Minimizing the Impact of Delay on Live SVC-based HTTP Adaptive Streaming Services

Niels Bouten, Steven Latré, Jeroen Famaey, Filip De Turck
Department of Information Technology, Ghent University - iMinds
Gaston Crommenlaan 8/201 B-9050 Ghent, Belgium
Email: niels.bouten@intec.ugent.be

Werner Van Leekwijck
Alcatel-Lucent Bell Labs
Copernicuslaan 50
B-2018 Antwerp, Belgium

***Abstract*—HTTP Adaptive Streaming (HAS) is becoming the de-facto standard for Over-The-Top video streaming services. Video content is temporally split into segments which are offered at multiple qualities to the clients. These clients autonomously select the quality layer matching the current state of the network through a quality selection heuristic. Recently, academia and industry have begun evaluating the feasibility of adopting layered video coding for HAS. Instead of downloading one file for a certain quality level, scalable video streaming requires downloading several interdependent layers to obtain the same quality. This implies that the base layer is always downloaded and is available for playout, even when throughput fluctuates and enhancement layers can not be downloaded in time. This layered video approach can help in providing better service quality assurance for video streaming. However, adopting scalable video coding for HAS also leads to other issues, since requesting multiple files over HTTP leads to an increased impact of the end-to-end delay and thus on the service provided to the client. This is even worse in a Live TV scenario where the drift on the live signal should be minimized, requiring smaller segment and buffer sizes. In this paper, we characterize the impact of delay on several measurement-based heuristics. Furthermore, we propose several ways to overcome the end-to-end delay issues, such as parallel and pipelined downloading of segment layers, to provide a higher quality for the video service.**

I. Introduction

HTTP Adaptive Streaming (HAS) is becoming the de-facto standard for Over-The-Top video streaming services. This shift was mainly induced by the advantages offered by HTTP-based streaming: reliable transmission over TCP, reuse of existing caching infrastructure and compatibility with NATs and firewalls. Initially, the HTTP-based video protocols required downloading the entire video file before playout. Later, progressive download techniques allowed playout to begin after a sufficient amount of data was stored in the buffer. However, when congestion arised in the network, these protocols were not able to cope with buffer starvations, leading to playout gaps, which have a negative impact on the provided service quality. The third evolution in HTTP-based streaming, being HAS, tackles these shortcomings by splitting the content into segments which are encoded at different quality levels. Client heuristics then decide at which quality rate the next segment should be downloaded, taking into account network statistics, buffer filling and device characteristics. HAS therefore provides service assurance, at a reduced quality, if congestion occurs in the network.

Traditionally, Advanced Video Coding (AVC) is used to encode the different segments, introducing a significant amount of redundancy across quality representations. Scalable Video Coding (SVC) can cope with these issues of content redundancy by creating dependencies between the base and enhancement layers. Adopting SVC in HAS significantly improves caching and bandwidth efficiency at the server side, while reducing the risk of running into frame freezes at the client, since for every segment the base layer is always downloaded. Furthermore, since SVC requires multiple layers to be downloaded, quality rate adaptations can be performed at higher granularity, since throughput fluctuations can now be detected earlier, allowing heuristics to react faster.

However, there are several drawbacks to adopting SVC in HAS [1], [2]. First, separating the video flow into several SVC layers introduces a coding penalty, which leads to an encoding overhead of approximately 10% per enhancement layer. Second, when downloading multiple layers per segment, we are generating more requests at the client to download subsequent layers of a single segment. These request-response cycles introduce a wait time between the reception of the last byte of the previous segment layer and the first byte of the next segment layer, which is equal to the round trip time (RTT). Third, when considering Live TV over HAS, the segment sizes should be as small as possible to reduce the latency on the broadcast signal. These smaller segments increase the decision granularity even further, but at the same time increase the idle time between two consecutive downloads, negatively impacting the service quality for the user.

In this paper we propose and compare several AVC and SVC-based client heuristics and evaluate their behavior in a Live TV setting where client side buffers need to be small to reduce latency on the broadcast signal, while still providing service assurance. The heuristics decide on the best quality to download next, for the video streaming service, based on low level monitoring data such as bandwidth measurements and the status of the client's play-out buffer. Furthermore we propose and compare several download scheduling approaches such as sequential, pipelined and parallel HTTP downloading and their ability to reduce the impact of latency on service delivery. Additionally, we investigate the impact of high round trip times on SVC-based HAS streaming in a Live TV setting.

