Collapsed Cooperative Video Cache for Content Distribution Networks

Edison Ishikawa[1,2], Cláudio Luis Amorim[2]

[1] Departamento de Engenharia de Sistemas e Computação - Instituto Militar de Engenharia (IME) – Praça General Tibúrcio, 80 – 22.290-270 - Rio de Janeiro – RJ - Brazil

[2] Programa de Engenharia de Sistemas – COPPE - Universidade Federal do Rio de Janeiro (UFRJ) - Rio de Janeiro – RJ - Brazil

ishikawa@ime.eb.br, amorim@cos.ufrj.br

Abstract. *In this work we describe the Collapsed Cooperative Video Cache (C-CVC), a novel scalable technique for delivering video streams over content distribution networks (CDNs). C-CVC is based on the idea of collapsing client buffers into the proxy caches on the CDN edges nearest to the clients while enforcing transparently cooperation among clients that share the same video content. As a result of storing videos close to the clients, C-CVC decreases the average initial video exhibition latency. C-CVC also reserves cache space in the proxy, therefore absorbing more jitter and using less CDN resources. Most importantly, by implementing the collapsed cooperative video cache globally across CDN's proxies, C-CVC enables VoD systems to become highly scalable.*

1. Introduction

The growth in popularity of video media can be explained by the dissemination of broadband accesses. However the available communication bandwidth in Video on Demand (VoD) servers and network backbones limit severely the distribution of video contents in an isochronous way. The problem is that typical VoD systems often reserve resources along the communication path whose limited bandwidth ultimately restricts the target audience. Recently, the development of proxy-based VoD systems, in which video proxies are deployed in the critical path between a VoD server and its clients, have emerged as a necessary requisite to support video stream delivery in a scalable way. Also, several authors proposed new scalable approaches that allow the reuse of video streams through adaptations of multicast/broadcast techniques such as Batching [Aggarwal, Wolf and Yu 1996], Piggybacking [Golubick, Sitaram and Shahabuddin 1996], Chaining [Sheu, Hua and Tavanapong 1997], Patching [Hua, Drops and Sheu 1998], Stream Tapping [Carter and Long 1997], Catching [Gao, Zhang and Towsley 1999], and Skimming [Eager, Vernon and Zahorjan 2000] or some combination of them [Acharya and Smith 2000], [Bommaiah, Guo, Hofmann and Paul 2000], [Verscheure, Venkatramini, Frossard and Amini 2002], [Almeida, Eager, Ferris and Vernon 2002], [Wang, Sen, Adler and Towsley 2002].

For instance, the work in [Verscheure, Venkatramini, Frossard and Amini 2002] describes a cache strategy that accounts for the establishment of Quality of Service (QoS), but the proxy does not reserve memory for its clients. Another scalable technique as proposed in [Almeida, Eager, Ferris and Vernon 2002] called proxy

skimming requires a lot of memory for implementing the client buffer, thus increasing the client equipment costs. To the best of our knowledge, there has been no proposal that reserves resources in the proxy placed on the edge of a VoD system in order to provide guarantee of minimum cache hit to exhibit videos without hiccups.

Our solution is motivated from the observation that typical client equipment of a VoD system treats the system jitter by using locally a video buffer that compensates for delays or advances in the media delivery. In addition, we note that a large enough client buffer can support large variation of QoS in the media delivery, while providing substantial bandwidth savings, which can be used to transmit even more streams simultaneously in a scalable way. However, a large client buffer means more expensive set-top boxes and increasing service latency. Bearing this in mind we propose the collapsed cooperative video cache (C-CVC) technique that as we will show allows client buffers to support the design of scalable proxy-based VoD servers in a cost-effective way.

The cooperative video cache (CVC) as described originally in [Ishikawa and Amorim 2001] follows a peer-to-peer architecture in which clients at the borders of the VoD system not only consume video stream contents they request, but also can cooperate with the system by acting as video stream providers to other clients in the system. In contrast, the collapsed CVC version enables the CVC concept to efficiently work also in more practical settings, as Figure 1 illustrates. More specifically, C-CVC is suited to work on the edge of a Content Distribution Network (CDN).

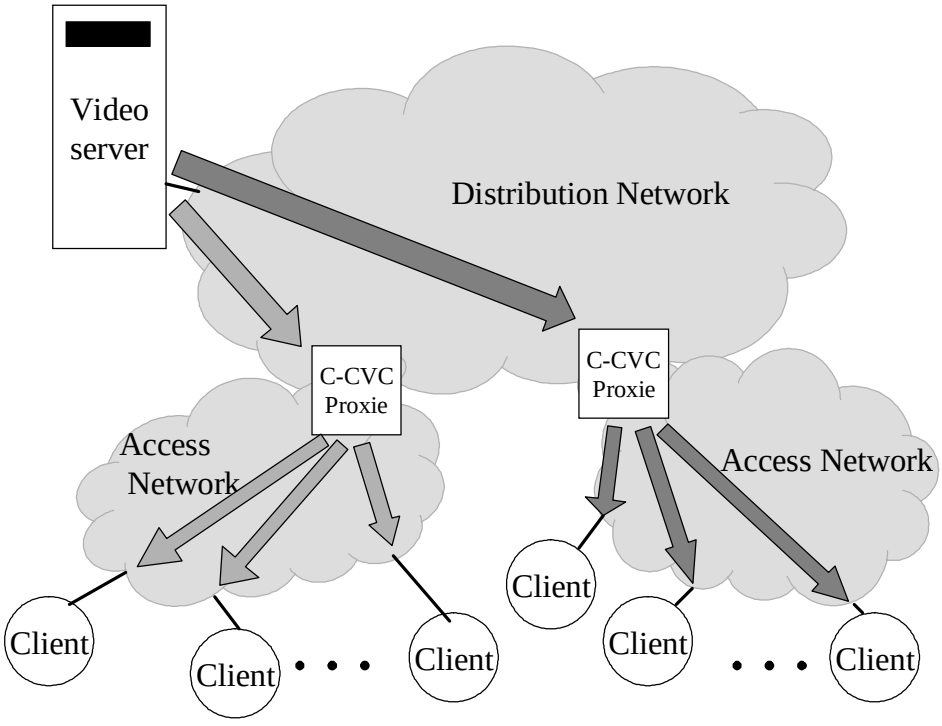

Figure 1. The components of distribution and access networks

The main contributions of this paper are:

- We developed a new scalable technique for video stream delivery called collapsed CVC that benefits from the original CVC technique while expanding its applicability.
- Using detailed simulations we show that the collapsed CVC technique enables scalable VoD systems to be built for delivering popular videos during prime time, in a cost-effective manner, even using low-cost set-top-boxes.
- We show that the resulting C-CVC-based VoD system also works for less popular videos by self-adjusting dynamically according to video access pattern.

The remainder of this paper is organized as follows. Section 2 overviews the CVC technique. Section 3 describes the collapsed version of CVC. Section 4 presents our experimental methodology and simulated results of a VoD server based on C-CVC. Finally, in Section 5 we conclude and outline future works.

2. Overview of the Cooperative Video Cache

The cooperative video cache (CVC) [Ishikawa and Amorim 2001] treats client buffers as part of a single cooperative memory, as shown in Figure 2. Like the client-server model, each CVC client uses its own video buffer for two main purposes: 1) handling the system jitter and 2) hold the incoming video stream for decoding and exhibition. Differently from the client-server model, each CVC client can cooperate with the VoD system by sharing the content of its local buffer with other active clients in the system.

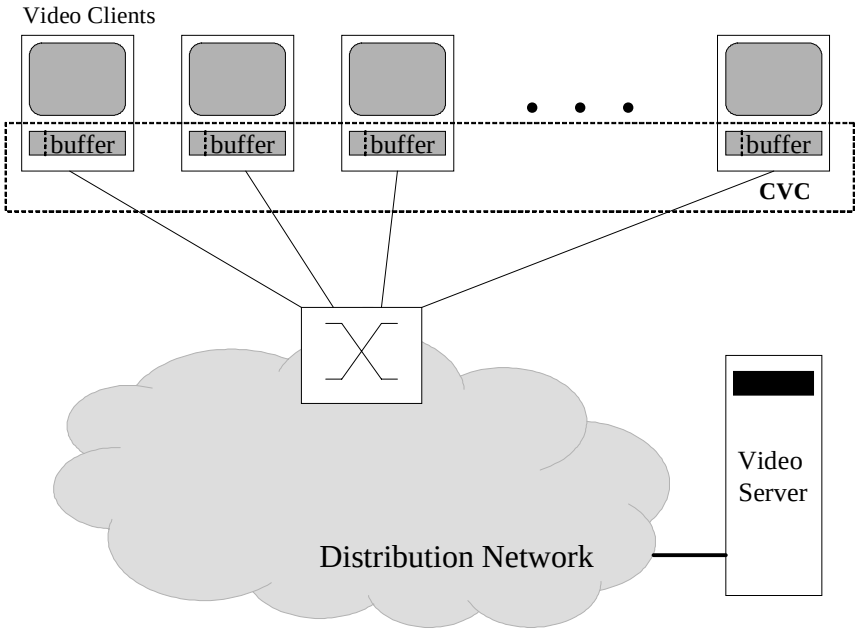

Figure 2. CVC treats the client buffers as a single global memory in a cooperative way.

Under conventional operation the incoming video stream constantly renews the content of the client's video buffer. We assume that the video unit is the MPEG Group of Frames (GoF). Also, arriving GoFs at the client are stored in the local buffer until

they are displayed, after which they are usually discarded. In contrast, instead of discarding a displayed GoF, CVC may reuse it, e.g., by joining it to a new video stream that is shared with another client that requested the same video. CVC applies such a stream reuse technique to new clients that log into the system provided they require any available video being exhibited. Ultimately, stream reuse can result in video streams that span long chains aka chaining [Sheu, Hua and Tavanapong 1997]. In addition, CVC uses buffer's contents for sending patches to complete other video streams in the system so that continuous chain of video streams will be formed when even necessary. CVC's patching differs from the original patching [Hua, Drops and Sheu 1998] by allowing clients also to send patches.

An implementation of CVC using a 100 Mbps Fast-ethernet switch [Pinho, Ishikawa and Amorim 2002] showed that CVC allows conventional VoD servers to reduce as much as 90% the demand on its logical video channels. CVC can improve system scalability using low-cost VoD servers and it is ideal to distribute institutional videos for training and other informational purposes. More detail of the CVC technique can be found in [Ishikawa and Amorim 2001], [Pinho, Ishikawa and Amorim 2002] and [Pinho, Ishikawa and Amorim 2003]. Nevertheless, the basic CVC presents some shortcomings:

- Increasing the traffic between clients at the network´s opposite sides may congest the network backbone, preventing the use of CVC within WAN environments.
- CVC is not suitable to asymmetrical links that prevail in the *last mile* to the home user;
- The reliability of a CVC-based VoD system is low due to client silent faults;
- CVC may restrict user's privacy.

Next, we examine how the collapsed CVC eliminates these restrictions, allowing to efficiently extending the concept of CVC to WANs.

3. The Collapsed cooperative video cache

To eliminate CVC´s drawbacks we propose to collapse CVC client buffers into the network access points. For simplicity, this paper focuses on the design, operation, and evaluation of a single proxy design based on the C-CVC technique. We plan to evaluate C-CVC across multiple CDN proxies in the near future.

