Assessing the Efficiency of Stream Reuse Techniques in Peer-to-Peer Video-on-Demand Systems

Leonardo Bidese de Pinho and Claudio Luis de Amorim
Parallel Computing Laboratory
COPPE Systems Engineering Program
Federal University of Rio de Janeiro, RJ, Brazil
{leopinho,amorim}@cos.ufrj.br

Abstract

Many works have reported simulated performance benefits of stream reuse techniques such as chaining, batching, and patching to the scalability of VoD systems. However, the relative contribution of such techniques have been rarely evaluated in practical implementations of scalable VoD servers. In this work, we investigate the efficiency of representative stream reuse techniques on the GloVE system, a low-cost scalable VoD platform whose resulting performance depends on the combination of the stream techniques it uses. More specifically, we evaluated performance of the GloVE system on delivering multiple 2-hour MPEG-1 videos to clients connected to the server through a 100 Mbps Ethernet switch, and arrival rates varying from 6 to 120 clients/min. We present experimental results including startup latency, occupancy of server's channels, and aggregate bandwidth that GloVE demands for several combinations of stream reuse techniques. Overall, our results reveal that stream reuse techniques in isolation offer limited performance scalability to VoD systems and only balanced combination of chaining, batching, and patching explains the scalable performance of GloVE on delivering highly popular videos with low startup latency while using the smallest number of server's channels.

1. Introduction

In recent years, considerable research efforts have been concentrated on the design of scalable Video on Demand (VoD) systems since they represent a key-enabling technology for several classes of continuous media applications, such as distance learning and home entertainment. In particular, a central problem VoD designers face is that in a typical VoD System with many videos, even a transmission of a single high-resolution video stream in a compressed format, consumes a substantial amount of resources of both the video server and the distribution network, which limits scalable performance of the VoD system. Given that users can choose any video and start playback at anytime, a significant investment on server's hardware will be required to support large audiences. Therefore, it is fundamental that a VoD server supports stream distribution strategies capable of using efficiently the server's resources in order to reduce its cost/audience ratio.

Typically, a VoD server supports a finite pool of stream channels, where the amount of channels is often determined by the server's bandwidth divided by the video playback rate. The reason that Conventional VoD systems cannot scale is because they dedicate one different channel to each active client, in an one-to-one approach. As a result, the total number of clients a conventional VoD system supports is equal to the limited number of server's channels.

The efficiency of a VoD system can be characterized by two metrics: scalability and startup latency. The former accounts for the number of simultaneous active clients a VoD system can support on transmitting video streams without network jitters whereas the latter defines the average time difference between a client request and starting video playback.

Due to the scalability limitation of the one-to-one approach, several scalable streaming techniques have been proposed in the literature. Basically, such techniques allow multiple clients to share the stream contents that are delivered through each of the server's channels, in an one-to-many approach. Two important examples of scalable streaming techniques are Chaining [22], where earlier arriving clients transmit video streams to the new clients, forming stream chains across the network, and Patch-

ing [12], in which active clients provide extra short streams, namely the patches, to the other active clients. In previous work, we proposed a new streaming technique, which we called the Cooperative Video Cache (CVC) [13] that combines chaining and patching under a P2P model. Essentially, CVC extends the functionality of clients' playout buffers[1], by treating them as parts of a globally distributed memory that caches and supplies stream contents to active clients in a cooperative way. In [19, 20] we reported performance results of a VoD prototype we developed, namely the Global Video Environment (GloVE) that confirmed that CVC adds scalable performance to conventional VoD servers in practical environments such as video distribution in an academic department.

In contrast with existing works that evaluated performance of scalable streaming techniques through simulations, in this work we assess the efficiency of representative streaming techniques in practical situations. More specifically, since the GloVE prototype allows the evaluation of streaming techniques either separately or in any combination of them, we examined their relative performance contribution to the scalability of VoD systems that GloVE encompasses. In addition, we addressed the impact of the popularity distribution of videos on the effective use of VoD system's resources.

In summary, the main contributions of our work are:

- We assess the relative contribution of stream reuse techniques to the scalable performance of VoD systems in practical settings.
- We quantify the influence of video prefetching rate on the performance behavior of a VoD server.
- We evaluate the impact of the distribution of video popularity on VoD system's performance.

The remainder of this paper is structured as follows. Section 2 presents an overview of stream reuse techniques. In Section 3, we describe briefly the main characteristics of the GloVE platform. In Section 4 we analyze GloVE's performance results for several configurations of streaming techniques with various workloads. Section 5 describes related works. Finally, we conclude in Section 6.

[1]Clients use playout buffers to hide the network jitter and inherent variations of VBR videos

2. Stream Reuse Techniques

This section presents briefly scalable streaming techniques that are representative examples of the class of multicast-based stream reuse techniques that have been proposed in the literature[15]. These techniques can be categorized as proactive or reactive depending whether the VoD server transmits periodically video streams through its channels independently of client requests or the VoD server sends streams through its channels in response to client requests.

2.1. Proactive Streaming Techniques

Broadcasting. Usually, broadcast schemes [11] divide the video stream into segments that are transmitted periodically to clients through dedicated server's channels. In this way, while a video segment plays back, its subsequent segment will arrive on time, thus guaranteeing no interruptions to the video playback. Despite its high scalability, broadcasting wastes many channels and imposes high latency (typically in the order of hundreds of seconds).

2.2. Reactive Techniques

Batching. In this approach, requests issued within a certain time interval to the same video are first enqueued and are served afterwards by a single multicast stream [5]. Two alternatives can be used to service the queue: the queue's length and waiting time. In the former, a multicast stream is sent to the waiting clients only after the queue length reaches a determined value. In the later, arriving clients are inserted into the queue and waits until a determined time, after which a video stream will be transmitted to them. Note that the first client that requested the video will wait more time than the subsequent ones, which tends to create unfairness on startup latency, especially when batching allows relatively long queuing time in an attempt to reduce channel occupancy.

Patching. In this technique [12], the server first starts a full video multicast to the first client. It has to be a full stream transmission because the server has to provide all video segments to the first client.

However, subsequent clients that request the same video will be added to the existing video multicast group while storing temporarily the streaming contents they receive in their local buffers. Concurrently, the server will transmit to each new client an extra video stream, which is called the patch, that contains the initial video segment the new client missed. Once the patch arrives, the client can start to playback the video patch, after which it switches to its local buffer to playback the next video segments. Assuming the VoD system uses a stream rate equal to its playback rate, the duration of the patch stream will be proportional to the time interval between the first request and the new one. Thus, a potential disadvantage of patching is that it may require local buffers with high capacity to cope with large patches.

Chaining. The strategy this technique [22] uses is to create chains of client buffers. Specifically, as long as a certain client holds in its buffer the initial part of a video, a subsequent request for the same video can be served by the playout buffer of that earlier client rather than the video server. We assume clients start video playback from the beginning. Note that chaining follows the peer-to-peer model, which is highly scalable. In practice, however, the scalability of chaining depends on the aggregated bandwidth that the distribution network supports, since it limits the amount of active streams across the network.

Cooperative Video Cache (CVC). This technique implements a global video cache management over the video streams that are dynamically stored in the client's local buffers. In addition, it uses a combination of chaining and patching to reuse video streams from the global video cache. Basically, CVC reuses the distributed video contents by chaining client buffers while applying patches whenever these offer effective reuse of active multicast streams [13]. Also, CVC exploits the Batching [5] technique to enable clients that arrive in a given time interval to form multicast groups. More detail of CVC can be found in [19, 20].

