Efficient Delivery of On-Demand Video Streams to Heterogeneous Receivers

BASHAR QUDAH, NABIL J. SARHAN, Wayne State University

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, such as Batching, Patching, and Earliest Reachable Merge Target (ERMT), can be used to address this problem by utilizing the multicast facility, which allows multiple requests to share the same set of server and network resources. They assume, however, that all clients have the same available download bandwidth and buffer space. We study how to efficiently support clients with varying available download bandwidth and buffer space, while delivering data in a client-pull fashion using enhanced resource sharing. In particular, we propose three *hybrid solutions* to address the variability in the download bandwidth among clients: *Simple Hybrid Solution* (SHS), *Adaptive Hybrid Solution* (AHS), and *Enhanced Hybrid Solution* (EHS). SHS simply combines Batching with either Patching or ERMT, leading to two alternatives: *SHS-P* and *SHS-E*, respectively. Batching is used for clients with bandwidth lower than double the video playback rate, 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 ranging between the video playback rate and double that rate. 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. Moreover, we consider the variability of the available buffer space among clients. Furthermore, we study how the waiting playback requests for different videos can be scheduled for service in the heterogeneous environment, capturing the variations in both the client bandwidth and buffer space. We evaluate the effectiveness of the proposed solutions and analyze various scheduling policies through extensive simulation.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.4 [**Computer Systems Organization**]: Performance of Systems; I.6.5 [**Simulation and Modeling**]: Model Development

General Terms: Design, Performance

Additional Key Words and Phrases: Adaptive stream merging, client heterogeneity, multimedia servers, Video-on-Demand (VOD), video streaming.

ACM Reference Format:

Qudah, B. and Sarhan, N. J. 2010. Efficient delivery of on-demand video streams to heterogeneous receivers. ACM Trans. Multimedia Comput. Commun. Appl. 6, 3, Article 20 (August 2010), 25 pages.
DOI = 10.1145/1823746.1823754 http://doi.acm.org/10.1145/1823746.1823754

A preliminary version of this article was presented at ACM Multimedia [Qudah and Sarhan 2006b]
This work is supported in part by NSF grant CNS-0626861.
Author's address: Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI; email: {bqudah, nabil}@wayne.edu.

DOI 10.1145/1823746.1823754 http://doi.acm.org/10.1145/1823746.1823754

1. INTRODUCTION

Recently, video streaming services have grown dramatically in popularity. Unicast is the simplest video delivery technique. It services each request using a full-length video stream at the video playback rate. Considering the high delivery cost in terms of the total number of bits or seconds of the delivered video data, the number of video streams that can be delivered concurrently is highly constrained. Resource sharing techniques address this challenge by utilizing the multicast facility. Multicast allows multiple requests to share the same streams and thus the same sets of server and network resources, including network bandwidth and disk I/O bandwidth. These techniques include *Batching* [Aggarwal et al. 2001; Dan et al. 1994; Tsiolis and Vernon 1997], *Stream Merging* [Hua et al. 1998; Cai and Hua 1999; Eager et al. 1999], and *Periodic Broadcasting* [Pâris 2001; Juhn and Tseng 1997]. Batching simply services all waiting requests for a video using one multicast stream. Stream Merging techniques, such as *Patching* [Hua et al. 1998], *Transition Patching* [Cai and Hua 1999], and *Earliest Reachable Merge Target* (ERMT) [Eager et al. 1999], further reduce the delivery cost by aggregating requests into larger groups that share the same multicast streams. These techniques are discussed in Section 2. Whereas Batching and Stream Merging techniques deliver data in a *client-pull* fashion, Periodic Broadcasting techniques employ the *server-push* approach. In particular, they divide each supported video into multiple segments and broadcast them periodically on dedicated channels. (A channel is a logical entity on which specific multimedia data is transmitted at a certain rate.) 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 become underutilized when videos are requested infrequently. This paper considers the client-pull approach.

Efficient support for a wide spectrum of clients with varying download bandwidth and buffer space is required so as to provide successful video streaming services over the Internet. Measurements of Internet client bandwidth [BroadBandReports 2006] 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 time of connection also impact the actual available bandwidth. Moreover, the available client buffer space varies significantly, considering the wide variety of devices that support video streaming. The variability in the resources and capabilities of clients is referred to as client heterogeneity.

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 channels, each at the video playback rate. Hence, if the video playback rate is r (measured in bits-per-seconds), Batching and Stream Merging require the client download bandwidth to be at least r and $2r$, respectively. *Bandwidth Skimming* [Eager et al. 2000] is an exception among Stream Merging techniques. It allows client download bandwidth to be lower than double the video playback rate, through special video encoding or complex data delivery, but still assumes receiver homogeneity in terms of download bandwidth. Client heterogeneity has not been addressed except by a few, recent studies, such as HeRo [Bagouet et al. 2003], BroadCatch [Tantaoui et al. 2004], and OHPB [Sessini et al. 2006], which use the server-push approach. With all these techniques, however, clients with bandwidth capacities lower than double the video playback rate are likely to face significant startup delay times compared to the video length.

This article studies how to support heterogeneous receivers while delivering video streams in a client-pull fashion. We propose three solutions to address the variability of the download bandwidth among clients: *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). It

classifies clients into two bandwidth classes. Clients with bandwidth capacities less than double the playback rate are serviced using Batching and the rest are serviced using Patching/ERMT. In contrast, AHS and EHS classify clients into multiple bandwidth classes and service them accordingly. They employ new stream types to service clients with bandwidth capacities ranging between the video playback rate and double the video playback rate. These streams are called *adaptive streams* and *enhanced adaptive streams*, respectively. Whereas the streams in Batching, Patching, and ERMT are all delivered at the video playback rate, adaptive streams are delivered at variable rates that are lower than the playback rate. Similarly, enhanced adaptive streams are delivered at variable rates but in two transmission phases. They are delivered initially at rates higher than the playback rate and subsequently at rates lower than the playback rate. The specific rates at which these two stream types are delivered depend on the corresponding client bandwidth capacities. 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 the variability in the available buffer space among clients. Moreover, we study how the waiting playback requests for different videos can be scheduled for service in the heterogeneous environment, capturing the variations in client bandwidth and buffer space. Scheduling refers to the selection of a subset of requests to be serviced whenever server resources become available. Furthermore, we discuss the applicability of the proposed solutions with layered video coding, which encodes the video into a base layer and one or more enhancement layers.

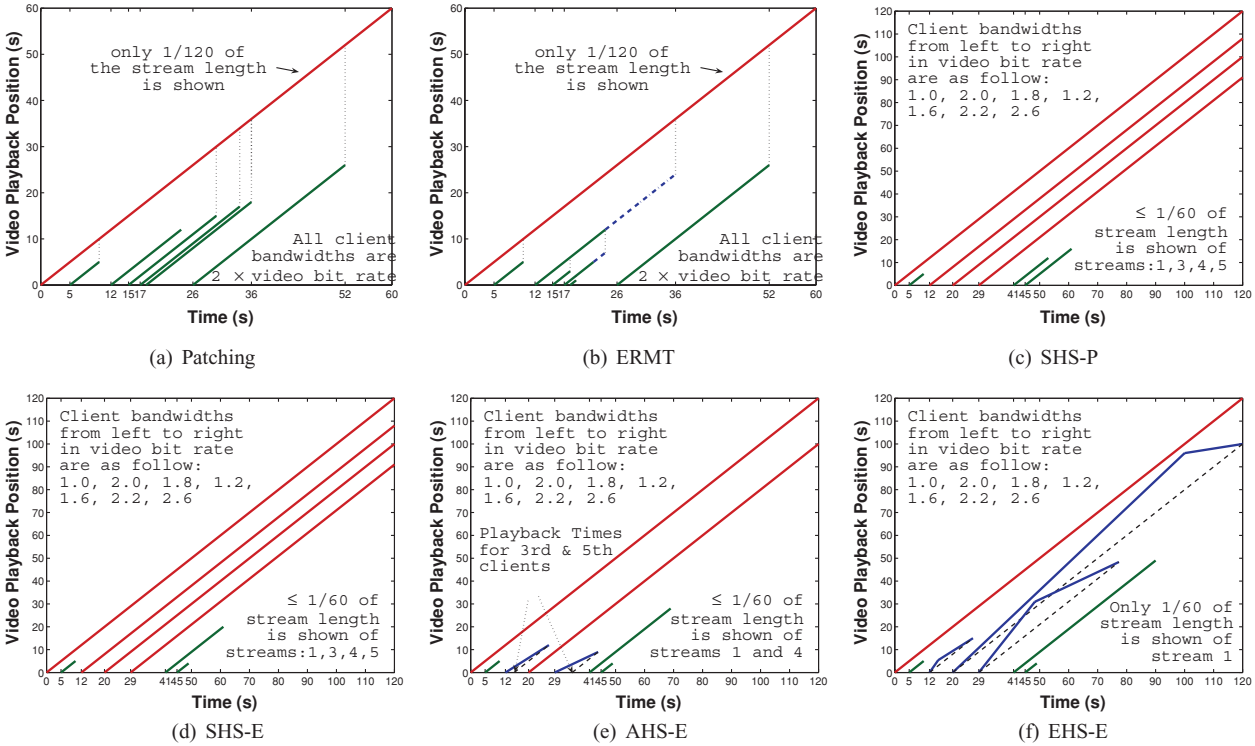
We evaluate the effectiveness of the proposed schemes and analyze various scheduling policies through extensive simulation. We study the impacts of many parameters, including client bandwidth and buffer space distributions, server capacity (i.e., server upload bandwidth), request arrival rate, customer 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 article is organized as follows. Section 2 provides background information. Sections 3 and 4 discuss the proposed support for heterogeneous client download bandwidth and available buffer space, respectively. Subsequently, Section 5 discusses request scheduling in the heterogeneous environment. Section 6 briefly discusses the applicability of the proposed solutions with layered video coding. Finally, Section 7 discusses the performance evaluation methodology and main results.