The remainder of this paper is structured as follows. Section II provides an overview of relevant work, followed by an overview of existing adaptation heuristics in Section III and of a novel SVC-based heuristic, specifically designed for small buffers in Section IV. Section V discusses how adapting the download scheduling can improve the quality and service assurance. Section VI elaborates on the experiment setup and evaluation results, which are summarized in Section VII.

978-3-901882-50-0 ©2013 IFIP

II. Related Work

The increased popularity of video consumption over the Internet has led to the development of a range of protocols that allow adaptive HTTP-based video streaming. Some of the major players have introduced their own protocols, server and client software such as Microsoft's Silverlight Smooth Streaming [3], Apple's HTTP Live Streaming [4] and Adobe's HTTP Dynamic Streaming [5]. More recently, a standardized solution has been proposed by MPEG, called Dynamic Adaptive Streaming over HTTP (DASH) [6]. Even though differences exist between these implementations, they adopt the same design principles. The video is split into several segments, encoded at different quality rates. These segments are offered by a web server and are transmitted over standard HTTP connections. The intelligent video client uses a selection heuristic to dynamically adapt the quality, based on the current network statistic, buffer filling and other device characteristics.

Optimizations of HAS-based delivery can be performed at the server, the network or the client. At the server side, optimizations are focussed on the encoding scheme. Traditional deployments of HAS use the H.264/AVC codec for creating the different representations of the video. For each representation a separate file needs to be stored at the video server, leading to an increased storage penalty due to the redundant information. Adopting a Scalable Video Codec (SVC) extension to H.264/AVC [7] or High Efficiency Video Coding (HEVC) [8], allows alleviating the storage issues with AVC at the server, while improving caching efficiency. Huysegems et al. discuss the advantages of using SVC instead of AVC, such as guaranteed playout during fluctuations since the base layer is always downloaded and a reduction in bandwidth and storage requirements at the server [2]. However, important challenges for SVC are indicated to be the encoding overhead of SVC and the increased vulnerability to high round trip times. In this paper, we focus on these high round trip times and how their negative impact on SVC-based HAS can be reduced.

Liu et al. present an in-network optimization of HAS for 3GPP networks. By parallelizing the download and request of HAS segments, a better resource utilization can be achieved [9]. Bouten et al. have discussed how a network provider can manage the quality that is offered to the clients in a HAS environment [10]. The solution takes into account subscription parameters and device parameters to restrict the qualities that are offered to the client. An autonomic delivery framework is presented in previous work [11], [12], which allows to reduce the consumed bandwidth by grouping unicast HAS sessions sharing the same content into a single multicast session. In this paper, we focus on measures at the client side rather than the network with a focus on the scheduling of segment requests in networks with high round trip times.

Each commercial HAS implementation comes with an existing video client heuristic of its own. Akhshabi et al. compare several commercial and open source HAS players and indicate significant inefficiencies in each of them [13]. Several heuristics have been proposed in literature as well, each focussing on a specific deployment. Liu et al. discuss a video client heuristic that is suited for CDNs by comparing the expected segment fetch time with the experienced segment fetch time to ensure a response to bandwidth fluctuations in the network [14], while Adzic et al. present a client heuristic which is tailored for mobile environments [15]. Jiang et al. aimed to develop an efficient, fair and stable heuristic by randomizing chunk scheduling to avoid synchronization, stateful bitrate selection and delayed update to avoid instability [16], and compared their approach to commercial players. Generic algorithms exist for selecting the next video quality to download, using a priority-based scheme where base layers receive higher priority in download scheduling compared to the enhancement layers [17]. Andelin et al. provide a heuristic which was specifically designed for SVC and using a slope to define the trade-off between downloading the next segment and upgrading a previously downloaded segment [18].

In this paper, several heuristics are evaluated that are either based on commercial players, algorithms described in literature or designed from scratch. Furthermore we focus on how downloads should be scheduled in networks suffering from high round trip times and compare sequential, pipelined and parallel download schedulers.

III. State of the Art Rate Adaptation Heuristics

In this section we summarize the state of the art rate adaptation heuristics for AVC and SVC-based HAS. For an extensive description of these heuristics, we refer to [1].

