2: Comparative Results of the Impacts of Server Capacity, Client Bandwidth, and Client Buffer Space

Input Parameters				Output		
Client Bandwidth Mean (Ch.)	Buffer Space Mean (Min.)			Server Capacity (Ch.)		
	From	To	%	From	To	%
1.5	10	20	100	1330.9	1294.7	2.72
2.0	10	20	100	1115.9	1049.5	5.95
2.5	10	20	100	996.6	908.7	8.82
Buffer Space Mean (Min.)	Client Bandwidth Mean (Ch.)			Server Capacity (Ch.)		
	From	To	%	From	To	%
10	1.5	2.0	33.3	1330.9	1115.9	16.15
30	1.5	2.0	33.3	1259.4	992.2	21.22
60	1.5	2.0	33.3	1230	933.3	24.12
Server Capacity (Ch.) [Client Bandwidth Mean = 2.0 Ch.]	Buffer Space Mean (Min.)			Defection Probability (%)		
	From	To	%	From	To	%
500	10	20	100	42.5	40.3	5.3
750	10	20	100	25.0	21.9	12.4
1000	10	20	100	8.0	4.3	46.9
Server Capacity (Ch.) [Buffer Space Mean = 10 Min.]	Client Bandwidth Mean (Ch.)			Defection Probability (%)		
	From	To	%	From	To	%
500	1.5	2.0	33.3	45.2	35.8	20.8
750	1.5	2.0	33.3	28.6	14.9	48.1
1000	1.5	2.0	33.3	13.4	0.0	100.0

nels. To achieve comparable results without requiring that much client bandwidth, the server capacity needs to be increased to more than 800 channels (more than a 60% increase). The defection rate at 800 channels is 25%.



(a) Impact of Buffer Space and Bandwidth on Server Capacity

(b) Impact of Bandwidth and Server Capacity on Defection

(c) Impact of Buffer Space and Server Capacity on Defection

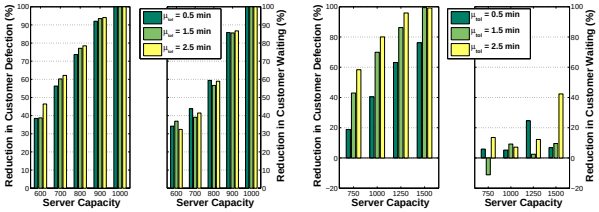
Figure 16: Impact of Client Bandwidth, Client Buffer Space, and Server Capacity on EHS-E Performance [Normal Bandwidth Dist. with Default $\mu_B = 2.0$, Normal Buffer Space Dist. with Default $\mu_S = 60$ min, MCF-P]

Impact of Customer Waiting Tolerance

Let us now discuss the impact of customer waiting tolerance. Figure 17 plots the percentage improvements achieved by EHS-E and AHS-E compared with SHS-E in terms of the defection percentage and the waiting time for different values of μ_{tol}. Four main conclusions can be drawn here. (1) The higher the customer waiting tolerance, the more efficient AHS becomes compared with SHS because longer waiting tolerance allows more clients to be serviced by a-streams even when longer buffering times are required. EHS, on the other hand, is less sensitive to the customer waiting tolerance. 2) Even with very low customer tolerance, such as 30 seconds, AHS achieves a significant improvement over SHS and can eliminate the majority of the defections compared to its. 3) Although the main advantage of AHS over SHS is reducing the defection rate and servicing a larger number of customers, it also results in a shorter average waiting time. 4) Regardless of the average waiting tolerance of customers, EHS keeps its lead over the other solutions, especially in terms of the defection rate and waiting time.

Impacts of Request Arrival Rate, Video Length, and Number of Videos

We discuss here the effectiveness of EHS-E and SHS-E in providing TVOD. (Recall that AHS cannot provide such service because of the required waiting time for buffering.) Figures 18, 19, and 20 compare the required server bandwidth to achieve TVOD by the two schemes versus request arrival rate, video length, and number of videos, respectively, under two bandwidth distributions: Normal with $\mu_B = 2$ and Zipf with $\theta = 0$. The impact of the video length is similar to the that of the request rate. The number of concurrent customers per video increases with each, and thus data sharing will be enhanced. This behavior explains why EHS becomes increasingly more efficient than SHS. EHS can achieve TVOD with a fraction of the requirement of SHS and it scales well with the increase in concurrent customers.