2. BACKGROUND INFORMATION

2.1 Resource Sharing

As discussed earlier, resource sharing techniques reduce the delivery cost (in terms of the total bits or seconds of the delivered video data) by using the multicast facility. Multicast allows requests to share the same streams, thereby increasing the sharing of server and network resources. The main resource sharing techniques include Batching, Patching, and ERMT. Batching simply services all waiting requests for a video using one full-length multicast stream, called *regular stream*. Thus, it requires only one download channel. Because streams are delivered at the video playback rate (r), the required client download bandwidth is r. The playback rate is measured in bits per second. Patching expands the multicast tree dynamically to include new requests. A new request joins the latest regular stream for the video and receives the missing portion as a *patch* stream at the video playback rate. Hence,



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

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

Fig. 1. Stream merging techniques.

it requires additional buffer space and two download channels, leading to a total required download bandwidth of $2r$. 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 prespecified value, called *regular window* (W_{reg}). Figure 1(a) further explains the concept of Patching in servicing 7 requests for a specific 2-hour video. This figure shows the video playback positions delivered by different streams versus time. In the example, all clients have $2r$ of download bandwidth. The first client arrives at time 0 and starts receiving the video as a 2-hour regular stream. The second client arrives after 5 seconds and starts to receive the existing regular stream immediately. Meanwhile, it receives the missing 5 seconds of video data as a 5-second patch stream. The third client arrives 12 seconds after the first, and receives the missing 12 seconds of data by another patch. The later clients are serviced in a similar manner.

ERMT is a near-optimal hierarchical Stream Merging technique. It also requires $2r$ of client download bandwidth and additional client buffer space, but it makes each stream sharable by later clients and thus leads to a dynamic merge tree (whereas Patching shares only regular streams with later clients). A new client joins the closest reachable stream (*target*) and receives the missing portion by a new stream (*merger*). A target stream is reachable if the client can merge with it before the target terminates. When more than one reachable stream is possible, the closest to the client's request arrival time is selected. After the merger stream finishes and merges into the target, the latter may get extended to satisfy the playback requirement of the new client(s), and this extension may affect its own merge target. For example, in Figure 1(b), Stream 3 started initially as a 12-second patch and then got

extended twice. The first time was at time 18 seconds, when Stream 4 merged into it, and it became the primary stream for the 4^{th} client. The 4^{th} client joined at time 15 seconds and started receiving data from both Stream 4 (as the merger) and Stream 3 (as the target). Stream 4 was scheduled to deliver only the missing 3 seconds from Stream 3. When Stream 4 finished at time 18 seconds, the 4^{th} client continued listening to Stream 3 and joined Stream 1 (the merge target for Stream 3). Because Stream 1 was then delivering data at playback position 18 seconds, Stream 3 had to be extended by additional 6 seconds to cover for the missing data from Stream 1. Stream 3 got extended again when Stream 5 merged into Stream 3 to cover for data missed by the 5^{th} and the 6^{th} clients.

2.2 Request Scheduling

To facilitate scheduling, a waiting request queue for each video is maintained in the main memory of the server. All requests in a selected queue are serviced using only one stream, whenever available resources become available. The main scheduling policies include *First Come First Serve* (FCFS) [Dan et al. 1994], *Maximum Queue Length* (MQL) [Dan et al. 1994], *Maximum Factored Queue Length* (MFQL) [Aggarwal et al. 2001], and *Minimum Cost First* (MCF) [Sarhan and Qudah 2007]. 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 to serve. The cost is measured as the number of seconds of video data to be delivered. We adopt this specific definition of the delivery cost in this paper and use the preferred implementation of MCF, called *MCF-P*, which considers the cost per request (as opposed to MCF-T, which considers only the total cost).

3. SUPPORTING VARIATIONS IN DOWNLOAD BANDWIDTH AMONG CLIENTS

We discuss next three proposed solutions for supporting variations in download bandwidth among clients. 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.

3.1 Simple Hybrid Solution (SHS)

Batching and all existing Stream Merging techniques treat clients as if they all have the same download bandwidth. Batching requires the client download bandwidth to be at least r, whereas Stream Merging techniques generally require at least $2r$. A simple and straightforward solution to support heterogeneous clients is to combine different resource sharing techniques. Hence, the proposed *Simple Hybrid Solution* (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. This leads to two alternatives: *SHS-P* (when Patching is used) and *SHS-E* (when ERMT is used). ERMT is the most efficient and Patching is the simplest among all Stream Merging techniques that require client bandwidth of $2r$ [Qudah and Sarhan 2006a].

Figures 1(c) and 1(d) further explain how SHS-P and SHS-E work. Clients (or a group of clients) with bandwidth capacities of $2r$ or greater (2^{nd}, 6^{th}, and 7^{th} in the figures) are serviced with Patching or ERMT, while those with lower bandwidth (1^{st}, 3^{rd}, 4^{th}, and 5^{th}) are serviced using Batching. Batching streams eliminate the need for some regular streams in Patching and reduce 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.

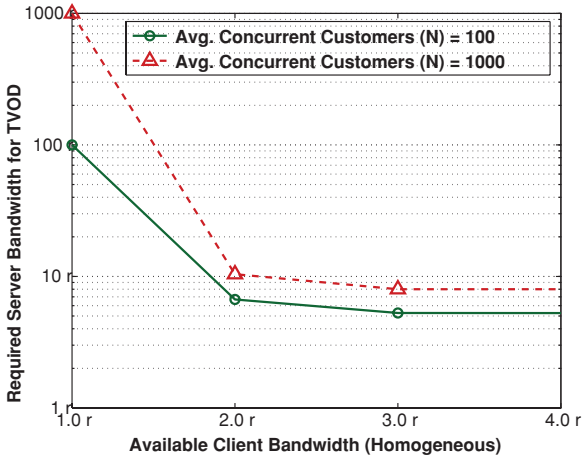


Fig. 2. Lower bound of server bandwidth requirement.

3.2 Adaptive Hybrid Solution (AHS)

As discussed earlier, 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 $2r$. Hence, it does not capture important opportunities for resource sharing within the first class because all clients in that class are dealt with in the same manner. For the second class, utilizing higher bandwidth than double the playback rate (i.e., $2r$) is not expected to lead to worthwhile performance benefits. The results in Eager et al. [2001] 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. Based on the analysis in Eager et al. [2001], Figure 2 shows the lower bound for server bandwidth requirement in servicing 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.

The question now arises as to how to utilize the extra client bandwidth in the first class of clients (with bandwidth capacities equal to or higher than r but lower than $2r$). This can be achieved by using streams with bit rates lower than the video playback rate and setting their rates based on the corresponding clients. Thus, 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 will 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 will 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. 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

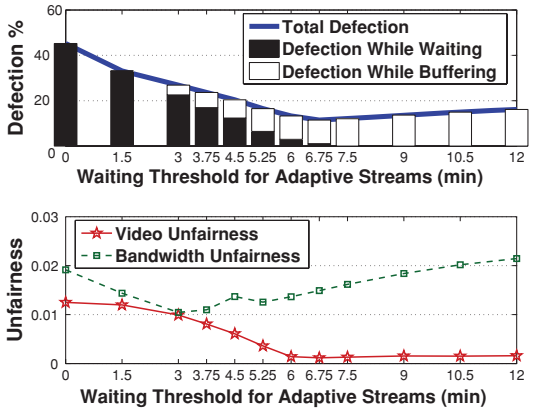


Fig. 3. Effect of AHS waiting threshold [MQL, server capacity = $1000r$].