A. AVC Microsoft's Smooth Streaming Heuristic

The heuristic for AVC video streaming is based on an open source version of the algorithm of the Microsoft Smooth Streaming (MSS) video player [1]. The heuristic can be configured using 3 thresholds: the panic (P), lower (L) and upper (U) threshold. There are two states: *buffering* and *steady state*. During the *buffering state*, quality decision is based on measured throughput and can only be increased with one level. When the buffer level is equal to or exceeds $L+(U-L)/2$, the heuristic goes into *steady state*. If during the *steady state*, the buffer filling level is slowly changing but lower than L, the quality level is decreased. When the buffer filling level is between L and U and quickly increasing or when the buffer filling level exceeds the upper threshold U, the heuristic attempts to improve the quality level if the throughput measurements indicate that the next segment can be downloaded in time. If the buffer filling level drops under the panic threshold P or the buffer filling level is quickly decreasing and lower than L, the next segment is downloaded at the lowest quality level and the heuristic returns to *buffering state*.

B. SVC MSS Heuristic

The AVC MSS heuristic can also be used for SVC-based HAS when the quality decisions are translated into subsequent layer downloads. SVC MSS however, allows adapting the quality decisions in between two consecutive layer downloads. Thanks to the finer granularity, SVC MSS is able to cancel the download of one or more enhancement layers when the measured throughput indicates that they will not arrive in time for playout.

C. SVC Slope Heuristic

The SVC Slope heuristic exploits two main advantages of SVC-based HTTP adaptive streaming. First, SVC video streaming allows more fine grained decisions as the quality decision can be adapted after every download of a layer rather than every segment. Second, since SVC video layers are interdependent, the heuristics can decide either to download

[1]Source available fromhttps://slextensions.svn.codeplex.com/svn/trunk/SLExtensions/AdaptiveStreaming

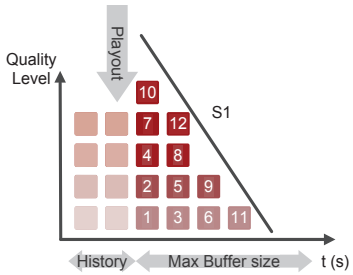


Fig. 1: Illustration how a steeper slope prioritizes backfilling over prefetching.

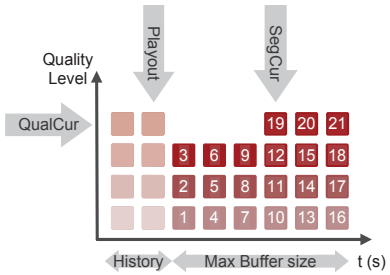


Fig. 2: Illustration of the backfilling operation by SVC Cursor when the quality cursor was improved.

the base layer of a new segment or to increase the quality of a previously downloaded segment by downloading additional enhancement layers. This backfilling is also possible with AVC, but since the redundant data is downloaded again, this affects the efficiency of the heuristic drastically.

Andelin et al. proposed a slope-based SVC heuristic [18], where the backfilling is limited by a moving buffer slope. The slope can be configured to give priority to either prefetching (downloading lower quality layers for future segments) or backfilling (downloading additional enhancement layers for buffered segments). This configuration is done by defining a slope in the heuristic: the steeper the slope, the more backfilling will be chosen over prefetching. Similarly, the flatter the slope, the more prefetching will be done for future segments. This filling behavior is illustrated in Figure 1, which shows the segment and layer download order for a configuration S_1 of the slope parameter.

IV. SVC Adaptation Heuristic for Small Buffers

A novel heuristic for scalable video was designed that is able to cope with small buffer sizes, which are common in a Live TV scenario. To be able to optimize service quality for the end-user, three factors are important for the client: 1) avoiding frame freezes and gaps in video playout, 2) ensuring quality stability limiting the number of quality switching occurrences and 3) allowing high quality streaming. The SVC cursor based algorithm is designed to avoid gaps and limit quality switches while trying to provide the highest possible quality. This is accomplished by using two distinct cursors: segment cursor and quality cursor, defining which segment is under consideration for the next decision and the goal quality respectively. Limiting the number of switches is further accomplished by using a timeout for the quality improvement decision. The segment cursor advances to the next segment when a) all qualities up to the quality cursor are downloaded for the current segment cursor or b) the layer under consideration cannot be downloaded in time. When a layer will not be downloaded in time for playout, the quality cursor is decreased and the improvement timer is reset. The quality cursor can only be incremented when all lower layers of every segment are downloaded and the improvement timer has timed out. The segment cursor is then moved based on the estimations of the arrival times of these enhancement layers and their playout times, evaluated from right to left. When some of the enhancement layers are estimated to be downloaded before their respective playout time, we increment the quality cursor and start downloading from left to right as shown in Figure 2, after which the improvement timer is reset. So when the quality cursor is increased, a backfilling operation updates the buffer to the required quality level.