Figure 3 shows client buffers collapsed in the proxy at the edge of a content distribution network, before the so called *last mile* to the client.

The main difference between the collapsed CVC and other proxy schemes is that C-CVC reserves proxy memory space for client's usage in a way that to every client buffer there exists one extra block associate to it in the proxy memory. Therefore, C-CVC guarantees cache hits for all the collapsed buffers in the proxy, or they are likely to have a minimum guarantee with high probability. In fact, a substantial part of the client buffer is collapsed in the proxy, while the remainder part that is left in the client is used to decode the video and to handle the proxy jitter due to the *last mile*. The collapsed buffer, whose content can be shared cooperatively with other users, absorbs the remaining system jitter. The collapsed buffer is filled accordingly to the contracted

QoS with both the VoD server and the distribution network. So, unless the server or the distribution network cannot sustain the contracted QoS, the cache hit in the collapsed buffer is guaranteed by the contracted QoS. Also, if a user stops making use of the system, the user's reserved proxy memory can be used alternatively to overall benefit of the system. It is important to notice that cache hits in the proxy depend on which part of the video is requested. If the part is in the collapsed buffer, the probability of cache hit of the remainder part of the video is high and is guaranteed by the contracted QoS, otherwise the cache hit has no QoS guarantee. For instance, a cache miss occurs on the first time the video is ordered, and the user will experience the initial exhibition latency while the collapsed buffer is filled.

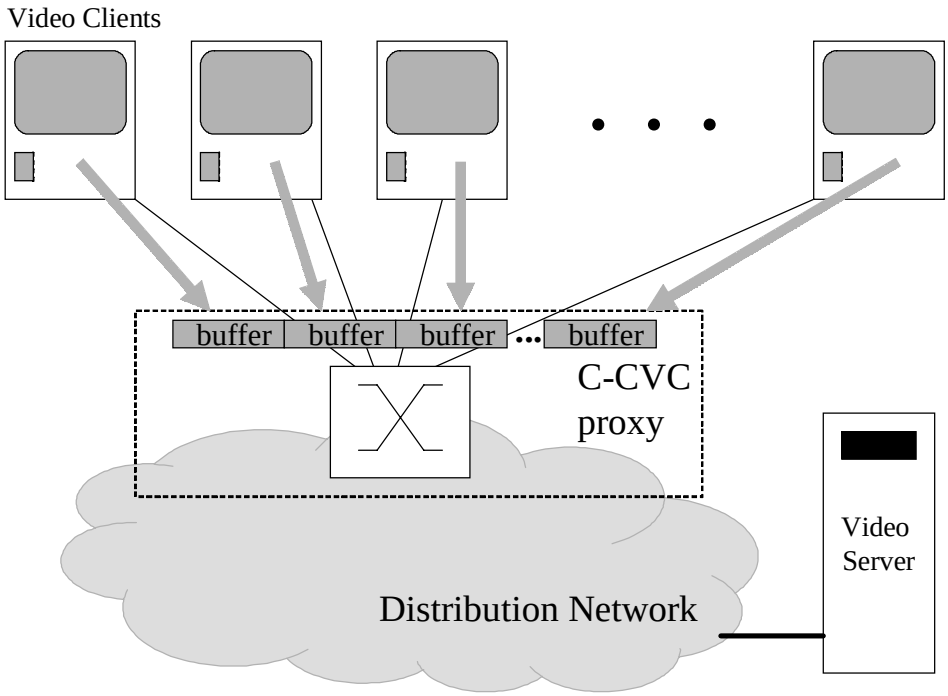

Figure 3. Collapsing of CVC buffers in a central access point.

3.1 C-CVC design

The collapsed buffer can be divided into four segments, as shown in Figure 4. The first segment lies between the beginning pointer **B** and the overflow pointer **O**. The second segment lies between **O** and the underflow pointer **U**. The pointer **W** indicates the average position in the buffer to write next incoming GoF units, and it should point into the second segment. The third segment is located between **U** and the reading pointer **R**. The fourth segment lies between **R** and end-of-buffer pointer **E**. The content's segment that cannot be discarded and has not been exhibited, or is to be exhibited shortly, is held in the second and third segments of the buffer.

Each slot of the collapsed buffer has enough memory to store a small number of video units, each of which corresponds to an average of few seconds of video stream.

In this way, the length of time of a video can be divided into time slots forming a circular vector that rotates clockwise over the video vector. Figure 5 illustrates the two vectors. The video vector represents the time units that compose the video and has also the relevant information that is used by C-CVC for managing the units.

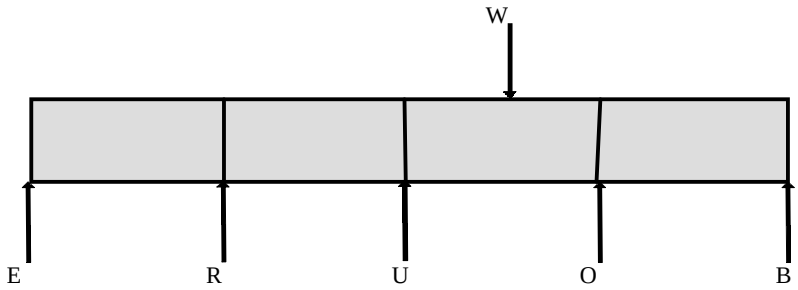

Figure 4. The division of the collapsed buffer.