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

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

Since T_{buf} is predictable, clients will 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 takes only positive real values and 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 3 further illustrates the impact of the waiting threshold on system performance under the default workload discussed in Section 7 and summarized in Table I. The available server capacity is assumed to be $1000r$.

To further reduce the additional waiting time caused by buffering, the clients perform "early snooping" on Batching streams. Basically, each client starts snooping on (i.e., listening to, or joining) the latest Batching stream for the video as soon as the client issues the request as opposed to waiting until admission 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 clients 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 3^{rd} and the 5^{th} clients are serviced using a-streams instead of Batching streams,

Table I. 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 μ_{tol} Time-of-service guarantee with threshold μ_{tol}
Waiting Tolerance Mean (μ_{tol})	0.5, **1.5** (default), 2.5 min
Targeted Maximum Client Waiting (T_{max})	**0.0** (default), 1.5, 3.0 min
Client Bandwidth Model	Zipf-like with θ_B = 0 and 0.5 Normal with mean μ_B = $1.5r$, **2.0r** (default), and $2.5r$, $\sigma_B = 0.5r$
Client Bandwidth Range	r to $11r$
Bandwidth Class Width	$0.05r$
Client Buffer Space Model	Fixed **60** min (default) Uniform Zipf (with $\theta_s = 0$) Normal with mean μ_S = 10, 30, and 60 min, $\sigma_S = 15$ min
Buffer Space Range	5 to 120 min
Buffer Space Class Width	10 sec

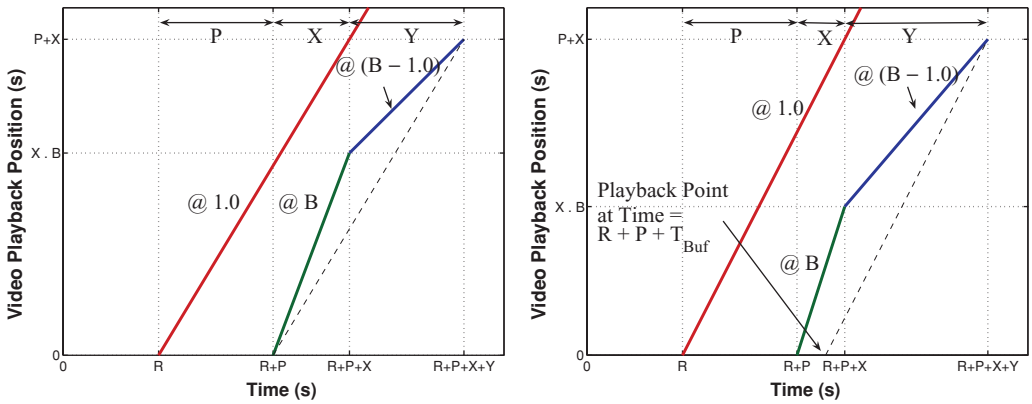
which reduces the cost significantly compared to SHS-E. The buffering times are 3 seconds for the 3^{rd} client and 6 seconds for the 5^{th}. Servicing the 4^{th} client using an a-stream would have resulted in 80 seconds of buffering time, which exceeds the assumed threshold, and thus, it is serviced by Batching.

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 the 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 resources to become available and the buffering time for smoothing out the variation in the end-to-end packet delay (delay jitter).

Figure 4(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



(a) With No Controlled Buffering Time ($T_{max} = 0$) (b) With Controlled Buffering Time ($T_{max} > 0$.)

Fig. 4. Enhanced adaptive streams.

must be satisfied. First, the last playback position (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 playback start time. 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 video data is determined by

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

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

The use of ea-streams is 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) illustrates the operation of EHS-S. In this figure, EHS-E is used to service the same requests as those discussed in Figures 1(c), 1(d), and 1(e). With EHS-E, the cost is further reduced, compared to both SHS-E and AHS-E.

3.4 Generalization of the Enhanced Adaptive Streams

We propose a generalized version of ea-streams that allows for some *controlled* buffering time (T_{buf}) to reduce the size of the delivered data. This controlled buffering time may be suitable for a Near Video-on-Demand (NVOD) service. Let us now assume that T_{max} is the targeted maximum client waiting time, and a client has already waited T_{org} seconds before being admitted for service. Subsequently, $T_{buf_{max}} = max(T_{max} - T_{org}, 0)$. In the generalized version, if an a-stream can achieve a smaller T_{buf} than $T_{buf_{max}}$, then it is used instead of an ea-stream. Thus, $X = 0$, $Y = \frac{P}{B-1}$, and the cost is simply P. Otherwise, an ea-stream is used. In this case, T_{buf} is set to $T_{buf_{max}}$ and the following two equations must be satisfied, as illustrated in Figure 4(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 for the ea-stream 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 lower than the video length. Otherwise, a new regular stream must be delivered instead.

3.5 Optimal Control of Regular Streams

Equations (6) and (11) highlight the importance of minimizing the average value of time skewness P since the last regular stream. Let us assume that for a certain D-second video, the request arrival rate is λ and the average time between two consecutive regular streams is W_{reg}. Thus, the number of requests arriving per video length is given by $N = \lambda D$ and the number of regular 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 is given by

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

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

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

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

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

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

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

Therefore, when the delivery cost is minimized, the total cost of ea-streams is equal to the total cost of regular 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 regular streams dynamically by computing the total cost of a-streams or ea-streams for each video since the last regular stream and initiating a regular stream for servicing the new requests for the video if the computed value exceeds the cost of a regular stream.

3.6 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 deals much better with this problem because of the fast delivery period of ea-streams. These streams 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 r. Before the end of the fast delivery period, X and Y should be recomputed using the client bandwidth history, and thus the fast delivery period will 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 in average bandwidth, where $0 \le \delta_B \le B-1$, the new fast delivery period $\overline{X}$ is calculated using

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

The additional cost is $\frac{\delta_B}{B-1}(P+T_{Buf})$. 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 average bandwidth drops beyond δ_B during the slow delivery period, the server can switch the service back to the fast 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 regular stream. A drop in bandwidth below r for Batching clients (or any other clients) cannot be solved unless a layered video coding is used and the video quality is reduced for some time to fit the client available bandwidth.

4. DEALING WITH VARIATIONS IN AVAILABLE BUFFER SPACE AMONG CLIENTS

The variation in the buffer space among clients is another concern, especially with the wide variety of devices capable of streaming videos. The available buffer space may be limited because of having a small buffer size, consumption of the buffer by user's applications and data, and/or requesting a high-quality video or a video with a low compression rate. For all Stream Merging techniques, including those proposed in this paper, the required client buffer space is $P+T_{buf}$. Recall that P is time skew of the request arrival from the last regular stream and that T_{buf} is the buffering time. As discussed earlier, $T_{buf}=0$ for all stream types other than a-streams and ea-streams with controlled buffering time (i.e., $T_{max}>0$).

Buffer space comes next in importance after the download bandwidth as long as some buffer space is available in each client. This is due to the following two reasons. First, the download bandwidth is generally the bottleneck and is decided by more than the receiving device. Second, while scalable resource sharing techniques (such as Patching and ERMT) require strict download bandwidth at almost every client, they are more tolerant of smaller buffer spaces. 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 will either defect or wait up to W_{reg} seconds for the next regular stream ($W_{reg}/2$ seconds on the average). Alternatively, the server needs to initiate regular streams earlier, but in this case, Patching may effectively lose its advantage over Batching if the clients with such bandwidth capacities are common. The situation is different when it comes to buffer space. In particular, a client with buffer space of S seconds of data that is slightly less than the required buffer space of W_{reg} seconds of data can be serviced as long as its request arrives within S seconds from the last regular stream. Otherwise, it must wait up to $(W_{reg} - S)$ seconds for the next regular stream, or the next regular stream must be initiated earlier.

In SHS, AHS, and EHS, the expected waiting time is usually smaller than in Patching because regular streams (Batching streams) will be delivered as well to support clients in the lowest bandwidth class, which results in a smaller effective W_{reg}. The option of initiating a regular stream earlier is generally preferred over forcing clients to wait, given that the server resources permit such an earlier service, but this leads to additional complications in request scheduling. In this article, we use this preferred option.

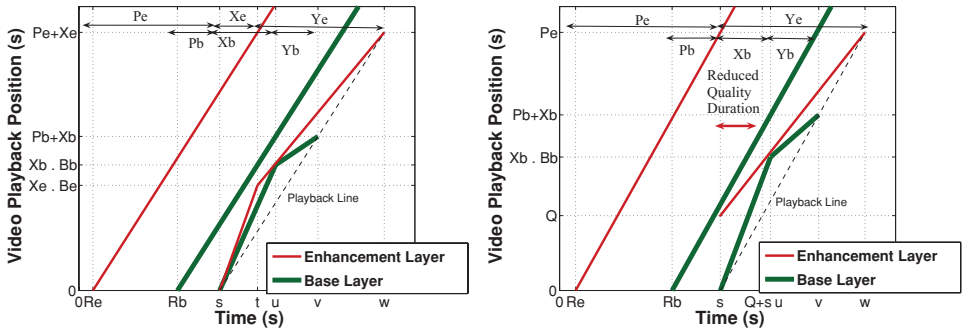
Handling the variations in buffer space is relatively easy to implement in the proposed hybrid resource sharing solutions but they along with the variations in download bandwidth lead to significant complications in request scheduling. In the next section, we discuss how request scheduling can be performed and implemented efficiently in the heterogeneous client environment.