V. Delay-optimized Download Scheduling

As discussed, adopting layered video coding has several advantages for HAS. But at the same time, there is a major drawback when considering networks with high RTTs. Since multiple layers per segment need to be downloaded, more request/response cycles are induced when downloading the subsequent segment layers. As a result, there is a wait time between receiving the last byte of the previous segment layer and the first byte of the next segment layer. This idle-time is equal to the RTT. When HAS is applied to a Live TV scenario, the drift on the broadcast signal is to be kept as small as possible to enable the streaming for live events and the use of second screen applications. As a consequence, the segment size should be as small as possible. When reducing the segment size however, the impact of the idle-time between downloading two consecutive segments even increases. When the RTT is 50msec and a 5-layer SVC video representation is used with a 1 second segment size, the idle-time accounts for 250msec when downloading the highest quality representation, which is one fourth of the available download window. This example shows the importance of alleviating the impact of RTT on SVC HAS in Live TV scenarios. This can be achieved by applying pipelined or parallel download scheduling, both of which are able to eliminate the incurred idle times.

A. Pipelined Scheduling

HTTP pipelining is a technique in which multiple HTTP requests are sent on a single TCP connection without waiting for the corresponding responses. Kaspar et al. proposed Pipelining to improve progressive downloading, the predecessor of HAS [19]. Pipelining all requests at once will of course not yield a viable solution, since we then sacrifice the ability for fast response to network changes. We propose to estimate the RTT and to schedule the next request RTT seconds before the current download will be finished. This technique allows postponing the decision on which segment layer to download as long as possible, while still eliminating the idle time between two consecutive downloads. With perfect estimations of download time and RTT, the delay could be completely eliminated as shown in Figure 3(a). However with varying RTT, this will not be the case and RTT will be overestimated or underestimated most of the time. When overestimating the RTT, the decision on which segment layer to download is taken too early and conditions could change during this period. But as illustrated in Figure 3(b), the overestimation of the RTT cannot be noticed at the client side which has no indication of the request being queued at the server. When underestimating the RTT however, there is a gap between the two consecutive

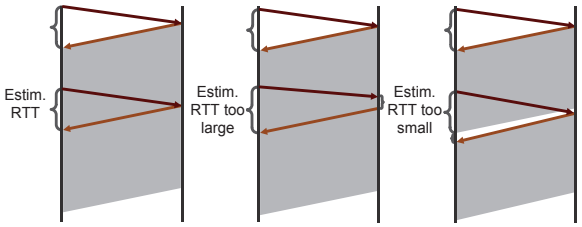


Fig. 3: Overview of estimation with pipelining a) accurate estimation b) overestimation c) underestimation.

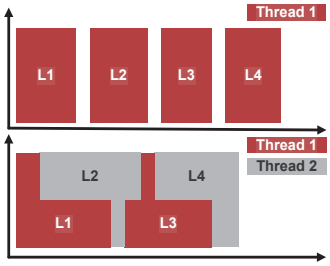


Fig. 4: Illustration of the delay masking behavior of parallel scheduled segment layer downloads.

downloads, which can be measured at the client side. Adding this measurement to the estimated RTT, yields an accurate estimation for the RTT as shown in Figure 3(c). The proposed approach for the pipelined scheduling is to linearly decrease the estimated delay until an underestimation is perceived ($t_{first\_byte\_s} - t_{last\_byte\_s-1} > 0$), and an accurate estimation for the delay can be established.