Associate to each video there is a double circular vector used by the C-CVC system manager. The video vector has the GoF timestamps and uses the GoF's sequential number to map every GoFs of the video. Also if a GoF is in the cache then the video vector has a pointer to it, otherwise the pointer is null. By using such a data structure, the video unit can be accessed/controlled by either the GoF sequential number or by its timestamp. When a client starts a video session, some slots of the time-slot vector (S) are reserved to form the collapsed buffer for the client's equipment. As the example in Figure 6 shows, the slots from S-1 to S+2 are mapped to the collapsed buffer, but only the contents of S+1 and S are really marked as reserved so that they cannot be discarded by the system. As the time-slot vector rotates over the video vector synchronously with the video exhibition, new units of video enter and leave the slots. In case one slot of the time-slot vector is marked as being the second slot of the collapsed buffer (marked S in Fig. 6), the overflow pointer **O** become active, and there is a stream feeding this buffer. If the writing pointer **W** crosses the overflow pointer **O**, a warning signal is generated to inform C-CVC of a risk of collapsed buffer's overflow. That signal generates a message to the stream provider so that it should decrease the speed of video delivery. Similarly, the S+1 slot of the collapsed buffer activates the underflow pointer **U** and the reading pointer **R**. If the writing pointer reaches or moves below the underflow level a warning is generated to the C-CVC proxy, which will search for a new source to provide the adequate stream content. Note that the C-CVC proxy is reactive in the sense that it interacts with the server to ask for sending GoFs quickly or slowly and also for supplying new video streams. The content pointed by the reading pointer (R) is transmitted to the clients that share the associate collapsed buffer.

The design of a VoD system based on C-CVC involves two key issues that we discuss next: (1) how to control the slots of collapsed buffers and (2) how to discover which slots should form the collapsed buffer in a cost-effective manner when a client logs into the system.

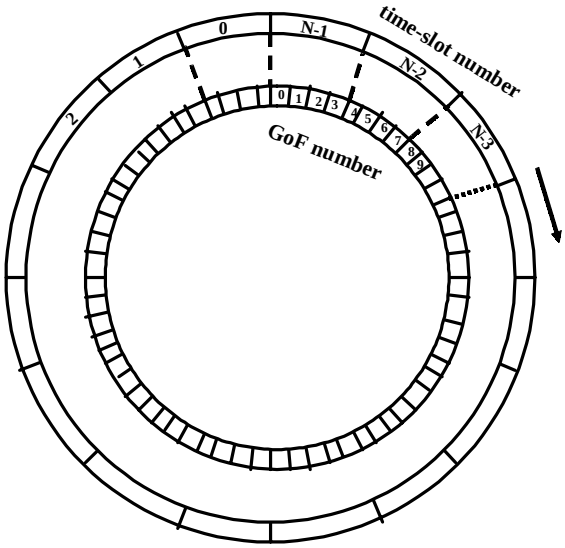

Figure 5. The time-slot vector (external circle) and the video vector (internal circle).

Upon the first request for a certain video arrives at the proxy, the C-CVC manager creates the data structure for the video, as shown in Figure 5, and sets its initial state appropriately. To see how that structure works, we can think the video vector is static while the time-slot vector runs clockwise at the playback speed over the video vector. The time-slot vector starts running at the time ST immediately after its creation. The initial slot S+2 of the collapsed buffer (see Fig. 6) that will be reserved for the client, is calculated by subtracting the current time CT and the initial time GTS (GoF TimeStamp) of the selected segment in relation to the video's starting time from time ST. The number of traversed slots is obtained by dividing the previous result by the slot duration (SD) and by using that number to the NS (number of slots) module operation we have the slot number (S+2). This calculation can be translated by the following expression:

$$(S+2)= [(ST-CT-GTS)/SD]\%NS$$

Usually, a video request asks for starting the video transmission at the beginning, i.e., GTS is equal zero, although the system also allows transmission to start at any point within the video stream.

The collapsed buffer is an useful abstraction since what actually controls both video content and its flow through collapsed buffers mapped in the proxy cache are the time slots that C-CVC implements. Every time-slot status corresponds to one place in the collapsed buffer so that if it is not in any of the collapsed buffers then it is set as non-reserved. A non-reserved time slot means that the slot does not monitor its virtual content. The content is considered as being virtual because the slot does not store video content - it only indicates which video units in the video vector it controls. Updating the limit pointers of a slot at every step is not cost-effective, so the C-CVC proxy calculates such pointer values on demand. Given ST, CT, PT (the period of time for a complete rotation, i.e., the video's duration plus padding), and the slot position (SP) within the slot vector, then the number of cycles NC can be calculated by:

$$NC = (ST - CT - SP * SD)/ PT$$

The slot starting time (SST) and the slot final time (SFT) are given by:

$$SST = ST - CT - NC * PT - SP * SD \text{ and } SFT = SST - SD$$

Both SST and SFT return time values relative to the media starting time. Since GoFs can be accessed using either its sequential number or its timestamp, the C-CVC proxy can delimit the GoFs managed by a certain slot using the simple expressions as stated above.

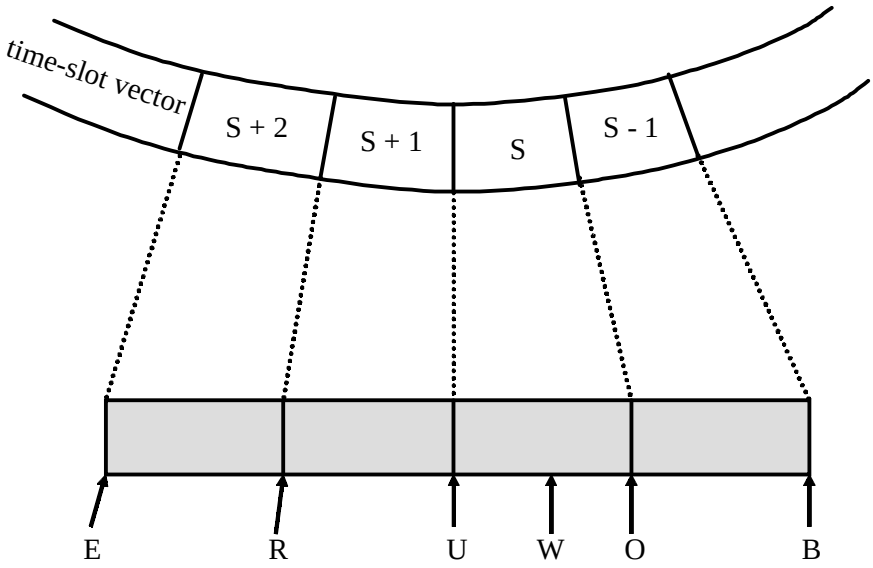

Figure 6. Projection of the collapsed buffer on the slots vector.