5. REQUEST SCHEDULING FOR HETEROGENEOUS-CLIENT ENVIRONMENT

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. Therefore, 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 and that the corresponding clients have r, $1.5r$, and $2r$ in download bandwidth, respectively. Assuming sufficient buffer space in all three, the server has three service options for video v: (i) servicing all three requests by Batching, (ii) servicing only R2 and R3 by a-streams or ea-streams, and (iii) servicing only R3 by Patching or ERMT. Thus, a scheduling policy in the heterogeneous environment has to select not only a specific video among all videos that have waiting requests but also the specific subset of requests in that video.

To implement scheduling in the heterogeneous environment, 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 requests for video v by clients 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, assuming a bandwidth class width of $0.05r$, *Class 0* is 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. Assuming a buffer space class width of 30 seconds, *Class 0* is for clients with buffer space within $[0, 30[$, *Class 1* for clients with buffer space within $[30, 60[$, and so on.

We propose to perform scheduling by dividing the requests in each video queue into virtual queues. Instead of selecting a video queue as in the homogeneous case, a scheduling policy selects a virtual queue in the set of all possible virtual queues (among all videos).



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

Fig. 5. Options for supporting layered video coding.

The existing scheduling policies, discussed in Section 2.2, can be used in the heterogeneous environment but with possible modifications to incorporate the virtual queue concept. Because of their nature, MQL and FCFS will still act the same as in the homogeneous case. In particular, they will always select for service *all* requests in a particular video queue and will not utilize the variations in client bandwidth and buffer space. The selected virtual queue is basically the one that contains all waiting requests in the video queue because it is the longest (in the case of MQL) or the one containing the oldest request (in the case of FCFS). MFQL and MCF-P, however, exploit these variations by operating at the virtual queue level, and thus not all waiting requests for a video are necessarily selected. To adapt MFQL to the heterogeneous environment, we modify the factored length to be equal to the result of dividing the virtual queue length (as opposed to the video queue length) by the square root of the relative access frequency of the 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.

6. UTILIZING LAYERED VIDEO CODING

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 is expected to provide a more efficient overall solution. Although the detailed investigation of this combination is left for future work, we provide here some details to shed the light on the applicability of the proposed solutions with layered coding. These solutions can operate simply on each layer independently to support clients with a wider range of bandwidth, including those incapable of watching the full-quality video. Additionally, using layered coding can provide additional options in servicing clients with higher bandwidth capacities. For example, assuming only two video layers, the server may provide these clients with (i) full video quality, (ii) reduced video quality, or (iii) full quality after an initial period of reduced quality. The first and last options are clarified by Figure 5. Note that the new request arrived at time s and that the b and e subscripts denote the base layer and the enhancement layer, respectively.

7. PERFORMANCE EVALUATION METHODOLOGY AND RESULTS

We have developed a simulator using the C language for a video streaming server that supports various resource sharing techniques and scheduling policies. The simulator has been validated by reproducing

graphs in several previous studies [Dan et al. 1994; Hua et al. 1998; Cai and Hua 1999; Eager et al. 1999], and when possible, comparing the results with those predicted by analytical models [Hua et al. 1998; Eager et al. 2001]. The simulation stops after a steady state 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*, *video unfairness*, *client-bandwidth unfairness*, and *buffer-space unfairness*. 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. The unfairness metrics quantify the bias against unpopular videos, clients with lower available download bandwidth, and clients with smaller available buffer space, respectively. They can be found as the standard deviation in the defection rate among various videos, bandwidth classes, or buffer space classes, respectively. The used class widths are shown in Table I. In TVOD, we compare the server bandwidth required to service all requests immediately.

7.1 Workload Characteristics

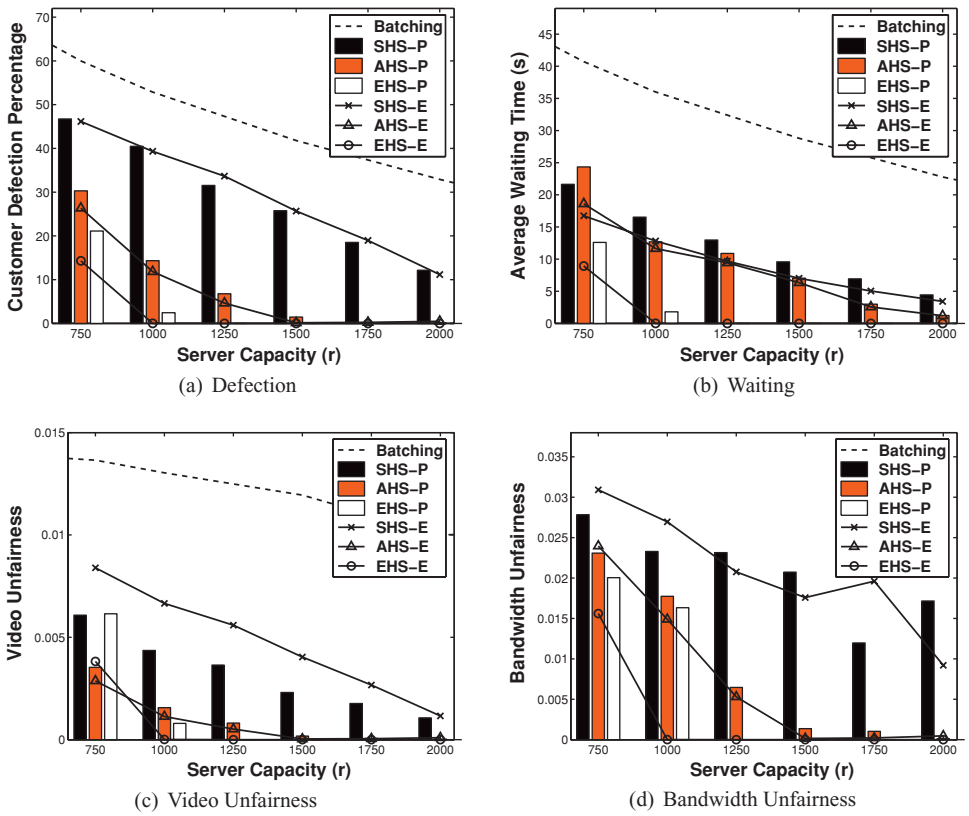
Like most prior studies, we assume that the arrival of the requests follows a Poisson Process with an average arrival rate λ a nd the accesses to videos are highly localized and follow a Zipf-like distribution with skewness parameter θ_v = 0.271. Generally, we characterize the waiting tolerance of customers by an exponential distribution with mean μ_{tol}. However, with AHS, we utilize the *time-of-service guarantee* (TSG) model [Alsmirat et al. 2007; Sarhan and Das 2004; Tsiolis and Vernon 1997] 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 I 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 minutes 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.

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 the video 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 six distributions: Fixed 60 minutes for all clients, Uniform, Zipf (with parameter $\theta_s = 0$), and three Normal distributions with $\mu_S = 10$, 30, and 60 minutes of video. The standard deviation of the Normal distributions is 15 minutes.

7.2 Result Presentation and Analysis

7.2.1 *Comparing the Performance of 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



(a) Defection (b) Waiting

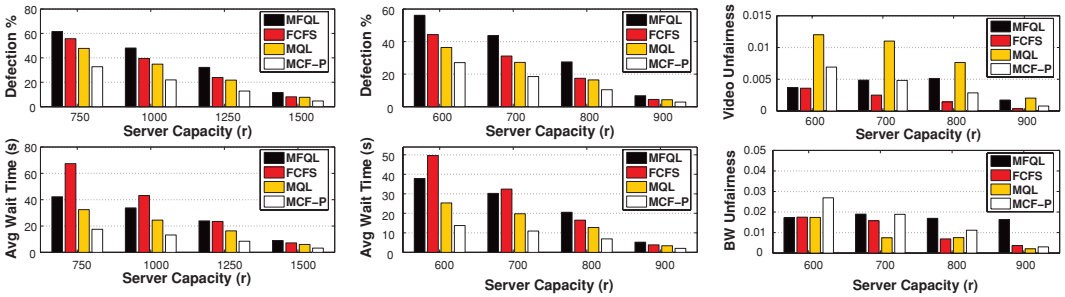
(c) Video Unfairness (d) Bandwidth Unfairness

Fig. 6. Comparing the performance of different resource sharing schemes supporting client bandwidth heterogeneity [Normal bandwidth dist. with $\mu_B = 2.0r$, MCF-P].

class unfairness. Figure 6 depicts the results under Normal bandwidth distribution with $\mu_B = 2.0r$ and Fixed buffer space distribution of 60 minutes at each client. At this stage, scheduling is performed using MCF-P, and EHS is implemented without requiring any additional waiting time due to buffering (i.e., $T_{buf} = 0$).