In this work we focus on reactive streaming techniques, which allow often to combine low latency with high scalability.

3. The GloVE System

In this section, we describe briefly the main features of the GloVE (Global Video Environment) a scalable peer-to-peer VoD system.

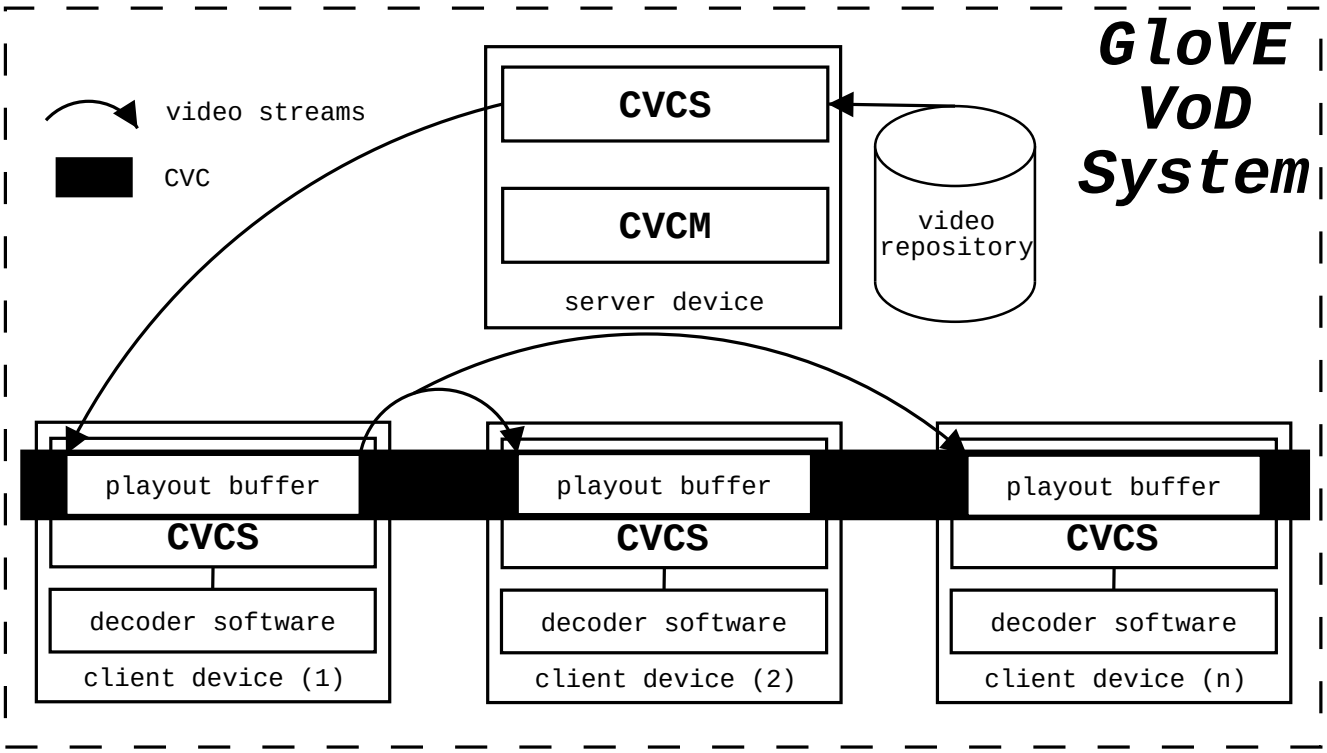

Figure 1. The architecture of the Global Video Environment (GloVE).

GloVE (Figures 1) implements the cooperative video cache while supporting a peer-to-peer system with a centralized metadata component, namely the CVC Manager (CVCM). The CVCM monitors globally the video contents stored in the playout buffers of the clients. The CVC Client (CVCC) software allows client buffer's contents to be shared with the other clients, reducing the demand on the CVC Server (CVCS). GloVE assumes that the communication network has IP multicast support and that the VoD server manages the video as a sequence of blocks with random access. Next, we describe the GloVE components in more detail.

CVC Server (CVCS). CVCS works as block-based, unicast-only video server. CVCS receives requests from clients and sends back 64K blocks to them using the UDP protocol over unicast transmission.

CVCS guarantees the arrival of blocks at the clients, and uses sub-block retransmission when necessary. Two types of server access are employed: coarse and smooth. With coarse accesses, clients request blocks without rate control, so that the response time is dependent on the available server's bandwidth which is inversely proportional to the amount of clients receiving streams from the server. With smooth server's accesses, clients request blocks at the playback rate.

CVC Client (CVCC). CVCC runs on the client device and manages its local playout buffer. CVCC uses multiple threads to implement the following main tasks: 1) Request/Receive blocks to fill the buffer, 2) Retrieve blocks from the buffer and send them to a third-part video decoder software, and 3) Send blocks through a multicast stream when signalized by the CVC manager.

CVC Manager (CVCM). CVCM can execute in the same server's unit, and uses a structure that keeps information on active clients in the system, such as the stream's last block, and last transmitted block. Based on these data, CVCM uses a state machine to categorize the clients and to support new requests according to three different options: (1) *Prefetching* requests, for which CVCM uses batching; (2) *Playback without discarding*, for which CVCM creates a full new multicast stream; (3) *Playback with discarding*, CVCM inserts the client into the group of receivers of the multicast stream already active, and orders a patch from the server to the client. Given these multiple options, GloVE supports two provider policies in the CVCM, which influence the resulting performance of the VoD system. First, the Emphasis on Chaining (PEC) policy, in which the CVCM always tries to create a new complete multicast stream from a client that is not a provider yet and has not discarded the first video blocks from its buffer. If CVCM cannot find such a client, it attempts to reuse an active multicast stream in combination with a patch. This strategy may lead to long chains of clients, where the majority of multicast streams serves single clients, although it is easier to recover from chain disruptions. However, it tends to cause a huge consumption of aggregated network bandwidth. Second, the Emphasis on Patching Policy (PEP), which gives priority to stream reuse, and requires less aggregated bandwidth, but as various clients share the same stream, it increases the complexity of recovering from broken chains, besides it uses a significant

amount of patches.

3.1. Operation Modes

Another interesting feature of the design of CVCM is the possibility to set different operation modes based on several combinations of stream reuse techniques available in the VoD prototype (Chaining, Patching, and Batching). So far, we have defined six operation modes, as follows.

- *Conventional*, which mimics a conventional VoD system without stream reuse.
- *Chaining*, which implements chaining as we proposed, creating client chains where all streams have only one receiving client.
- *Batching*, while a client prefetches for a new video for the first time new arriving clients that request the same video will form a receiving group to which the first client will transmit to.
- *Patching+Batching*, which implements a combination of patching and batching modes.
- *CVC*, which implements CVC whose operation is influenced by the provider policy it adopts either PEC or PEP.
- *CVC+Batching*, which implements the batching extension to the CVC mode.

In this work, we will present results only for the CVC+Batching mode under the emphasis on patching (PEP) policy, which produced the best performance according to our past experiments [19, 20].

4. Experimental Methodology

In this section, we present our testbed, the evaluation methodology, and experimental results we obtained.

4.1. Hardware Platform

The experiments we describe below were performed on a 6-node cluster of Intel Pentium IV 2.4 GHz, 1 GB RAM, running Rocks Linux 2.3.2 with kernel 2.4.18 using a Fast Ethernet switch with IP Multicast support (3Com Superstack II 3300, version 2.69). One node executed both the video server (CVCS) and the CVC manager (CVCM) whereas each of the other five nodes executed multiple instances of clients.