4. Experimental Methodology

Using the NS-2 version 2.1b7 simulation tool [Breslau et all 2000] we evaluated the effectiveness of the Collapsed CVC technique for delivering popular videos in prime time periods. We make no considerations about the access point equipment that connect the clients to the CDN, which can be either a router, a switch or even a DSLAM (Digital Subscriber Line Access Multiplexer). Connected to this access point a C-CVC proxy capable of supporting up to 1000 simultaneous connections. It is worth to note that current DSLAM companies offer equipment that support 2000 or more ADSL ports with downstream rate that reaches up to 11 Mbps [ALCATEL 2002]. Our simulations use the trace of the first one hour of the Oliver Stone's motion picture Platoon, with MPEG2 wide-screen format and an average throughput rate of 8 Mbps (refer to Fig. 7). We choose the DVD video quality because VoD services will compete with cable TV, HDTV (High Definition TV) and users are unwilling to subscribe for low quality services.

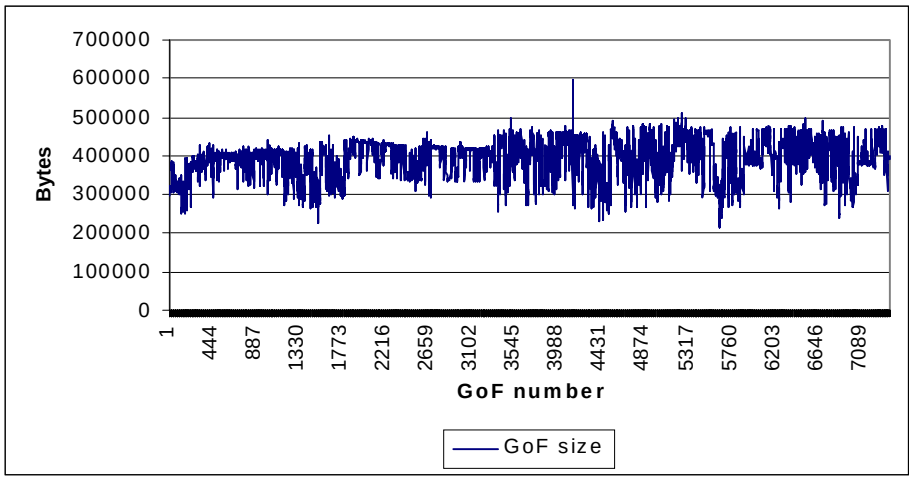

Figure 7. Graphic with the Platoon video GoF size used in the simulation. GoFs with 12 pictures on average for a 29.97 frame rate per second.

For evaluation purposes, we assume that the server´s capacity corresponds to 1000 logic channels, which is equivalent to that of a conventional unicast system that supports the same amount of simultaneous clients. Also, the server can store 100 videos, each of which lasts one hour and occupies 3.2 GBytes. For simulation purposes, we assume the videos are different, although we used a single video trace. The client requests follow a Poisson arrival process and the selected video is chosen according to the Zipf distribution [Zipf 1929].

Table 1. Simulation Parameters

Parameter	Standard value	Variation
Time of simulation	1 hour	-
Avg time between arrivals	3.1 s	3.1 to 300 s
Video length	1 hour	-
Video size	3.2 GBytes	-
Average transmission rate	8 Mbps	-
Slot size	8 Mbytes	-
Collapsed buffer size	4 slots	-
Cache size	32 Gbytes	0 to 320 GBytes
Client buffer size	8 Mbytes	-

Table 1 shows the simulation's main parameters. The initial simulations used only one single video to study the local behavior of the collapsed CVC. Next, we present simulations results for multiple videos so as to analyze the C-CVC's global behavior under highly-demanding working conditions. Also, we choose to use a large cache size because it can be split between main memory and local disk, besides the fact that the client playback buffer could be reduced substantially. In particular, by combining the

client buffer with the collapsed buffer opens new system design possibilities. One is to reduce the buffer size of each client's set-top-box, making it cheaper. Another possibility is to use the extra buffer size to save CDN resources. For instance, allowing a video flow with greater jitter (less QoS) that can be absorbed by the collapsed buffer.

We can estimate the amount of memory necessary for a given video quality using the video's average bit rate. As the Figure 8 shows, a low-quality video that is typically used in the Internet, with 7.1 frames-per-second resolution and bandwidth of 10 Kbps needs only 40 Mbytes of proxy cache, keeping other simulations parameters fixed. Also, a VCD-quality video with 1.5 Mbps needs 6 GB cache to serve 100 different videos. Apparently, 32 GB of proxy cache to handle 8 Mbps MPEG2 DVD quality video is an enormous quantity of memory. However, available 64-bit microprocessors, like the Intel Itanium, is sold with mother boards supporting more than 128 GB of RAM at an affordable price. An alternative is to use MPEG4 compression with DVD quality video and an average of 1.5 Mbps, but the decoder processing power prevents its use in low-cost set-top-box.