(a) EHS-E vs. SHS-E (b) AHS-E vs. SHS-E

Figure 17: Impact of Customer Waiting Tolerance (μ_{tol})

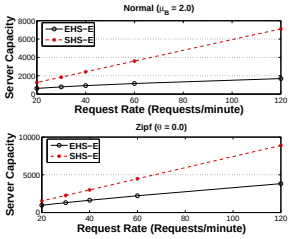


Figure 18: Impact of Variable Request Rate (TVOD)

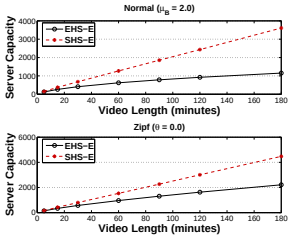


Figure 19: Impact of Variable Video Length (TVOD)

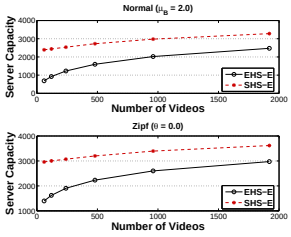


Figure 20: Impact of Variable Video Number (TVOD)

For example, as the request rate increases 6 folds (from 20 to 120 requests/minute), the required server bandwidth to provide TVOD service by EHS-E increases only 2.5 folds under the Normal bandwidth distribution and only 4 folds under the Zipf distribution, whereas the server bandwidth with SHS increases 5.6 to 5.8 folds, with the two bandwidth distributions, respectively. The impact of the number of videos is different. Increasing the number of videos while fixing the other two parameters reduces the number of concurrent customers per video, thereby reducing the chances for data sharing and narrowing the performance gap between EHS and SHS. In other words, as the number of videos increases while fixing the total request arrival rate, the videos become increasingly less popular and thus data sharing decreases.

0.8 Conclusion

We have studied scalable on-demand delivery of video streams to heterogeneous receivers. We have proposed three solutions to address heterogeneity in the download bandwidth: *Simple Hybrid Solution* (SHS), *Adaptive Hybrid Solution* (AHS), and *Enhanced Hybrid Solution* (EHS). SHS simply combines existing resource sharing techniques and deals with clients as two bandwidth classes. AHS and EHS, however, classify clients into multiple bandwidth classes and service them accordingly. Depending on whether Patching or ERMT is used for clients with bandwidth capacities of double the video playback rate or higher, the three solutions lead to six schemes: SHS-P, SHS-E, AHS-P, AHS-E, EHS-P, and EHS-E. Moreover, we have studied the support for heterogeneity in the client buffer space. Furthermore, we have discussed how scheduling policies can be adapted to capture the variations in client bandwidth.

We have evaluated the effectiveness of the proposed schemes and have analyzed various

scheduling policies through extensive simulation. We have considered two service models: *True Video-on-Demand* (TVOD) and *Near Video-on-Demand* (NVOD). The main results can be summarized as follows.

- The proposed schemes can achieve high degrees of resource sharing.
- AHS significantly outperforms SHS and Batching in the considered performance metrics. For example, it can provide a service with no customer defections, while SHS and Batching cause more than 27% and 41% defections, respectively. Additionally, it can reduce the average request waiting time and tends to be more fair towards both unpopular videos and low bandwidth classes.
- EHS achieves significant improvements over AHS under all workloads and service models. In addition, it can provide a TVOD service (i.e., zero waiting time), whereas AHS cannot. EHS is also more tolerant to variations in client bandwidth during service. With the same server bandwidth, EHS-E can achieve zero defections, while AHS-E, SHS-E, and Batching cause 13%, 45%, and 53% defections, respectively.
- The three proposed solutions vary significantly in performance, whereas the two variants of each perform close to one other. In other words, using ERMT instead of Patching for serving clients with bandwidth capacities of $2r$ or higher is of less significance than changing the solution (and thus the method of service for clients with bandwidth capacities higher than b but lower than $2r$).
- Client available buffer space, in general, has less impact on system performance than client bandwidth, but its impact can be very significant if it becomes the main bottleneck.
- The performance depends greatly on the applied scheduling policy. For example, with EHS-E, choosing the fair FCFS instead of the efficient, cost-oriented MCF-P can increase the number of defected customers by more than 50%. In certain situations, MQL performs better than MCF-P but only when SHS is used.

We conclude that EHS-E and EHS-P are the best overall performers. EHS-E performs slightly better but is much more complicated due to the high implementation complexity of ERMT. When any one of these two schemes is employed, it is best to schedule the waiting requests using MCF-P.

Bibliography

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. The maximum factor queue length batching scheme for Video-on-Demand systems. *IEEE Trans. on Computers*, 50(2):97–110, Feb. 2001.

[2] M. Alsmirat, M. Al-Hadrusi, and N. J. Sarhan. Analysis of waiting-time predictability in scalable media streaming. In *Proc. of ACM Multimedia*, pages 791 – 794, Sept. 2007.

[3] O. Bagouet, K. A. Hua, and D. Oger. A periodic broadcast protocol for heterogeneous receivers. In *Proc. of Multimedia Computing and Networking Conf. (MMCN)*, Jan. 2003.