4.2. Evaluation Methodology

We used a MPEG-1 video (352x240, 29.97 frames/s) of Star Wars IV movie, with average transmission rate near to 1.45 Mbps. To allow multiple clients per node, we developed an emulator of MPEG-1 decoder, which uses a trace file containing the playback time of each segment of the video. The trace file was obtained by collecting the blocks' consumption pattern of the real decoder over a previous execution of the video. The workload we assumed ranges from medium to high clients' arrival rates according to a Poisson process with values of 6, 10, 20, 30, 60, 90, and 120 clients/min.

We assume that the CVCC video server supports at most 56 MPEG-1 streams simultaneously, according to previous measurements we made using the same network infrastructure. Also, we evaluated the playout buffer for sizes 64, 128, and 256 blocks, which can store video sequences of 22, 44, and 88 seconds, respectively.

We evaluate performance of a VoD system under practical network conditions for four operation modes: CVC+Batching, Chaining, Batching, and Patching+Batching. Specifically, instead of simulation [16], we employed an emulation approach by setting GloVE variables as follows: 1) 64 KB blocks as the system access unit; 2)prefetch limit of 32 blocks; 3) patch limit equal to half of playout buffer size minus 4 blocks of tolerance, and 4) discarding limit equal to playout buffer size minus prefetch limit minus 8 blocks of tolerance. The CVC manager introduces a tolerance of 4 blocks due to eventual information difference in the video contents that the playout buffers actually store and the metadata the manager maintains of playout buffers in the system.

The values of GloVE variables determine the time window where contents can be potentially reused. Given the average consumption rate of the video we used in the tests, the average block playtime is equal to 0.345 s. The duration of prefetch phase will be determined by the stream source, as follows. For coarse access to the server, the prefetch duration will depend on the number of active streams in the server, with a minimal value near to 350 ms. For smooth access to the server the prefetch duration is about 11 s. When a client is the streaming source, the prefetch duration will depend on the kind of reuse it is employed, ranging from few seconds with patching up to 22 seconds with batching. As an example, consider a playout buffer of 128 blocks and the patch limit of 64 blocks. In this way, streams the clients generate can be reused for patching within 20 s (block play time x patch limit) duration after their creation. Moreover, a client can provide a new complete stream up to 30 s (block play time x discard limit) after starting playback.

For the experiments with multiple videos, we emulated a collection of eight videos, where the video popularity follows a general Zipf distribution with $\alpha = 0.7$ [5]. In addition, we also investigated the sensitivity of the VoD system to $\alpha = 0$ which represents an uniform distribution, and $\alpha = 1$, which represents Zipf without skew.

4.3. Experimental Results

We present results of our experiments in three separate parts: single video, multiple videos, and a sensitivity analysis of server's performance to the popularity distribution of the videos.

Experiments with a Single Video

Channel utilization. Figure 2 shows the utilization of server's channels using block request rate with smooth and coarse access for different GloVE modes and playout buffer sizes. Note that channel occupancy indicates the relative degree of scalability of each particular technique. Specifically, the lower

channel occupancy a technique produces for a given amount of active clients the higher is its scalability.

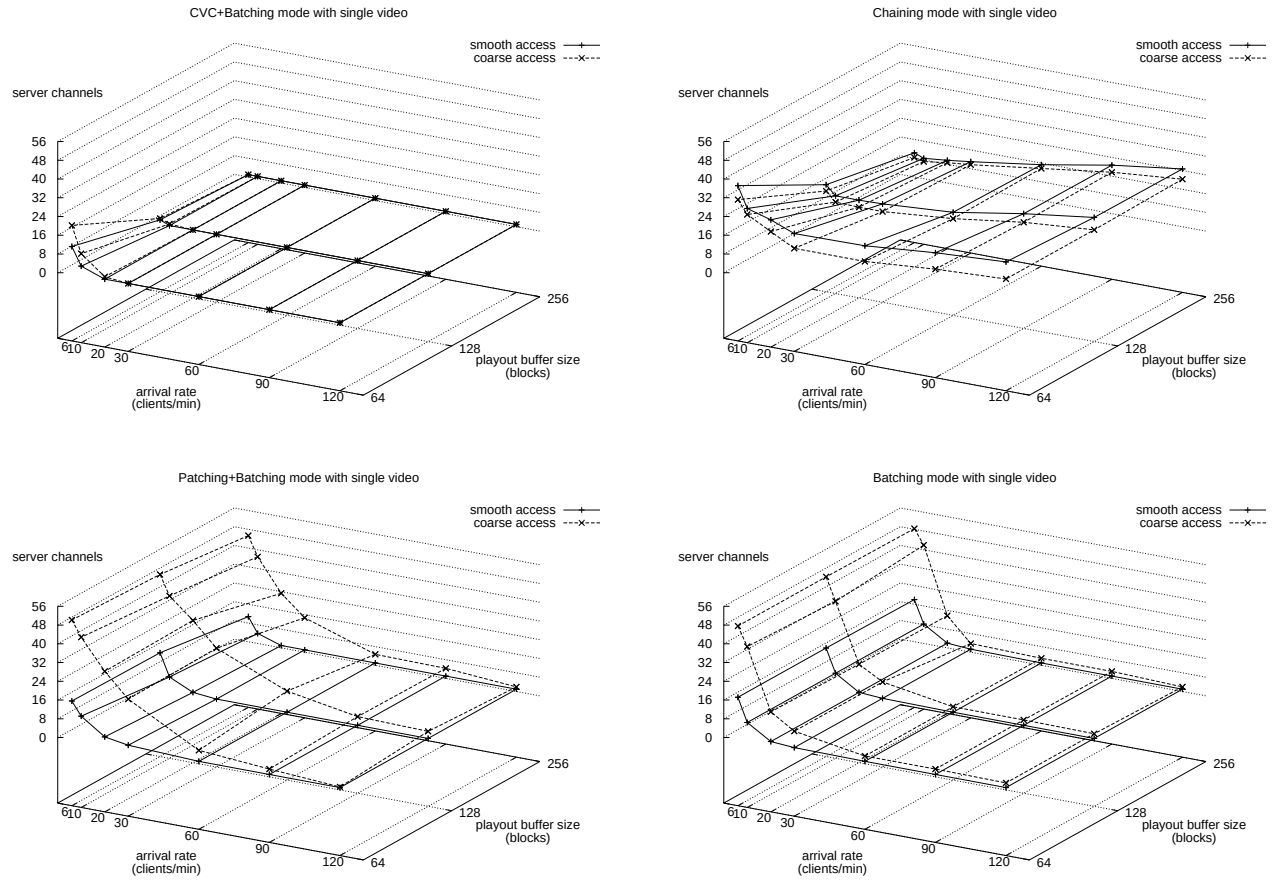

Figure 2. Utilization of server channels under each GloVE operation mode. (a) CVC+Batching. (b) Chaining. (c) Patching+Batching. (d) Batching.

Figure 2(a) shows that the CVC+Batching mode occupies only one channel for playout buffers with size larger than 64 blocks. For arrival rates lower than 20 clients/min, CVC+batching changed its behavior because the average interval between consecutive requests is at least 6 seconds. Recalling that 64-block playout buffers allow stream reuse after starting playback within intervals of at most 8 s for chaining and 10 s for Patching, the chance of clients missing the reuse window is high for arrival rates less than 20 clients/min, which reflects the low use of channels. As the reuse window depends on prefetching for modes that include Batching, the opportunity for stream reuse is higher for smooth accesses to the server. The reason is that smooth accesses generate long prefetchings, leading to lower

channel's occupation in comparison with coarse accesses, which issue short prefetchings.