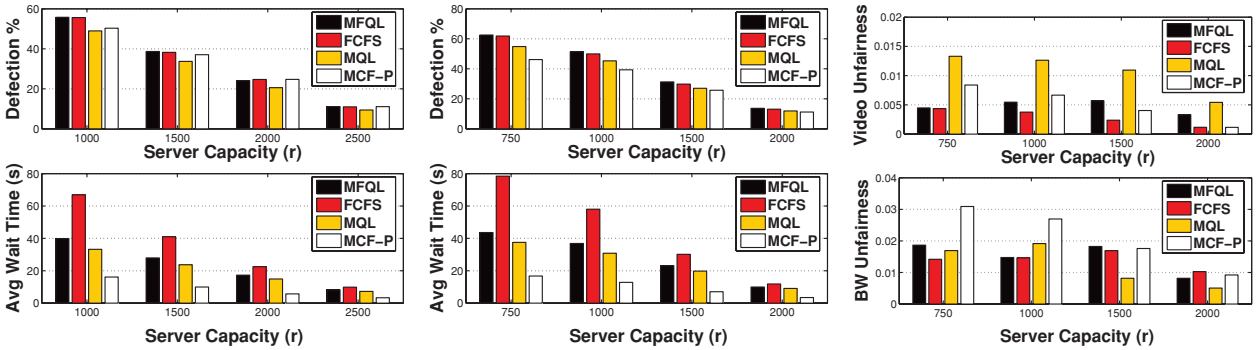
The results demonstrate the advantages of utilizing ea-streams in EHS and a-streams in AHS, with a noticeable lead of the former. For example, with $1000r$ in server bandwidth, when EHS-E achieves TVOD (i.e., 0 defections and 0 waiting time), AHS-E, SHS-E, and Batching cause defections of approximately 12%, 39%, and 53%, respectively. In contrast, AHS-E requires more than $1500r$ in server bandwidth to achieve 0 defections, and it can never completely eliminate the waiting times due to the requirement of the initial buffering time. On the other hand, SHS-E and Batching require approximately $2500r$ and $4800r$ in server bandwidth, respectively, to achieve TVOD (not shown in the figure). EHS has also the advantage of achieving the shortest waiting times, followed by AHS, SHS, and then Batching. 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 generally the fairest towards lower bandwidth classes. EHS and AHS, however, are generally the fairest among all schemes (including Batching) towards unpopular videos.

We observe that the three proposed solutions vary significantly in performance, whereas the two variants of each generally perform closer to one other. In particular, using ERMT instead of Patching



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

Fig. 7. Comparing the performance of scheduling policies for EHS-E.



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

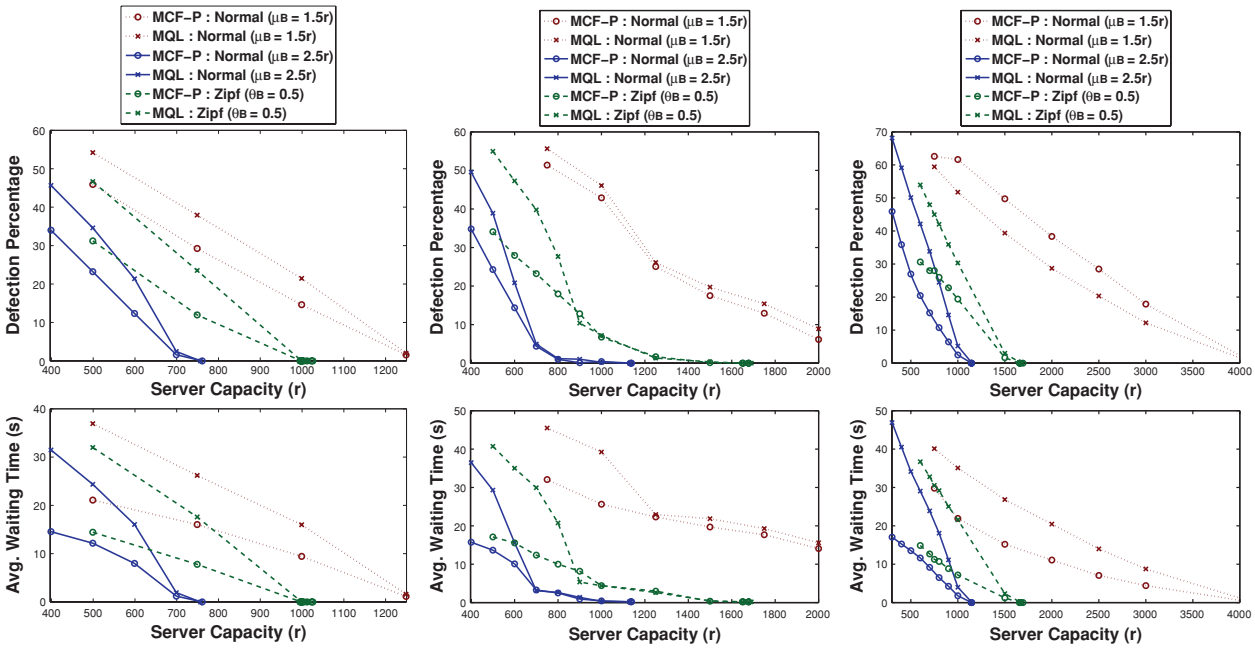
Fig. 8. Comparing the performance of scheduling policies for SHS-E.

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 r but lower than $2r$). However, the more efficient the solution is, the larger is the performance gap between its two variants.

From this point on, unless otherwise needed, only the ERMT variants of the three solutions are considered: EHS-E, AHS-E, and SHS-E.

7.2.2 *Comparing the Performance of Scheduling Policies.* Let us now discuss the impact of scheduling. Figures 7 and 8 compare scheduling policies when applying EHS-E and SHS-E, respectively. The results for AHS-E are very similar in their trend to EHS-E. The results are shown under two bandwidth distributions: Zipf with $\theta_B = 0.0$ and Normal with $\mu_B = 2.0r$. FCFS and MFQL in general perform worse than MCF-P and MQL in terms of defection rate and waiting time. Figure 9 compares scheduling policies under the remaining bandwidth distributions but in only the two main metrics (defection rate and waiting time) to reduce the number of figures. 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 because it schedules requests for service based on their delivery costs, which depend on their available download bandwidth. Overall, MCF-P is generally the best choice with EHS and AHS. The



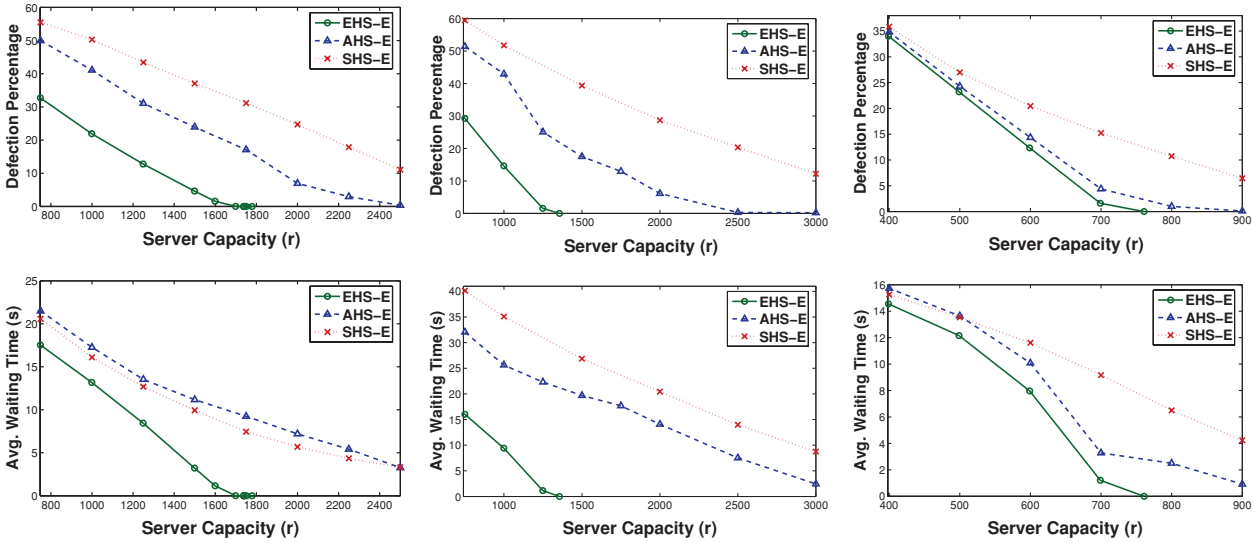
(a) EHS-E (b) AHS-E (c) SHS-E

Fig. 9. Comparing the performance of scheduling policies with different bandwidth distributions.