[4] BroadBandReports. `http://broadbandreports.com/`.

[5] Y. Cai and K. A. Hua. An efficient bandwidth-sharing technique for true video on demand systems. In *Proc. of ACM Multimedia*, pages 211–214, Oct. 1999.

[6] Y. Cai and K. A. Hua. Sharing multicast videos using patching streams. *Multimedia Tools and Applications journal*, 21(2):125–146, Nov. 2003.

[7] Y. Cai, W. Tavanapong, and K. A. Hua. Enhancing patching performance through double patching. In *Proc. of 9th Int'l Conf. on Distributed Multimedia Systems*, pages 72–77, Sept. 2003.

[8] A. Dan, P. Shahabuddin, D. Sitaram, and D. Towsley. Channel allocation under batching and vcr control in movie-on-demand servers. *Journal of Parallel and Distributed Computing*, 30(2):168–179, Nov. 1995.

[9] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling policies for an on-demand video server with batching. In *Proc. of ACM Multimedia*, pages 391–398, Oct. 1994.

[10] D. L. Eager, M. K. Vernon, and J. Zahorjan. Optimal and efficient merging schedules for Video-on-Demand servers. In *Proc. of ACM Multimedia*, pages 199–202, Oct. 1999.

[11] D. L. Eager, M. K. Vernon, and J. Zahorjan. Bandwidth skimming: A technique for cost-effective Video-on-Demand. In *Proc. of Multimedia Computing and Networking Conf. (MMCN)*, pages 206–215, Jan. 2000.

[12] D. L. Eager, M. K. Vernon, and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. *IEEE Trans. on Knowledge and Data Engineering*, 13(5):742–757, Sept. 2001.

[13] L. Gao, J. Kurose, and D. Towsley. Efficient schemes for broadcasting popular videos. In *Proc. of the Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, July 1998.

[14] K. A. Hua, Y. Cai, and S. Sheu. Patching: A multicast technique for true Video-on-Demand services. In *Proc. of ACM Multimedia*, pages 191–200, 1998.

[15] K. A. Hua and S. Sheu. Skyscraper broadcasting: A new broadcasting scheme for metropolitan Video-on-Demand system. In *Proc. of ACM SIGCOMM*, pages 89–100, Sept. 1997.

[16] C. Huang, R. Janakiraman, and L. Xu. Loss-resilient on-demand media streaming using priority encoding. In *Proc. of ACM Multimedia*, pages 152–159, Oct. 2004.

[17] L. Juhn and L. Tseng. Harmonic broadcasting for Video-on-Demand service. *IEEE Trans. on Broadcasting*, 43(3):268–271, Sept. 1997.

[18] J.-F. Pâris. A fixed-delay broadcasting protocol for Video-on-Demand. In *Proc. of the Int'l Conf. on Computer Communications and Networks*, pages 418–423, 2001.

[19] B. Qudah and N. J. Sarhan. Analysis of resource sharing and cache management techniques in scalable video-on-demand. In *Proc. of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 327–334, Sept. 2006.

[20] M. Rocha, M. Maia, I. Cunha, J. Almeida, and S. Campos. Scalable media streaming to interactive users. In *Proc. of ACM Multimedia*, pages 966–975, Nov. 2005.

[21] N. J. Sarhan and C. R. Das. An integrated resource sharing policy for multimedia storage servers based on network-attached disks. In *Proc. of the Int'l Conf. on Distributed Computing Systems (ICDCS)*, pages 136–143, May 2003.

[22] N. J. Sarhan and C. R. Das. A new class of scheduling policies for providing time of service guarantees in Video-On-Demand servers. In *Proc. of the 7th IFIP/IEEE Int'l Conf. on Management of Multimedia Networks and Services*, pages 127–139, Oct. 2004.

[23] N. J. Sarhan and B. Qudah. Efficient cost-based scheduling for scalable media streaming. In *Proc. of Multimedia Computing and Networking Conf. (MMCN)*, page 65040C, Jan./Feb. 2007.

[24] P. Sessini, L. Shi, A. Mahanti, Z. Li, and D. L. Eager. Scalable streaming for heterogeneous clients. In *Proc. of ACM Multimedia*, pages 337–346, Oct. 2006.

[25] M. A. Tantaoui, K. A. Hua, and T. T. Do. Broadcatch: A periodic broadcast technique for heterogeneous Video-on-Demand. *IEEE Trans. on Broadcasting*, 50(3):289–301, Sept. 2004.

[26] A. K. Tsiolis and M. K. Vernon. Group-guaranteed channel capacity in multimedia storage servers. In *Proc. of ACM SIGMETRICS*, pages 285–297, June 1997.