Regarding buffer size, the Chaining mode (Figure 2(b)) also presents different behavior with 64-block playout buffers for low arrival rates. In particular, the reuse window is limited to Chaining within 8 s intervals, which restricted the minimum occupation to 8 channels. Another characteristic of chaining mode is the significant influence of the prefetching on arrival rates higher than 20 clients/min. In this case, the higher is the arrival rate, so is the probability of a client to arrive while previous ones are either prefetching or providing video streams. Note that as the prefetching phase is shorter with coarse accesses to the server, the channel occupancy presented slightly lower values.

As Figure 2(c) shows, the Patching+Batching mode is influenced strongly by the server access type. The fact is that this mode requires initial stream reuse using Batching, because a multicast stream must be active to allow reuse via Patching. Given a shorter stream prefetching using coarse access to the server, the use of channels tends to be very high for arrival rates lower than 60 clients/min. In contrast, with smooth access the opportunity for stream reuse with Batching is increased significantly, so that the minimum channel occupancy is reached at 20 clients/min. Note that the playout buffer size influences the chances of using Patching, especially when using larger buffers for low arrival rates, which led to better use of channels.

As can be seen in Figure 2(d), the server access type also determines the efficiency of the Batching mode. Specifically, when using the server with smooth access, channel's occupation is very similar to that achieved by the Patching+Batching mode. However, Batching mode does not benefit from larger buffers, as occurred with the Patching+Batching mode for low arrival rates. The difference of channel occupancy curves of the two models for arrival rates from 20 to 90 clients/min comes from the longer average prefetching the Batching mode generates. When Patching is used, the average prefetching is substantially reduced, decreasing the opportunity of stream reuse for the next arriving clients.

Active streams. Figure 3 shows the amount of active streams in the system for each server mode. The

values in the figure indicate the aggregated bandwidth as required by each mode. Note that the minimum amount of active streams is at least equal two, which includes one stream from both the server and the first client.

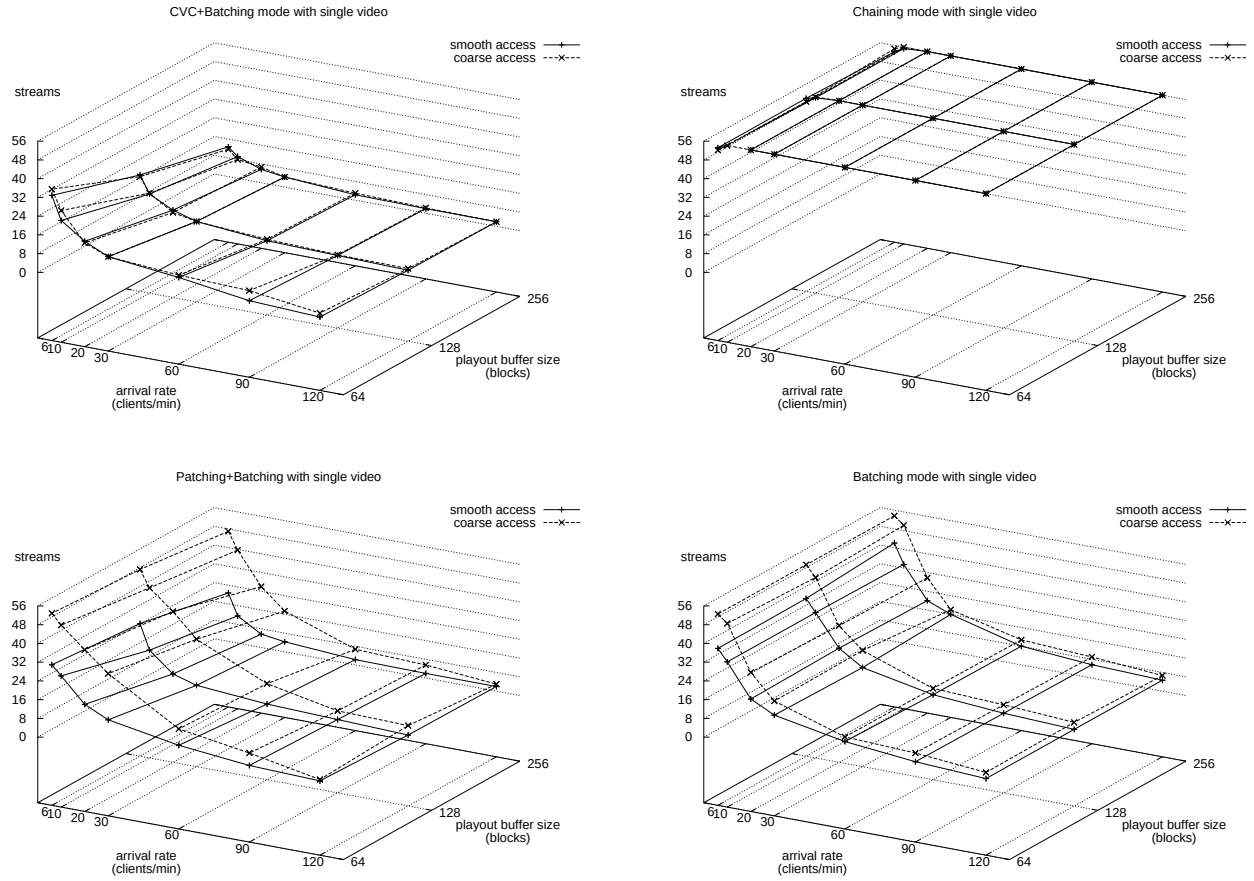

Figure 3. Number of active streams under each GloVE operation mode. (a) CVC+Batching. (b) Chaining. (c) Patching+Batching. (d) Batching.

For the CVC+Batching mode, as shown in Figure 3(a), the number of active streams is practically the same whether using smooth or coarse accesses to the server. The size of a playout buffer only influences the amount of active streams for low arrival rates. Another factor that affects the number of active streams is the type of stream reuse. Whenever Patching can be used no full streams will be generated. Otherwise, the server attempts to use Chaining, and generates a new stream if possible. The latter alternative occurs often for low arrival rates and playout buffers with small size.

Intuitively, the Chaining mode (Figure 3(b)) led to the same amount of active streams and clients, since the streams Chaining generates have only one receiver and the Chaining mode is independent of buffer size and server's access type.

Figure 3(c) presents curves of active streams for the Patching+Batching mode. As the figure shows, this mode is influenced by both playout buffer size and server access type, especially for arrival rates lower than 60 clients/min. For smooth access, the server has more opportunities for stream reuse through Patching, which led to less active streams. For coarse access, as the arrival rate decreases, the probability of stream reuse using Batching increases with just one receiver per stream, and also with many more streams coming from the server because no stream reuse is possible.

The number of active streams for the Batching mode as shown in Figure 3(d) depends on both arrival rate and server access type. For arrival rates higher than 30 clients/min the influence of server access type is negligible, with many clients sharing the same stream. For arrival rates lower than 30 clients/min, the majority of streams for a server with coarse access comes from the server itself, because there is no opportunity to use Batching. When using a server with smooth access, the streams Batching generates are the dominant ones.

Startup latency. The latency between a video request and its playback is shown in Figure 4. Intuitively, an efficient yet fair VoD system must have short startup latency with minimum time variance.