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 $\mu_B = 1.5r$), 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, which 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($\mu_B = 1.5r$), 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.

7.2.3 *Comparing the Performance of Resource Sharing Schemes with Best Scheduling.* Figure 10 compares EHS-E, AHS-E, and SHS-E under three different bandwidth distributions. (Recall that Figure 6 compares all schemes under Normal bandwidth distribution with $\mu_B = 2.0r$.) Only the two most important metrics are analyzed here to reduce the number of figures. 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.



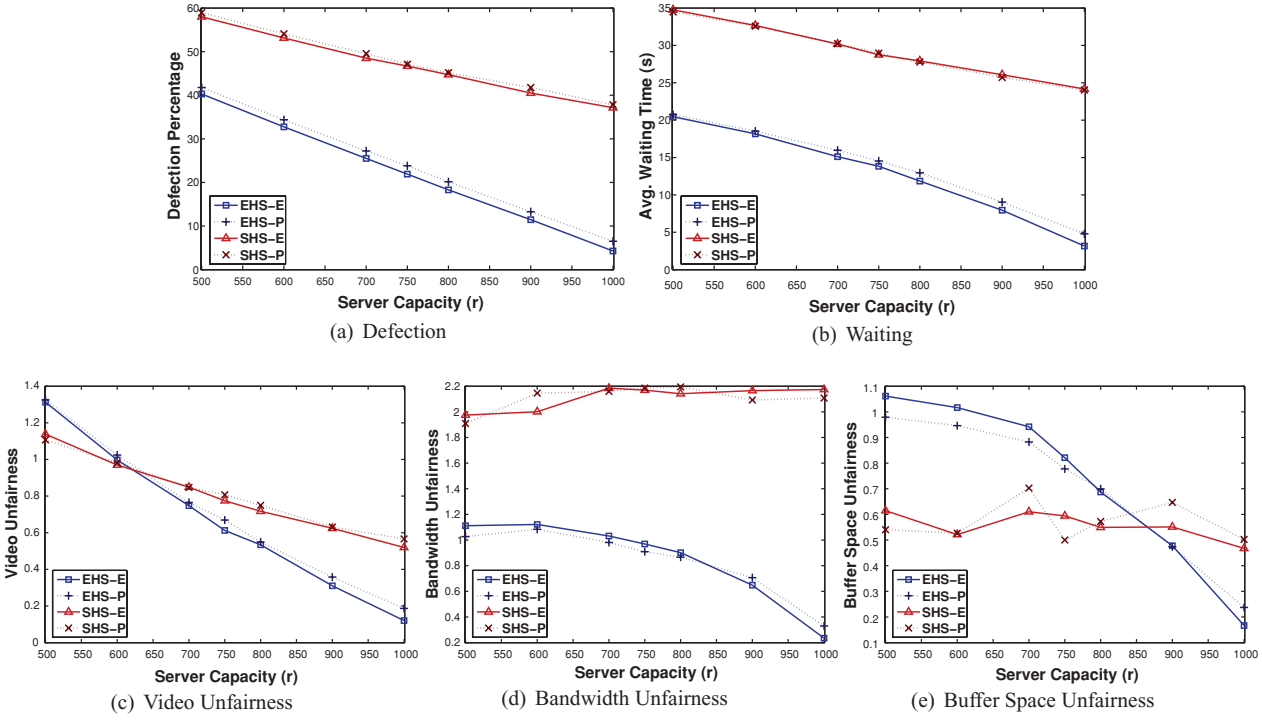
(a) Zipf ($\theta_B = 0.0$) (b) Normal ($\mu_B = 1.5r$) (c) Normal ($\mu_B = 2.5r$)

Fig. 10. Comparing the performance of the different hybrid solutions with best scheduling.

7.2.4 *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 11, limited buffer space and its diversity 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 12 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 $500r$) reduces the defection probability from 43% to 36%, and the average waiting time from 22 to 18 seconds. As expected, 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 regular (Batching) streams which indirectly helps the clients with lower bandwidths.

7.2.5 *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 13 plots the results under different combinations of buffer space availability levels for these three main resources. Table II provides additional insights. The main results can be summarized as follows.

(1) 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 $4800r$ in server bandwidth to achieve TVOD, while a 10-minute buffer space reduces the requirement



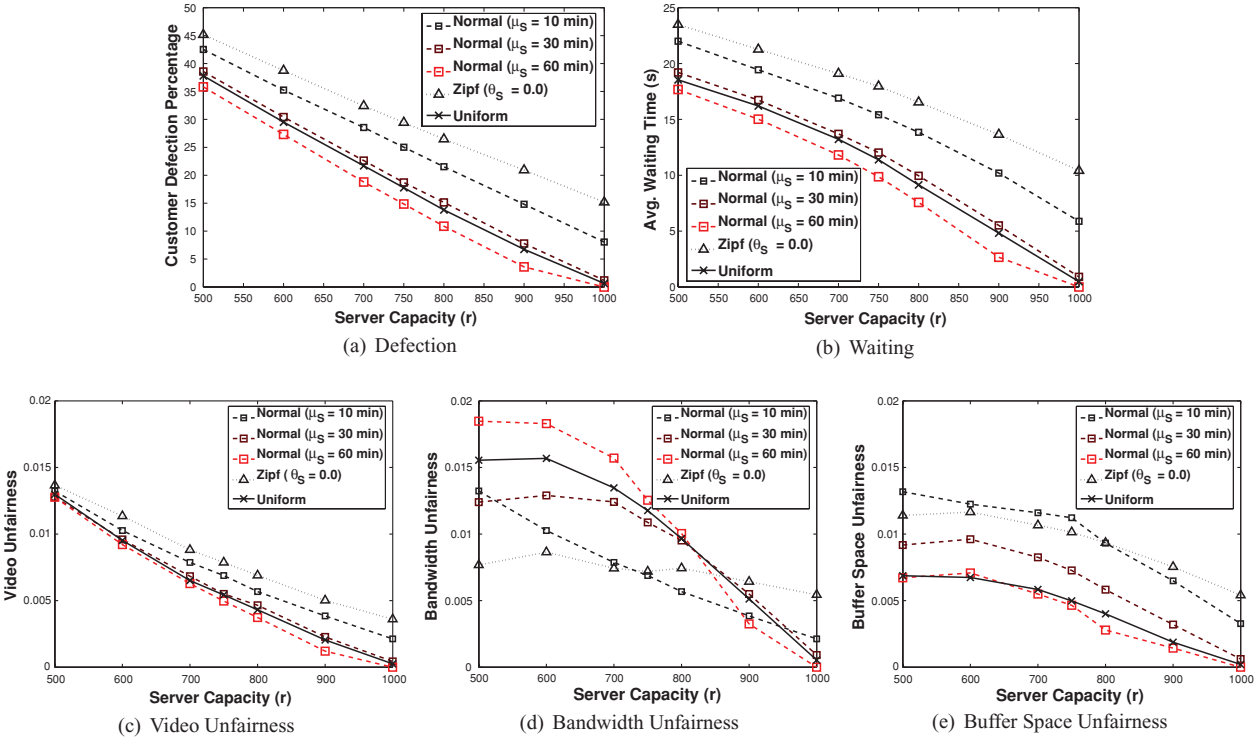
(a) Defection (b) Waiting (c) Video Unfairness (d) Bandwidth Unfairness (e) Buffer Space Unfairness

Fig. 11. Comparing the performance of various schemes with bandwidth and buffer space diversity [Normal bandwidth dist. with $\mu_B = 2.0r$, normal buffer space dist. μ_S = 20 min, MCF-P].

to approximately $1331r$. Doubling that average buffer space, however, reduces this requirement to $1295r$ (i.e., by only 3%). Even increasing it to six times that average buffer space reduces the requirement to $1230r$ (only 7.5%). In contrast, increasing the average download bandwidth from $1.5r$ to $2.0r$ (approximately 33%) reduces the required server capacity to $1116r$ (approximately 11%). Similarly in NVOD, when the server capacity is $500r$, increasing the buffer space from 10 minutes to 60 minutes reduces the defection rate approximately from 42.5% to 36%, while increasing the server capacity to $600r$ results in a defection rate of 35%.