B. Parallel Scheduling

Another approach to avoid idle time between consecutive downloads is to request several segments at the same time using parallel TCP connections. This allows request/response cycles to interleave with active downloads, reducing the idle time. A simplified example of this masking behavior is shown in Figure 4. An additional advantage of parallel download scheduling is the improved performance when using parallel TCP connections. However, since now the segment layer downloads are requested over concurrent TCP connections, they compete for the available bandwidth and the download times are proportional to the number of threads. The disadvantage is of course that the base layers take longer to complete and thus an increased risk of buffer starvations. Therefore, the number of parallel threads needs to be limited.

VI. Experimental Results

The performance of the video client heuristics and download schedulers was evaluated using the NS-3 network simulator[2] in combination with the Network Simulation Cradle[3]. Figure 5 illustrates the used network topology with N clients connected to the HAS Server via a router. Each client has a playout buffer of P seconds which is varied during the experiments and has a connection link with bandwidth B_c. The shared bandwidth B_s and the total RTT R are varied during the experiments.

The simulations were conducted using the traces of a variable bitrate (VBR) video file, encoded both for H.264/AVC

[2]NS-3 Network Simulator - http://www.nsnam.org/

[3]WAND Network Research Group: Network Simulation Cradle - http://research.wand.net.nz/software/nsc.php

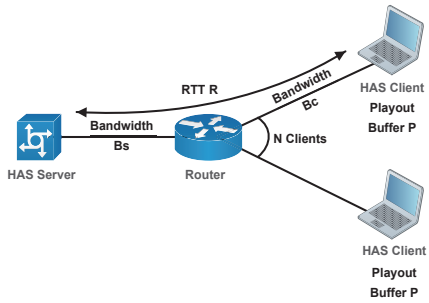


Fig. 5: Experimental setup offering a HAS-based video streaming to N clients. The parameters B_s, B_c, P, R are varied.

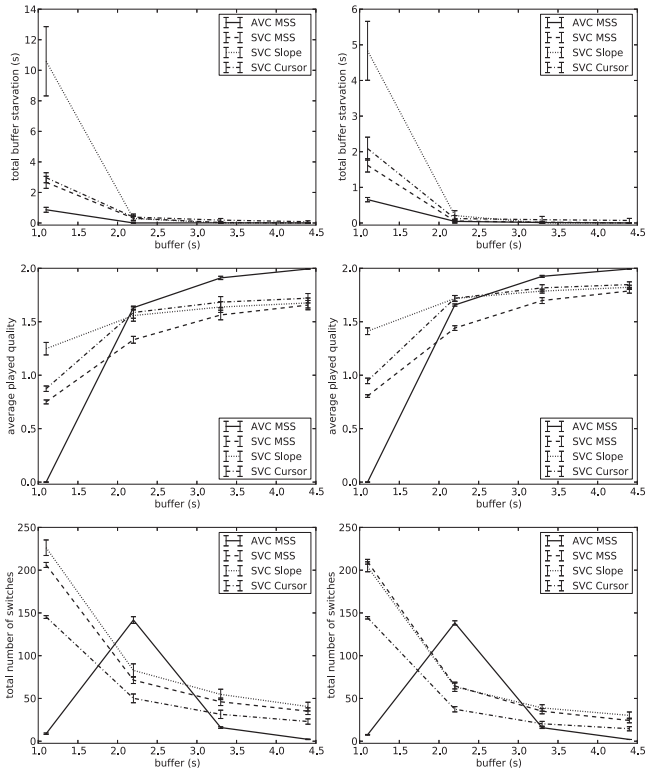
and H.264/SVC using the JSVM 9.19.15 Encoder. The video has a frame rate of 30fps and GOP size of 32 frames, which leads to a minimum segment size of 1.06667s when using I-frame segmentation. The clients were started using a Weibull startup process with average 900 seconds and shape 2.5.