Figure 4(a) presents the behavior of startup latency of a server with smooth accesses. The average latency for the CVC+Batching mode is near to 10 s, which is slightly less than the latency clients experience (11 s) when they receive streams from the server at playback rate. The minimum latency value is near to 5 s, which occurs when Patching is used and the patch stream corresponds to the half of the prefetch limit (16 blocks x block play time = 5.5 s). The maximum value is near to 18 s, which is achieved when a client is inserted into the group of a prefetching client (maximum latency = 2 x prefetch limit = 22 s). For the Chaining mode, the average latency is near to 11 s, and the minimum

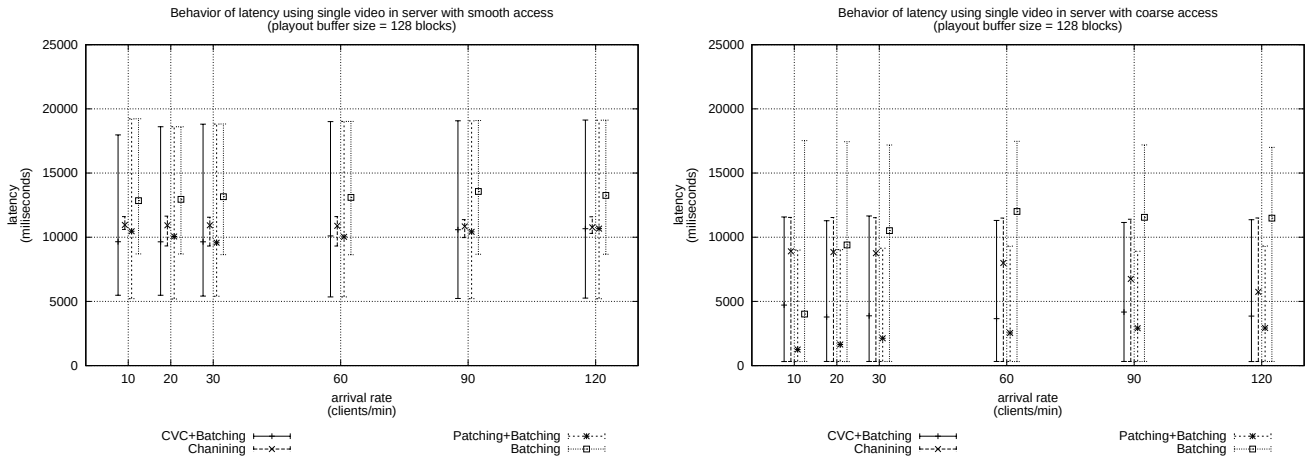

Figure 4. Error bars showing playback start latency (max,avg,min) under different modes using playout buffers of 128 blocks. (a) With smooth prefetching. (b) With coarse prefetching.

and maximum values are very close to the average latency. This occurs because in Chaining all streams follow the playback rate and no waiting time happens between video request and starting prefetching. The Patching+Batching mode has similar behavior to that of CVC+Batching. In particular, it led to a slightly decrease on the average latency for arrival rates ranging from 20 to 60 clients/min, because the use of Patching is increased within this range. For the Batching mode, the average latency is near to 13 s because a lot of clients will be inserted into the group of a client that performs prefetching, which also increases the minimum startup latency to about 8 seconds.

Figure 4(b) presents startup latency for a server with coarse accesses. Notice that all modes have minimum latency near to 350 ms, which is experienced by the first client that received a stream from server. For the CVC+Batching mode, the average latency is close to 4 s, with maximum value less than 12 s, which happened to a client that was inserted in the group of a prefetching client (350 ms + 11 s). For the Chaining mode, the average latency is approximately 9 s for arrival rates from 10 to 30 clients/min. However, for higher arrival rates, the average latency starts to decrease due to the prefetching effect, which required additional server channels. The maximum value is similar to that of CVC+Batching. The Patching+Batching mode has an average latency that is dependent on the arrival rate. As the arrival rate increases, so does the average latency because more clients are served through Patching instead of

the VoD server. The maximum Latency value is near to 9 s, and follows the Patching behavior. For the Batching mode, the average latency increases according to the arrival rate until 60 clients/min, at which the average latency is 12 s, while maintaining this value for higher arrival rates. Batching produced the maximum average startup latency for this server's access type, near to 17 s, as the result of a client being inserted into the multicast group of a prefetching client.

Experiments with Multiple Videos

Channel utilization. Figure 5 presents the utilization of server's channels as required by the different GloVE modes with eight videos stored in the VoD server. The videos' popularity follows a Zipf distribution with $\alpha = 0.7$. In the best case, at least one channel will be used for each different video the clients requested. In the worst case, one channel will be used for each client request, which is ultimately the behavior of a conventional VoD system.

The curves of Figure 5(a) show that the minimum amount of channels the server used was reached with at least 20 clients/min and 256-block playout buffers. The curve for 128-block playout buffers indicates that the minimum amount occurred at 60 clients/min. Recall that for a single video the minimum was reached with near to 10 clients/min. The same relationship remained for 64-block playout buffers, which demonstrates the direct influence of the number of videos on channel occupancy. The negative impact of the server access type on channel occupancy was significant for average playout buffers and average arrival rates. This result contrasts with the case of a single video where the negative impact was restricted to playout buffers with the lowest size and small arrival rates.

Similarly to the single video case, the Chaining mode (refer to Figure 5(b)) continued to suffer the negative consequence of prefetching, specially for arrival rates higher than 30 clients/min. At this arrival rate, we noticed the minimum channel occupancy with 16 channels. For the same reason, as in the single video case, Chaining achieved better performance using the server with coarse access.