(2) 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 $500r$. To achieve comparable results without requiring that much client bandwidth, the server capacity needs to be increased to more than $800r$ (more than a 60% increase). The defection rate at $800r$ is 25%.

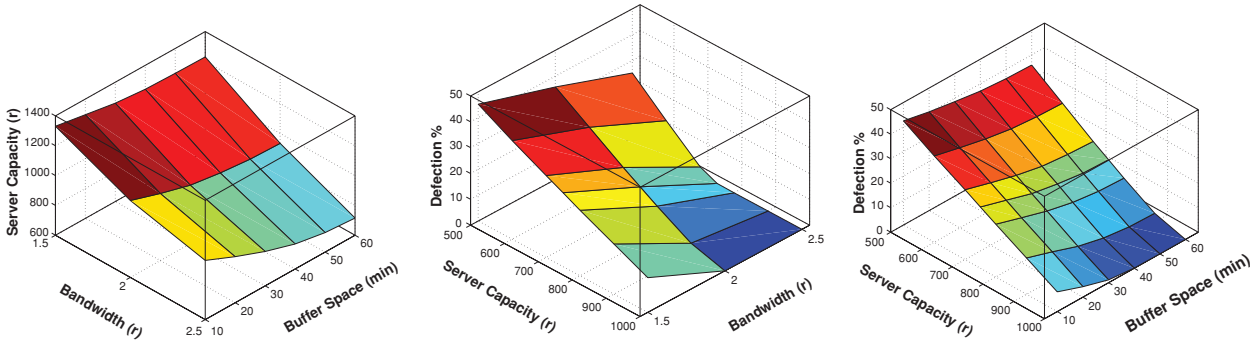
7.2.6 *Impact of Customer Waiting Tolerance.* Let us now discuss the impact of customer waiting tolerance. Figure 14 plots the percentage improvements achieved by EHS-E and AHS-E compared with SHS-E in terms of the defection percentage and the waiting time for different values of μ_{tol}. Four main conclusions 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



(a) Defection (b) Waiting

(c) Video Unfairness (d) Bandwidth Unfairness (e) Buffer Space Unfairness

Fig. 12. Impact of buffer space distribution on EHS-E [Normal bandwidth dist. with $\mu_B = 2.0r$, MCF-P].



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

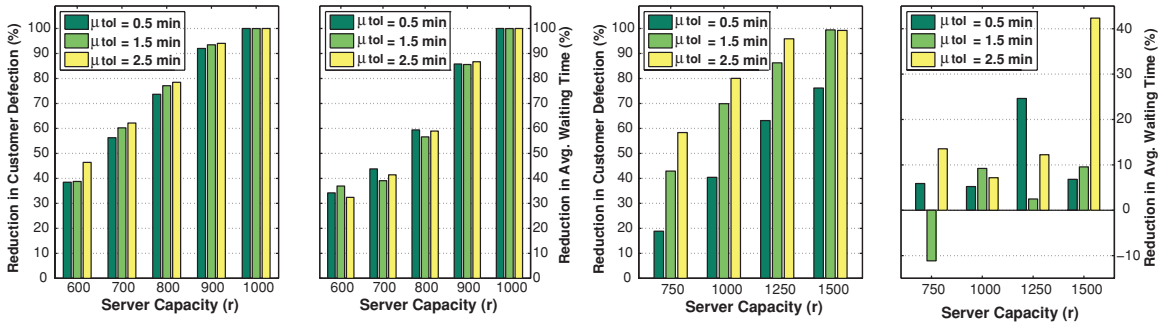
(b) Impact of Bandwidth and Server Capacity on Defection

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

Fig. 13. Impact of client bandwidth, client buffer space, and server capacity on EHS-E performance [Normal bandwidth dist. with default $\mu_B = 2.0r$, normal buffer space dist. with default $\mu_S = 60$ min, MCF-P].

Table II. Comparative results of the impacts of server capacity, client bandwidth, and client buffer space [normal bandwidth dist. with default $\mu_B = 2.0r$, normal buffer space dist. with default $\mu_S = 60$ min, EHS-E, MCF-P]

Input Parameters				Output		
Client Bandwidth Mean (r)	Buffer Space Mean (min)			Server Capacity (r)		
	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 (r)			Server Capacity (r)		
	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 (r) [Client Bandwidth Mean = $2.0r$]	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 (r) [Buffer Space Mean = 10 min]	Client Bandwidth Mean (r)			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



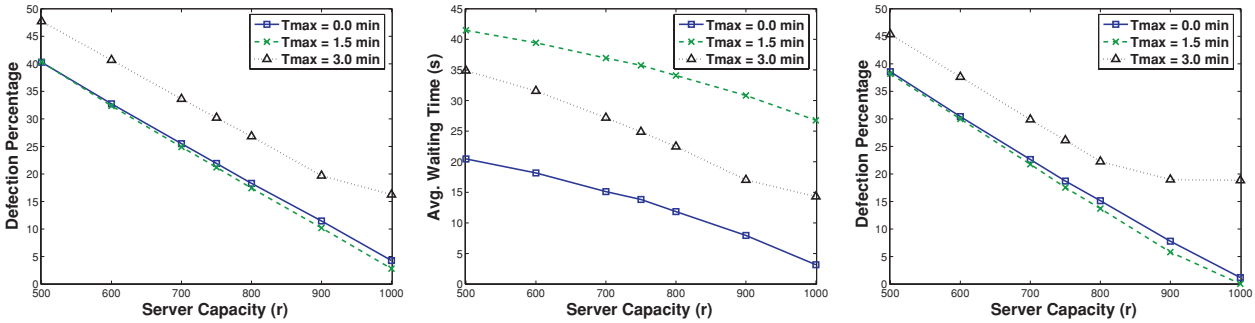
(a) EHS-E vs. SHS-E

(b) AHS-E vs. SHS-E

Fig. 14. Impact of customer Waiting Tolerance (μ_{tol}).

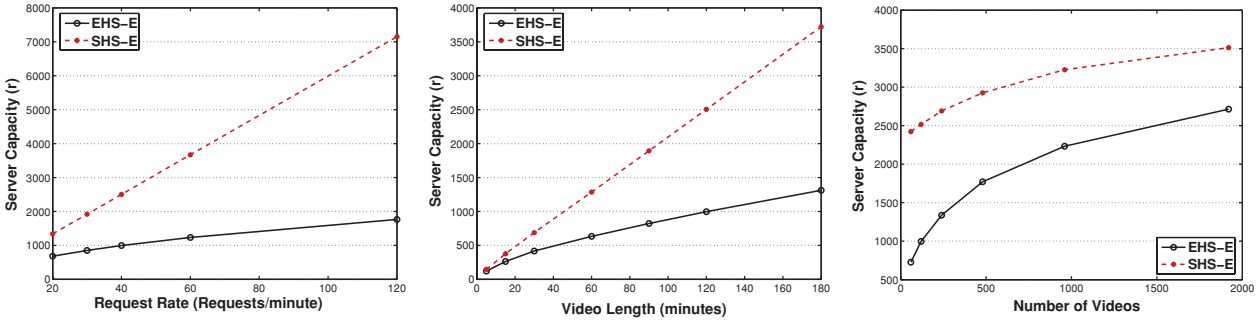
compared to it. (3) Although the main advantage of AHS over SHS is reducing the defection rate and servicing a larger number of customers, it also results in a shorter average waiting time. (4) Regardless of the average waiting tolerance of customers, EHS keeps its lead over the other solutions, especially in terms of the defection rate and waiting time.

7.2.7 *Impact of T_{max} with EHS-E.* As discussed in Section 3, EHS-E can be implemented with some additional buffering time (T_{buf}). Figure 15 shows the results for three values of targeted maximum client waiting time (T_{max}): 0.0, 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 controlled waiting time with EHS and to always target 0.0 waiting time. Interestingly,



(a) Defection (μ_S = 20 min) (b) Waiting (μ_S = 20 min) (c) Defection (μ_S = 30 min)

Fig. 15. Impact of T_{max} with EHS-E [normal bandwidth dist. with $\mu_B = 2.0r$, normal buffer space dist., MCF-P].