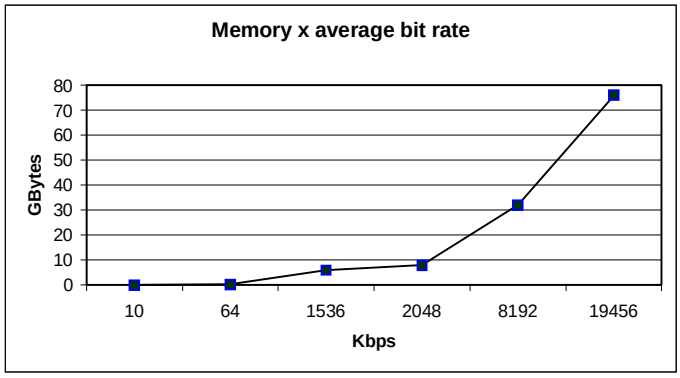

Figure 8. Proxy cache size versus video bit rate assuming the cache stores up to 20% of one hundred 60-minute videos.

4.1 Evaluating the Collapsed CVC behavior under one single video

To evaluate the C-CVC behavior we used three performance metrics according to the request arrival rate: (a) the amount of busy slots, (b) the average number of free slots between busy slots, and (c) the average latency. The number of busy slots indicates the number of collapsed buffers, which are related to the clients that will receive the proxy stream, so as to distinguish them from the temporary reserved slots. The average number of free slots between busy slots is a measure of how close is one busy slot to another. The average latency indicates the length of time that clients keep waiting between the video request and its exhibition.

Figures 9 and 10 show that as the arrival rate increases so does the number of busy slots while the average number of free slots between busy ones decreases. Overall, we infer that the busy slot density grows directly with the arrival rate, which implies that C-CVC should adjust dynamically the video contents held in the cache according to video popularity. In other words, if a video has a high arrival rate then it is very popular and almost all of its video GoFs are likely to be kept in the proxy. In contrast, if a video is unpopular then only few parts of it are necessary to be stored in the proxy´s cache. Another important detail is that even with high arrival rates the amount of clients per

busy slots is on average no more than 2 clients per slot. This reveals that the memory economy of C-CVC is obtained partially by more than one user sharing the slots while most of the economy comes from the busy slot density. Therefore, the larger the number of busy slots, the greater is the amount of reserved video units in the cache and consequently the larger is the recycled content.

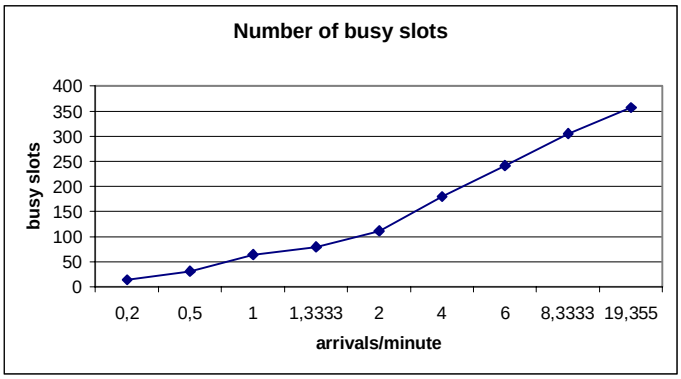

Figure 9. The number of busy slots for one single video versus arrival rate.

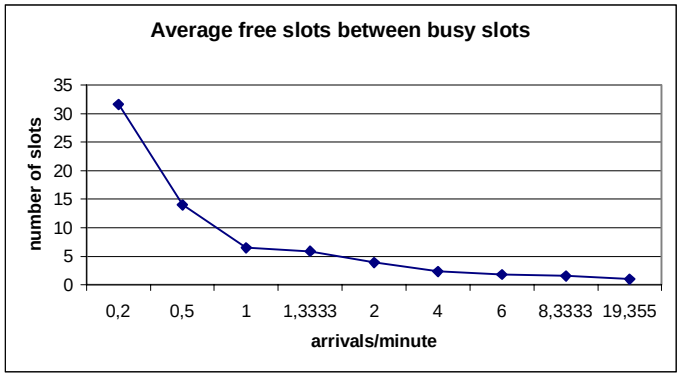

Figure 10. The average number of free slots between busy slots.

Figure 11 shows C-CVC's average latency. As can be seen in the figure, the average latency decreases with the increase of the arrival rate. This result is mainly due to the latency of first requests, whose delays to fill a buffer are greater for buffers that are not in the cache. When a larger number of clients share the same video, the initial latency is reduced to the time necessary to complete the smaller set-top-box buffer. This is one of the advantages of using video prefix [Sen, Rexford and Towsley 1999], which retains the initial part of the video in the cache. Ideed, C-CVC enforces low discard priority for prefix of popular videos by keeping them stand-by and outside of the collapsed buffers.

4.2 Evaluation of collapsed CVC under multiple videos

In order to study the C-CVC behavior with several videos, we employed the 3 metrics we used for evaluating one single video performance. In Figure 12, the graphics show that popular videos with large ranks have greater number of busy slots and also smaller average number of free slots between busy slots.

The remaining 70 videos are not plotted in Figures 12 and 13 because the slot densities are very low at 0 to 5 clients per video, each one occupying different slots most of the time. On average, 10% of all the requests are spread over the 70 least popular videos. C-CVC treats such cases, by using the memory savings that more popular videos provide to compensate for the low slot density of low-rank videos. This solution represents a compromise between using proxy memory instead of network/server bandwidth.

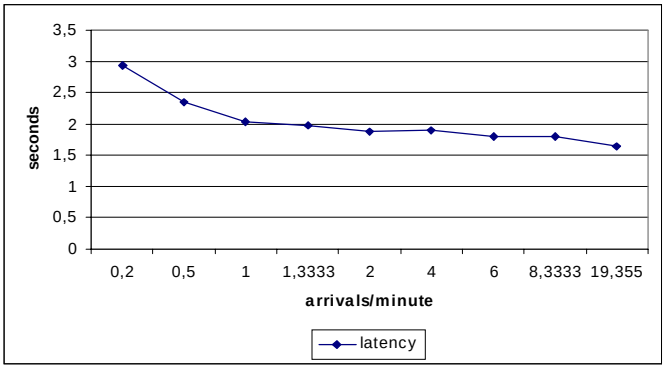

Figure 11. The average latency to start a video exhibition.