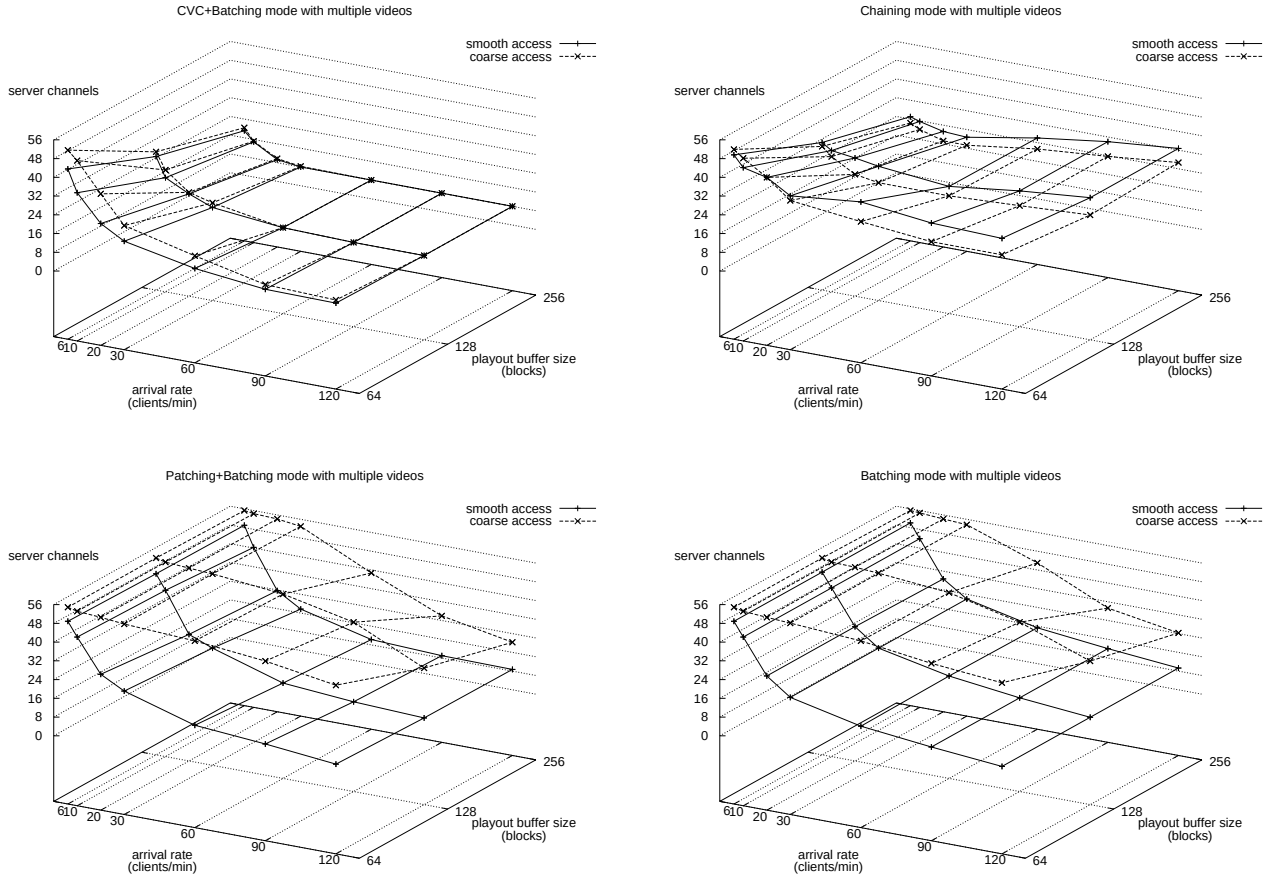

Figure 5. Utilization of server channels under each GloVE operation mode. (a) CVC+Batching. (b) Chaining. (c) Patching+Batching. (d) Batching.

For Patching+Batching mode, the observations we pointed out for single video hold also for multiple videos, as can be seen in Figure 5(c). The main difference is that there exist considerably less opportunities to reuse streams with either Batching or Patching. For arrival rates ranging from 60 to 120 clients/min, the minimum channel occupation is reached with smooth accesses to the server. Note that with coarse access to the server, 20 channels is the minimum occupancy value that is achieved for the highest arrival rate.

In Figure 5(d), channel's occupancy behavior is very similar to that of Patching+Batching mode previously analyzed. However, it led to slightly higher channel occupancy for arrival rates ranging from 10 to 90 clients/min.

Active streams. The amount of active streams appears in Figure 6. The minimum number of active streams tends to be 16 provided that all the videos are requested twice at least.

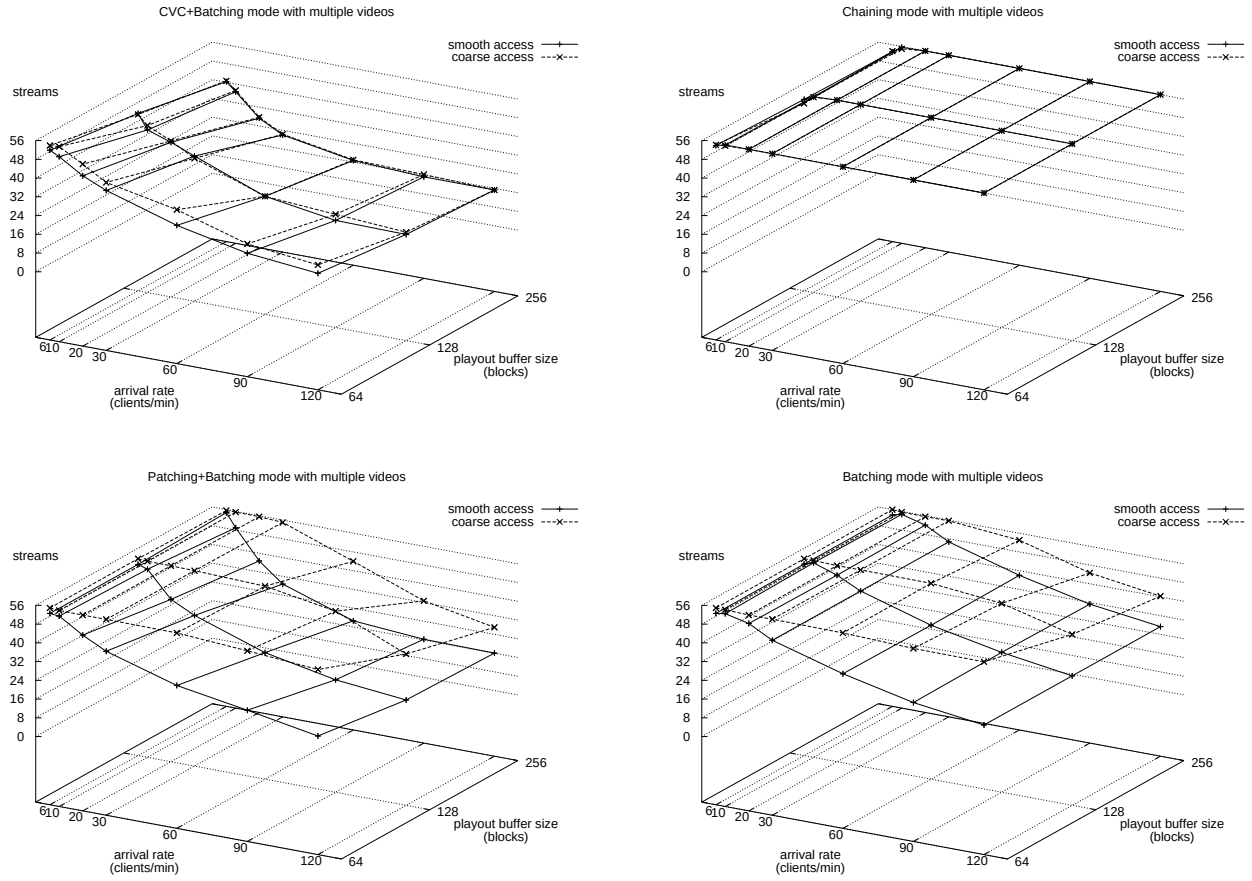

Figure 6. Number of active streams under each GloVE operation mode. (a) CVC+Batching. (b) Chaining. (c) Patching+Batching. (d) Batching.

In the CVC+Batching mode (Figure 6(a)), the curves remained similar for both coarse and smooth accesses to the server. The size of the playout buffer affected arrival rates from low to medium. As explained in the case of a single video, the emphasis on Patching led to the lowest amount of active streams. Naturally, Chaining tends to be used more within shorter arrival rates and with smaller playout buffers.

In Chaining mode (Figure 6(b)) the number of active streams is equal to the number of active clients,

as in the case of single video.

Comparing the two modes as shown in Figures 6(c) and (d), it becomes clear that both modes are similar. The main differences appear mainly for playout buffer sizes of 128 and 256 blocks. With either playout buffer size, the Patching+Batching mode generates less streams due to the opportunity of stream reuse through Patching, which is not available in the Batching-only mode. Also, while the former achieved a minimum count of 16 streams , the latter created as many as 26 streams at least.

Startup latency. The startup latency between a client request and starting video playback is shown in Figure 7.