A. Comparison of Adaptation Heuristics

Figure 6 shows the total buffer starvation in seconds, the average playout quality level and the total number of switches for an increasing buffer size P. All four client heuristics are compared using a sequential download scheduler. For a buffer size of 1 segment ($P = 1.1s$), AVC MSS remains in panic mode, since only one segment fits in the buffer and it needs to be played out completely before the next segment is able to fit in the buffer. This causes AVC MSS to always download the lowest quality, which explains the low number of switches and buffer starvations. For a two segment buffer however, AVC MSS constantly switches between *buffering* and *steady state*, constantly switching between representations, which explains the large increase in the number of switches. We can thus conclude that for AVC MSS, a minimum buffer size of 3 segments is required to obtain acceptable quality while avoiding quality oscillations. For small buffers, SVC Slope is able to yield the highest quality, but at the cost of a high number of switches and buffer starvations, diminishing stability and quality assurance. For buffer sizes of 3 segments and up, AVC MSS yields higher quality than the SVC algorithms. This can be attributed to two reasons. First, the overhead introduced by the SVC encoding causes a higher load on the bottleneck B_s. Second, for a larger bottleneck B_s, as illustrated in Figure6(b), the problems of the SVC-based algorithms persists because of the vulnerability of SVC to high delays. These delays in combination with the increased number of request-response cycles of SVC-based algorithms limit the throughput causing lower efficiency and thus a lower average quality. SVC Cursor is able to minimize buffer starvation time and the number of switches, while yielding the highest quality for buffer sizes containing 2 or more segments. SVC Slope tends to yield lower quality, more frequent switches and a higher overall gap time. We attribute this to the difficulty of configuring the slope parameter for the varying situations. Overall, it can be concluded that SVC Cursor outperforms all other SVC-based heuristics in terms of quality, switches and buffer starvations, therefore we use this heuristic for the subsequent results.

B. Impact of Download Scheduling

As shown in the previous results, SVC-based HAS suffers from high RTTs, leading to inefficient use of the available throughput and reducing the benefits of SVC-based HAS in



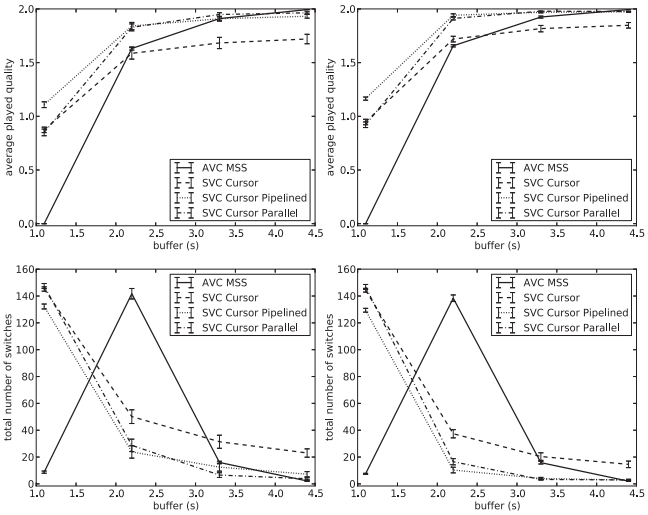
(a) (b)

Fig. 6: Total buffer starvation (s), average played quality level and total number of switches in function of the buffer size P (s) with $B_c = 10Mbps$, $N = 20$, $R = 50ms$ and (a) $B_s = 50Mbps$ (b) $B_s = 100Mbps$.

terms of caching efficiency and server bandwidth utilization. Therefore we suggest more optimized HAS-scheduling by using HTTP Pipelining and parallel downloads. Figure 7 shows the results for the same configurations as before, but with the pipelined and parallel scheduled variants of SVC Cursor. Here we can clearly see the advantages of using optimized scheduling for SVC-based HAS for high RTTs. SVC Cursor Pipelined and Parallel are able to improve the quality level, while lowering the number of quality switches at the same time. When the bottleneck is tight ($B_s = 50Mbps$), the quality yielded by SVC based HAS is only slightly lower than when using AVC-based HAS, this is caused by the encoding overhead of SVC. But when we take a look at the 100Mbps scenario, SVC-based HAS is able to outperform AVC-based HAS, in terms of switches and average quality. Even when the buffer only contains 2 segments, the quality level and number of switches are at an acceptable level.

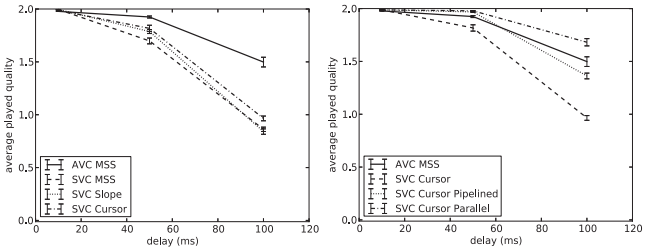
C. Impact of Delay

Figure 8 illustrates the impact of high RTTs on the different base algorithms and on the SVC Cursor algorithm in combination with pipelined and parallel scheduling for a buffer containing 3 segments. These graphs show that parallel scheduling in combination with SVC Cursor is able to outperform AVC MSS in terms of average quality when delay increases, while yielding comparable buffer starvations and a lower number of switches (graphs omitted due to space limitations). This enables service providers to deploy SVC-based HAS services, benefitting from higher caching efficiency, while avoiding the



(a) (b)

Fig. 7: Average played quality level and total number of switches in function of the buffer size P (s) with $B_c = 10Mbps$, $N = 20$, $R = 50ms$ and (a) $B_s = 50Mbps$ (b) $B_s = 100Mbps$.