Figure 14 shows the server utilization rate for a proxy cache that can store from 0% to 100% of the 100 video contents that are held in the server. The figure suggests that without a cache the system behaves like a conventional system. In this case, all of the 1000 clients that logged into the system occupied the 1000 logic channels that the server supports. For a 10% cache that can store up to 10% of the capacity of 100 videos stored in the server, the server utilization rate drops drastically to less than 31%. A dramatic decrease is achieved with a 20% cache in which the server utilization rate is reduced to 13% only.

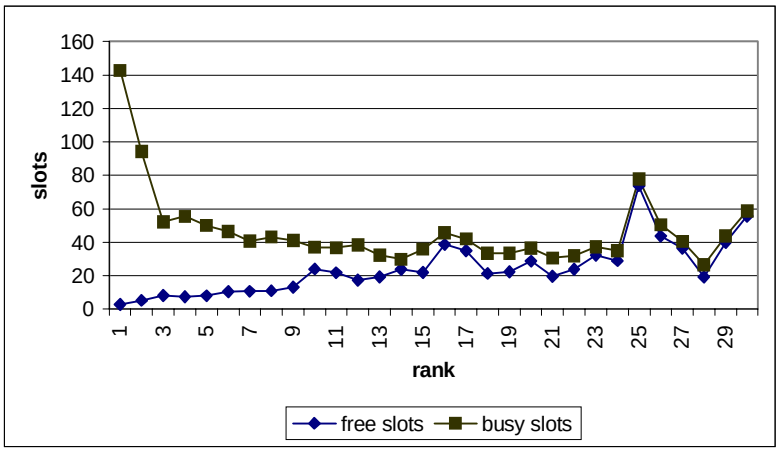

Figure 12. Number of busy slots and average number of free slots between busy slots versus video rank

Although the 20% cache corresponds to a 64 GB-size cache, which the C-CVC VoD system requires to support DVD-quality video, actually more than half of the cache size can be stored in the local hard disk. Also, we note that C-CVC reduces substantially buffer size requirements of the client set-top box since most part of the

system jitter is handled by the collapsed buffer, and each of the 8 MBytes that each client buffer saved resulted in total memory savings of 80 GBytes for the system with 1000 active clients.

Figure 13. Number of clients per video versus the video rank acording to the Zipf distribution

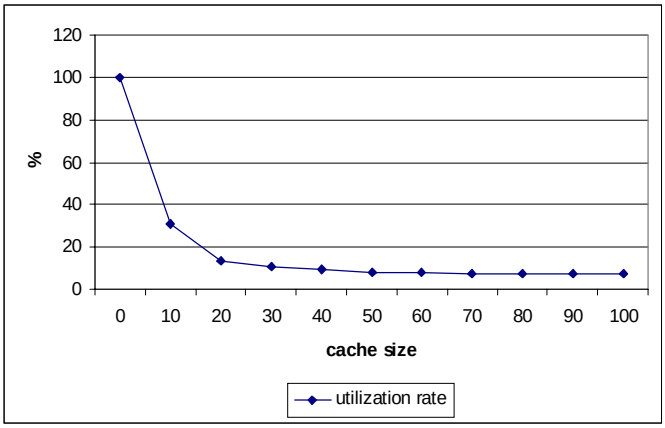

Figure 14. C-CVC performance for several cache sizes. The cache size ranges from 0 to 100 % of the space necessary to store 100 videos.

Figure 15 shows the amount of server streams versus cache size. It is worth to note that with a 10% cache, the number of server streams is quite large although the utilization rate decreases. This is explained by the increasing of short streams (patches) to refill discarded contents of the cache. This also explains a substantial decrease of the server utilization rate for the 20% cache. With larger caches less patches are necessary, which translates into the smaller utilization rate.

Figure 16 shows that without a cache the average latency is bigger because it is necessary to fill the client buffer. In case of using a cache, the client buffer is only the minor portion of the total buffer, since the major part is held in the collapsed buffer, whose initial content corresponds to the beginning of the video, which in turn has a high probability of being in cache. From the graphics in the figure we conclude that C-CVC can sustain on average small reply´s latency to client requests.

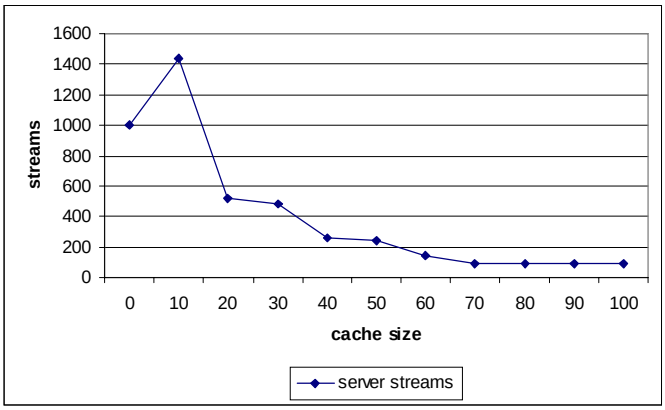

Figure 15. The effect of cache size on the amount of regular streams and patches the server generates. The cache size ranges from 0 to 100 % of the space necessary to store 100 videos.