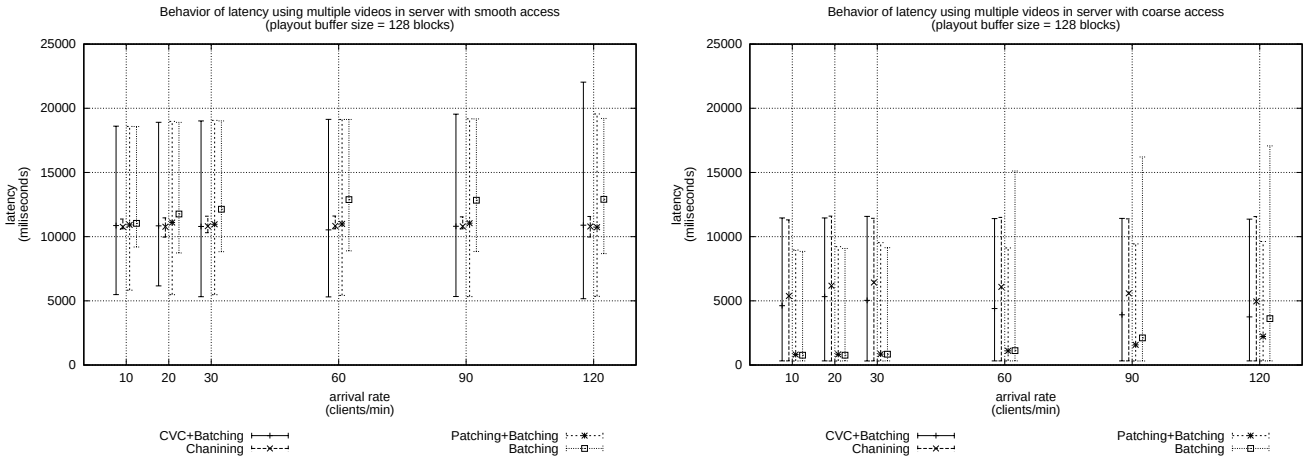

Figure 7. Error bars showing playback start latency (max,avg,min) under different modes using playout buffer of 128 blocks. (a) With smooth prefetching. (b) With coarse prefetching.

Figure 7(a) shows the startup latency when using a server with smooth access. Again, all values the Figure presents are very similar to those of a single video, so the same observations remained valid.

Figure 7(b) presents the startup latency for a server with coarse accesses. In contrast with smooth access, the average values for this access mode revealed interesting modifications to the startup latency, especially for Patching+Batching and Batching modes. As the number of clients receiving streams from the server raised significantly, the average startup latency decreased to values near to 1 for arrival rates lower than 60 clients/min. For higher arrival rates, the average startup latency started to increase.

Figure 8 illustrates the impact of the distribution of video's popularity on the occupancy of server's channels.

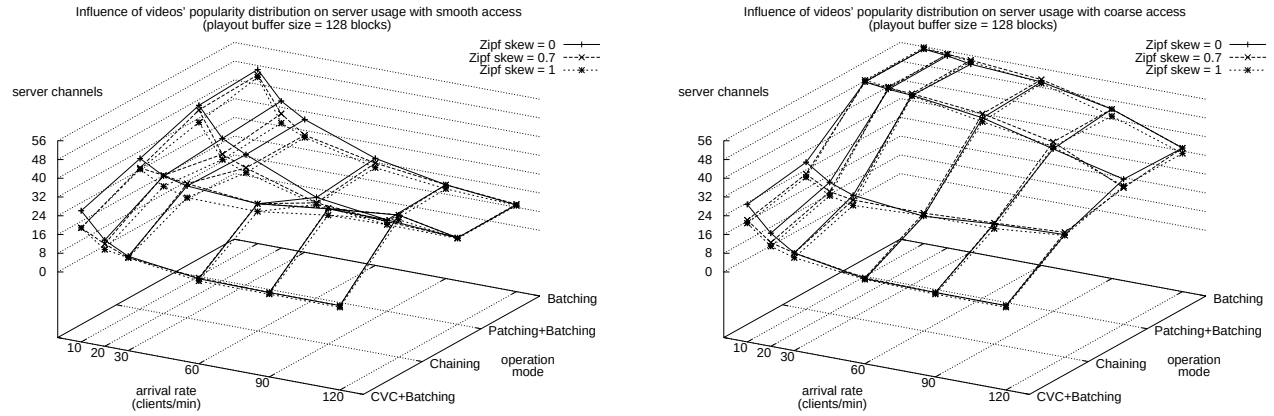

Figure 8. Utilization of server channels under different values of Zipf skew. (a) smooth access. (b) coarse access.

As can be seen in Figures 8(a) and (b), the variations on the distribution of popularity of videos did not impact significantly the amount of channels the VoD system required to service the client requests. Indeed, none of the operation modes presented variations higher than 20% on the occupancy of server's channels .

4.4. Discussion

In the following, we discuss results presented in the previous section. We assume the average values we obtained with playout buffers of 128 blocks for arrival rates of 6, 10, 20, 30, 60, 90, and 120 clients/min[2].

Table 1 presents the results for a server with single video. These values show that the CVC+Batching mode achieved the best results for all the performance metrics we evaluated.

[2]Except for latency and popularity distribution, which did not consider the value of 6 clients/min.

Operation Mode	Server Channels		Amount of Streams		Latency (ms)	
	smooth	coarse	smooth	coarse	smooth	coarse
CVC+Batching	1.2	1.4	8.7	8.7	10048	4010
Chaining	17.3	14.0	55.7	55.5	10886	7832
Patching+Batching	4.1	23.6	10.4	26.5	10204	2234
Batching	4.7	17.1	18.2	26.1	13146	9828

Table 1. Average results measured for single video and playout buffer size of 128 blocks.

Table 2 summarizes the average results for multiple videos. Again, the CVC+Batching mode outperformed the other modes for all the performance metrics we tested.

Operation Mode	Server Channels		Amount of Streams		Latency (ms)	
	smooth	coarse	smooth	coarse	smooth	coarse
CVC+Batching	14.5	15.7	31.2	31.9	10789	4507
Chaining	30.9	25.8	55.6	55.3	10797	5769
Patching+Batching	24.6	49.2	35.2	51.9	10960	1232
Batching	26.0	49.8	43.4	53.8	12260	1538

Table 2. Average results measured for multiple videos and playout buffer size of 128 blocks.

Finally, Table 3 shows average channel occupancy for several videos' distribution of popularity. The results in the table demonstrate clearly the low impact of the α value on channel occupancy. Also, they give evidence that the system should perform well in widely different environments.

Operation Mode	Zipf with $\alpha = 0$		Zipf with $\alpha = 0.7$		Zipf with $\alpha = 1$	
	smooth	coarse	smooth	coarse	smooth	coarse
CVC+Batching	13.6	14.8	12.1	13.1	11.3	11.9
Chaining	30.7	25.9	30.3	24.6	27.0	22.9
Patching+Batching	23.4	47.7	20.5	48.1	18.5	47.3
Batching	24.7	48.2	22.1	48.8	20.3	47.6

Table 3. Average channel occupancy for different Zipf skews and playout buffer size of 128 blocks.

5. Related Work

Works focusing on reactive stream reuse techniques. The original works that introduced the techniques studied in this work are Batching [5], Chaining, Patching, and CVC [13]. GloVE uses the Batching technique in a different way as proposed in its original form [5]. Specifically, in GloVE clients instead of the server can apply the Batching technique [3]. Regarding to Chaining, the GloVE implements the standard approach described in the original work we described in previous section. GloVE implements the original Patching technique as previously explained. Transition Patching was proposed in [4] that allows patches for patch streams, which GloVE does not support. The work in [2] presents a comparison of stream merging algorithms, but does not report results of any practical implementation.