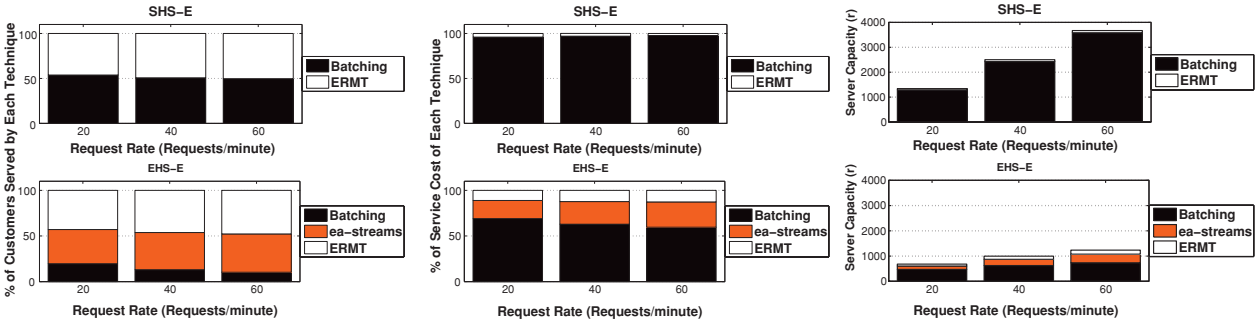


(a) Impact of Request Rate (b) Impact of Video Length (c) Impact of Video Number

Fig. 16. impacts of request arrival Rate, video length, and number of videos [TVOD, normal bandwidth dist. with $\mu_B = 2.0r$, normal buffer space dist. with $\mu_S = 30$ min].

the average waiting time is smaller for $T_{max} = 3.0$ minutes compared to $T_{max} = 1.5$ minute. This is because only serviced customers are considered in the average waiting time. Controlled 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.

7.2.8 *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.) Figure 16 compares the required server bandwidth to achieve TVOD by the two schemes versus request arrival rate, video length, and number of videos. The impact of the video length is similar to 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. 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 2.6 folds (from $682r$ to $1766r$), whereas the server bandwidth with SHS increases 5.3 folds (from $1342r$ to $7151r$). 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



(a) Usage Percentage (b) Cost Percentage (c) Cost

Fig. 17. Technique-level statistics in SHS-E and EHS-E [TVOD, Normal bandwidth dist. with $\mu_B = 2.0r$, normal buffer space dist. with $\mu_S = 30$ min].

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.

7.2.9 *Delving inside the Hybrid Solutions.* To gain better understanding of the hybrid solutions, particularly SHS-E and EHS-E, Figure 17 provides usage statistics on the individual techniques (Batching, ea-streams, and ERMT) composing these two solutions. These technique-level statistics include the percentage of served customers, the percentage of the delivery cost, and the delivery cost in terms of server channels. The main observations drawn from these results can be summarized as follows. First, in both solutions, especially SHS-E, as the request rate increases, the percentage of requests being served by ERMT approaches the percentage of clients having download bandwidth capacities of $2r$ or higher (referred to as high-end clients). This is due to the reduction in the average time between consecutive Batching streams and buffer space becoming a less important issue. Second, the percentage of the cost to serve the high-end clients is higher with EHS-E compared to SHS-E because of the increased actual cost of serving these clients (due to the longer average time between consecutive Batching streams in EHS-E) and the reduced cost of serving the others (due to utilizing ea-streams). And third, despite the increased cost of serving high-end clients with EHS-E compared to SHS-E, the overall cost is lower with EHS-E because of the bigger reduction in the cost of serving the other clients. With SHS-E, approximately half of the requests are served by Batching, whereas with EHS-E, less than 20% of the requests are served by Batching. EHS-E replaces 60% to 80% of SHS-E Batching streams with the cheaper ea-streams.

8. CONCLUSIONS AND FUTURE WORK

We have evaluated the effectiveness of the proposed schemes and have analyzed various scheduling policies through extensive simulation. The main results can be summarized as follows.

(1) The proposed schemes can achieve high degrees of resource sharing.

(2) AHS significantly outperforms SHS and Batching especially in the defection percentage. For example, it can provide a service with no customer defections, while SHS and Batching cause more than 25% and 41% defections, respectively.

(3) 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. For example, with the same server bandwidth, EHS-E can achieve zero defections, while AHS-E, SHS-E, and

Batching cause 12%, 39%, and 53% defections, respectively. EHS is also more tolerant than AHS to variations in client bandwidth during service.

(4) The three proposed solutions vary significantly in performance, whereas the two variants of each perform closer to one other. Using ERMT instead of Patching for serving clients with bandwidth capacities of $2r$ or higher (referred to as high-end clients) is of less significance than changing the solution (and thus the method of service for clients with bandwidth capacities higher than r but lower than $2r$). (5) The percentage of the cost to serve the high-end clients is higher with EHS compared to SHS because of the increased actual cost of serving these clients (due to the longer average time between consecutive Batching streams in EHS) and the reduced cost of serving the others (due to utilizing ea-streams). Despite this increased cost, however, the overall cost is lower with EHS because of the bigger cost reduction in serving the other clients. For example, with SHS-E, about half the requests are served by Batching, whereas with EHS-E, less than 20% are served by Batching.

(5) Although the client available buffer space has generally a less significant impact on system performance than the client download bandwidth, its impact can be very significant if it becomes a bottleneck.

(6) 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 schemes. 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.

In future work, we plan to study the effectiveness of combining the proposed resource sharing solutions with layered video coding. This combination is expected to provide a more efficient overall solution. In particular, we plan to enhance these solutions to capture additional resource sharing opportunities when layered video coding is used.

REFERENCES

AGGARWAL, C. C., WOLF, J. L., AND YU, P. S. 2001. The maximum factor queue length batching scheme for Video-on-Demand systems. *IEEE Trans. Comput. 50,* 2, 97–110.

ALSMIRAT, M., AL-HADRUSI, M., AND SARHAN, N. J. 2007. Analysis of waiting-time predictability in scalable media streaming. In *Proceedings of the ACM Multimedia Conference*. 791–794.

BAGOUET, O., HUA, K. A., AND OGER, D. 2003. A periodic broadcast protocol for heterogeneous receivers. In *Proceedings of the Multimedia Computing and Networking Conference (MMCN)*.

BROADBANDREPORTS. 2006. `http://broadbandreports.com/`.

CAI, Y. AND HUA, K. A. 1999. An efficient bandwidth-sharing technique for true video on demand systems. In *Proceedings of the ACM Multimedia Conference*. 211–214.

DAN, A., SITARAM, D., AND SHAHABUDDIN, P. 1994. Scheduling policies for an on-demand video server with batching. In *Proceedings of the ACM Multimedia*. 391–398.

EAGER, D. L., VERNON, M. K., AND ZAHORJAN, J. 1999. Optimal and efficient merging schedules for Video-on-Demand servers. In *Proceedings of the ACM Multimedia*. 199–202.

EAGER, D. L., VERNON, M. K., AND ZAHORJAN, J. 2000. Bandwidth skimming: A technique for cost-effective Video-on-Demand. In *Proceedings of the Multimedia Computing and Networking Conference (MMCN)*. 206–215.

EAGER, D. L., VERNON, M. K., AND ZAHORJAN, J. 2001. Minimizing bandwidth requirements for on-demand data delivery. *IEEE Trans. Knowled. Data Engine. 13,* 5, 742–757.

HUA, K. A., CAI, Y., AND SHEU, S. 1998. Patching: A multicast technique for true Video-on-Demand services. In *Proceedings of the ACM Multimedia*. 191–200.

JUHN, L. AND TSENG, L. 1997. Harmonic broadcasting for Video-on-Demand service. *IEEE Trans. Broadcast. 43,* 3, 268–271.

PÂRIS, J.-F. 2001. A fixed-delay broadcasting protocol for Video-on-Demand. In *Proceedings of the International Conference on Computer Communications and Networks*. 418–423.

QUDAH, B. AND SARHAN, N. J. 2006a. Analysis of resource sharing and cache management techniques in scalable video-on-demand. In *Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. 327–334.

QUDAH, B. AND SARHAN, N. J. 2006b. Towards scalable delivery of video streams to heterogeneous receivers. In *Proceedings of the ACM Multimedia*. 347–356.

SARHAN, N. J. AND DAS, C. R. 2004. A new class of scheduling policies for providing time of service guarantees in Video-On-Demand servers. In *Proceedings of the 7th IFIP/IEEE Int'l Conf. on Management of Multimedia Networks and Services*. 127–139.

SARHAN, N. J. AND QUDAH, B. 2007. Efficient cost-based scheduling for scalable media streaming. In *Proceedings of the Multimedia Computing and Networking Conference (MMCN)*. Vol. 6504. 65040C.

SESSINI, P., SHI, L., MAHANTI, A., LI, Z., AND EAGER, D. L. 2006. Scalable streaming for heterogeneous clients. In *Proceedings of the ACM Multimedia*. 337–346.

TANTAOUI, M. A., HUA, K. A., AND DO, T. T. 2004. Broadcatch: A periodic broadcast technique for heterogeneous Video-on-Demand. *IEEE Trans. Broadcast. 50,* 3, 289–301.

TSIOLIS, A. K. AND VERNON, M. K. 1997. Group-guaranteed channel capacity in multimedia storage servers. In *Proceedings of the ACM SIGMETRICS*. 285–297.

Received February 2008; revised September 2008, February 2009; accepted June 2009