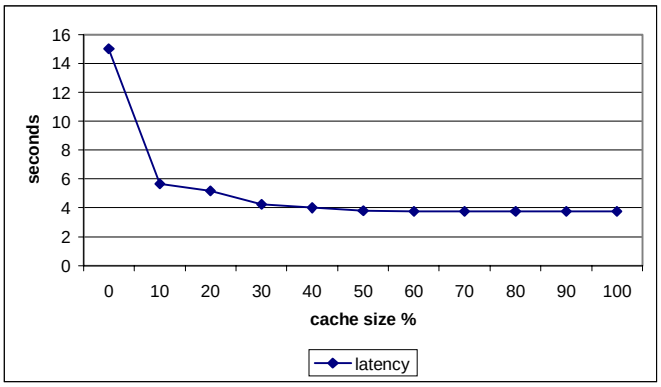

Figure 16. Latency.

5. Conclusions and Ongoing Work

In this work we introduced and evaluated the collapsed CVC technique for building scalable VoD systems using a high-quality video stream with MPEG2 wide-screen format and average transmission rate of 8 Mbps. We showed that C-CVC accomplishes QoS and significant bandwidth savings in both the content distribution network and the VoD server. Central to C-CVC performance is the collapsed buffer, a new mechanism that C-CVC introduces in the proxy server, which reduces the buffer length of the client's set-top-box and its associate costs.

Our performance results revealed that under C-CVC a proxy cache that can store at most 20% of the total video contents allows the VoD server to decrease significantly the average exhibition latency. Most importantly, C-CVC reduced the network traffic and busy channels in the server to only 13% in comparison with that of the VoD server without C-CVC. Furthermore, our results were obtained for a VoD server under stressing conditions, in which the system demand corresponded to 1000 clients accessing concurrently one hundred videos stored in the server**.**

Overall, the video proxy based on C-CVC is an excellent choice for an interface between the distribution network and the access network, using either ADSL or SDSL

connections in the so-called *last mile* to the home user. In this context, WANs, could have several DSLAM equipments and several proxies that collaborate among themselves to increase the overall performance of VoD systems. So, the next step in our research is naturally to study the behavior that accrues from the cooperation of several C-CVC proxies. In addition, using C-CVC to provide VCR-like interactive services is under way.

6. ACKNOWLEDGMENTS

The authors thank Brazil´s funding agencies Finep, Capes and CNPq, for supportting this research.

REFERENCES

Ishikawa, E. and Amorim, C. L. (2001) “Cooperative Video Caching for Scalable and Interactive VoD Systems”, *IEEE ICN´01*, Colmar, France, July.

Aggarwal, C., Wolf, J. and Yu, P. (1996) “On Optimal batching policies for video-on-demand storage servers”, *IEEE International Conference on Multimedia Computing and Systeems*, June.

Golubick, L., Sitaram, D. and Shahabuddin, P. (1996) “Adaptive piggybacking: a novel technique for data sharing in video-on-demand servers”, *ACM Journal of Multimedia System*, 4(3), pp. 140-155, June.

Sheu, S., Hua, K. A. and Tavanapong, W. (1997) “Chaining: The generalized batching technique for video-on-demand systems”, *IEEE International Conference on Multimedia Computing and Systems*, Ottawa, Ontario, Canada.

Hua, K. A., Drops, Y. and Sheu, S. (1998) “Patching: The multicast technique goes true video-on-demand services”, *ACM Multimedia´98*, Bristol, England, September.

Carter, S. and Long, D. (1997) “Improving video-on-demand server efficiency through stream tapping”, *International Conference on Computer Communications and Networks*.

Gao, L., Zhang, Z. and Towsley, D. (1999) “Catching and\Selective Catching: Efficient Latency Reduction Techniques for Delivering Continuous Multimedia Streams”, *ACM Multimedia´99*.

Eager, D., Vernon, M. and Zahorjan, J. (2000) “Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand”, *Multimedia and Computing 2000*, San Jose, CA, January.

Pinho, L. B., Ishikawa, E. and Amorim, C. L. (2002) “GloVE: A Distributed Environment for Low Cost Scalable VoD Systems”, IEEE 14th Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2002, Brazil, October.

Pinho, L. B., Ishikawa, E. and Amorim, C. L. (2003) “GloVE: A Distributed Environment for Scalable VoD Systems”, to appear in The International Journal of High Performance Computing Applications, volume 17, number 2, summer.

Zipf, G. K. (1929) "Relative frequency the determinant of phonetic change", *Harvard Studies in Classical Philology*, Volume XL.

Sen, S., Rexford, J. and Towsley, D. (199) “Proxy prefix caching for multimedia streams”, *IEEE Infocom´99*, New York.

Acharya, S. and Smith, B. (2000) “MidleMan: A Video Caching Proxy Server”, *NOSSDAV 2000*, Chappel Hill, NC, June.

Bommaiah, E., Guo, K., Hofmann, M. and Paul, S. (2000) “Design and Implementation of Caching System for Streaming Media over the Internet”, *IEEE Real Time Technology and Applications Symposium*, Washington, DC, May.

Verscheure, O., Venkatramini, C., Frossard, P. and Amini, L. (2002) “Joint server scheduling and proxy caching for video delivery”, *Computer Communications 25 (2002) 413-423*.

Almeida, J. M., Eager, D. L., Ferris, M. and Vernon, M. K. (2002) “Provisioning Content Distribution Networks for Streaming Media”, *IEEE Infocom 2002*, New York, NY, June.

Wang, B., Sen, S., Adler, M. and Towsley, D. (2002) “Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution”, *IEEE Infocom 2002*, New York, NY, June.

ALCATEL (2002) “ALCATEL 7300 ASAM data sheet”, *http://www.alcatel.com*

ALCATEL (2002), “ADSL line card (24-port) data sheet”, *http://www.alcatel.com*

Breslau, L. et all (2000) “Advances in Network Simulation”, IEEE Computer, 33(5), p.p. 59-67, May.