Works focusing on peer-to-peer systems. The explosive growth of peer-to-peer systems started with the development of file-sharing software, noticeably Napster [17] and Gnutella [7]. Napster employed a centralized server to keep the metadata of contents stored at the peers across the Internet. Gnutella introduced the distributed metadata approach, which obviated the need for a central management. GloVE is similar to Napster in the sense that it uses CVCM, which is a central metadata manager. In the context of peer-to-peer video streaming, various works have been developed in recent years, which are closely related to GloVE: CoopNet [18], P2Cast [8], ZigZag [23], GnuStream [14], DirectStream [9], and PROMISE [10] are valuable academic examples whereas Allcast [1], vTrails [24], and C-span [3] are examples of commercial software. The idea behind CoopNet is to use clients as stream providers of contents they received from the server previously, but only over periods of server overload (called "flash crowd"). P2Cast uses clients to create application level multicast trees for streaming, where these clients can send patches to other clients to recover from video sequences they missed. ZigZag also proposes an application-level multicast tree scheme, but its main goal is to support small end-to-end delay. The current version of GloVE uses IP Multicast facility as provided by typical LANs, but can be extended

[3] Refer to [6] for a classification of Batching Policies.

to work with application-level multicast too. GnuStream introduced the novelty of working over the widespread Gnutella network, which offers already a huge amount of continuous media. DirectStream is an on-demand video streaming service that uses a directory server to administer active clients in a manner very similar to that of GloVE. PROMISE offers support for client heterogeneity, reliability, and limited capacity, which make the system suitable for the Internet as shown in the reported experiments. This is not the case of our current version of GloVE. Given the limited information about commercial examples, no direct relationship can be made with GloVE. In general, such works report simulated performance of their proposals, in contrast with our work that evaluated a prototype using real network traffic.

6. Conclusions and Ongoing Work

In this work we have compared the efficiency of reactive stream reuse techniques in a practical peer-to-peer VoD system called the Global Video Environment (GloVE) we implemented in the Parallel Computing Laboratory at COPPE/UFRJ. In particular, we described the design and evaluation of different operation modes of GloVE according to stream reuse techniques it uses, namely Batching, Patching, Chaining, and CVC. Also, we measured the influence on system behavior of client's access pattern to the server. Finally, we analyzed the impact of video popularity distribution on the system behavior.

Overall, the CVC+Batching mode outperformed the other modes on all the metrics we assessed, both for a single video and for multiple videos stored in the server. Also, CVC+Batching is not significantly affected by the server's access pattern, which gives evidence that such a combining technique works with different servers. Furthermore, we showed that skews on videos' popularity distribution did not impact substantially the VoD system's performance. Thus, we speculate that the VoD system will have good performance for different audiences.

Currently, we are working on mechanisms that will support scalable peer-to-peer VoD systems for mobile environments with heterogeneous devices. Also, we plan to improve GloVE as to dynamically

self-adapt to variations on network and peer conditions [21].

7. Acknowledgments

The authors wish to thank Edison Ishikawa for his helpful comments on all the phases of this work, Leonardo Bragato for the design and implementation of the CVCS video server, and the Brazilian Agencies Capes, Cnpq and Finep for their financial support.

References

[1] AllCast. http://www.allcast.com.

[2] A. Bar-Noy, J. Goshi, R. E. Ladner, and K. Tam. Comparison of stream merging algorithms for media-on-demand. In *Proceedings of the SPIE Multimedia Computing and Networking (MMCN)*, San Jose, CA, January 2002.

[3] C-star. http://www.centerspan.com.

[4] Y. Cai and K. A. Hua. An Effcient Bandwidth-Sharing Technique for True Video on Demand Systems. In *Proceedings of the ACM Multimedia*, pages 211–214, 1999.

[5] A. Dan, D. Sitaram, and P. Shahabuddin. Dynamic Batching Policies for an On-Demand Video Server. *Multimedia Systems*, 4(3):112–121, 1996.

[6] D. Ghose and H. J. Kim. Scheduling video streams in video-on-demand systems: A survey. *Multimedia Tools and Applications*, 11:167–195, 2000.

[7] Gnutella. http://www.gnutella.com.

[8] Y. Guo, K. Suh, J. Kurose, and D. Towsley. P2cast: Peer-to-peer patching scheme for vod service. In *Proceedings of the 12th Word Wide Web Conference (WWW)*, pages 301–309, Budapest, Hungary, May 2002.

[9] Y. Guo, K. Suh, J. Kurose, and D. Towsley. A peer-to-peer on-demand streaming service and its performance evaluation. In *Proceedings of the International Conference on Multimedia and Expo (ICME)*, volume 2, pages 649–652, July 2003.

[10] M. Hedeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. Promise: Peer-to-peer media streaming using collectcast. In *To Appear in the Proceedings of the ACM Multimedia*, Berkeley, CA, November 2003.

[11] A. Hu. Video-on-Demand Broadcasting Protocolos: A Comprehensive Study. In *Proceedings of the IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.

[12] K. A. Hua, Y. Cai, and S. Sheu. Patching: A Multicast Technique for True Video-on-Demand Services. In *Proceedings of the ACM Multimedia*, pages 191–200, 1998.

[13] E. Ishikawa and C. Amorim. Cooperative Video Caching for Interactive and Scalable VoD Systems. In *Proceedings of the First International Conference on Networking, Part 2*, Lecture Notes in Computer Science 2094, pages 776–785, 2001.

[14] D. Jiang, Y. Dong, D. Xu, and B. Bhargava. Gnustream: a p2p media streaming system prototype. In *Proceedings of the International Conference on Multimedia and Expo (ICME)*, volume 2, pages 325–328, July 2003.

[15] H. Ma and K. G. Shin. Multicast video-on-demand services. *ACM Computer Communication Review*, 32(1), January 2002.

[16] I. MacGregor. The Relationship Between Simulation and Emulation. In *Proceedings of the Winter Simulation Conference (WSC'02)*, pages 1683–1688, San Diego, CA, December 2002.

[17] Napster. http://www.napster.com.

[18] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proceedings of the NOSSDAV*, pages 177–186, Miami, Florida, May 2002.

[19] L. B. Pinho, E. Ishikawa, and C. L. Amorim. GloVE: A Distributed Environment for Low Cost Scalable VoD Systems. In *Proceedings of the 14th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 117–124, Vitoria, ES, Brazil, October 2002.

[20] L. B. Pinho, E. Ishikawa, and C. L. Amorim. GloVE: A Distributed Environment for Scalable Video-on-Demand Systems. *International Journal of High Performance Computing Applications (IJHPCA)*, 17(2):147–161, Summer 2003.

[21] M. K. Ramanathan, V. Kalogeraki, and J. Pruyne. Finding good peers in peer-to-peer networks. In *Proceedings of the International Parallel and Distributed Processing Symphosium (IPDPS)*, pages 24–31, Fort Lauderdale, Florida, April 2002.

[22] S. Sheu, K. A. Hua, and W. Tavanapong. Chaining: A Generalized Batching Technique for Video-On-Demand. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 110–117, 1997.

[23] D. Tran, K. Hua, and T. Do. Zigzag: An efficient peer-to-peer scheme for media streaming. In *Proceedings of the IEEE INFOCOM*, pages 1283–1292, San Francisco, CA, March-April 2003.

[24] vTrails. http://www.vtrails.com.