(a) (b)

Fig. 8: Average played quality level in function of the RTT R (s) for $B_s = 100Mbps$, $B_c = 10Mbps$, $N = 20$ and $P = 3.3s$

drawbacks of SVC-based HAS in a high delay setting.

D. Impact of Parallel Threads

The behavior of AVC MSS and SVC Cursor in combination with parallel scheduling is illustrated in Figure 9. These results show that with increasing number of threads, parallel scheduling yields higher quality and lower switches for both AVC MSS and SVC Cursor. However, AVC MSS suffers from higher gap time with an increasing number of parallel threads, while SVC Cursor is able to lower the total buffer starvation time, while increasing quality and minimizing the number of switches.

It can be concluded that for Live TV, the smallest possible buffer should at least contain 2 segments to attain acceptable service quality. The evaluations have shown that we were indeed able to reduce the impact of high RTTs on SVC-based HAS by using parallel downloads. Furthermore, our novel SVC Cursor heuristic is able to outperform existing heuristics when considering a Live TV setting with small buffers.

VII. Conclusion

This paper quantitatively reveals the drawbacks of using SVC-based HAS Live streaming in high delay networks and proposes techniques to tackle these issues. The root causes of these are 1) the encoding overhead, yielding larger data

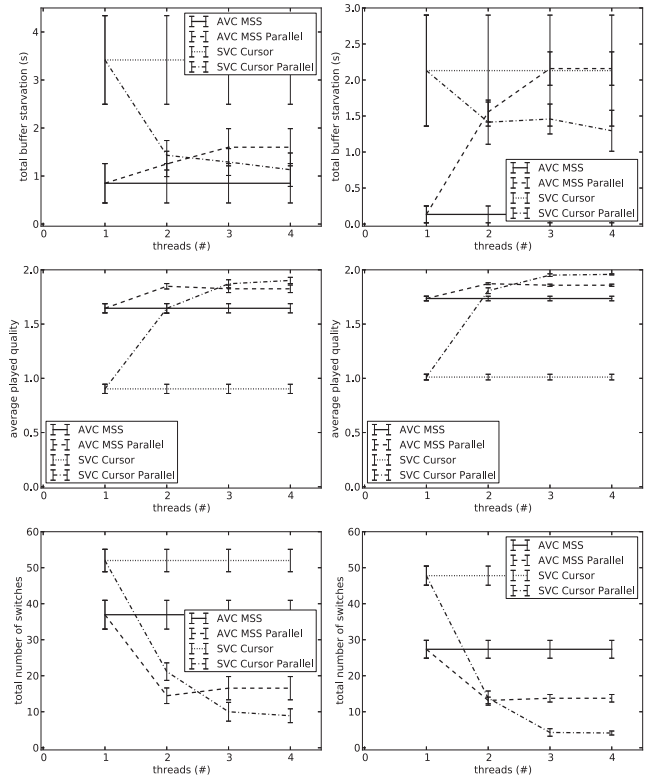


Fig. 9: Impact of the number of parallel threads with $B_c = 10Mbps$, $N = 20$, $R = 100ms$, $P = 4.4s$ and (a) $B_s = 50Mbps$ (b) $B_s = 100Mbps$, for AVC MSS and SVC Cursor in combination with sequential and parallel scheduling.

transfers to attain the same quality as with AVC based HAS and 2) the increased request-response rate in SVC-based HAS, which as a result of the high RTT's, leads to inefficient use of the available bandwidth. This paper proposes to overcome the second issue by using HTTP Pipelined and Parallel download schedulers, eliminating the idle time between two consecutive downloads. Next to the schedulers, we also propose a cursor based SVC client heuristic, which outperforms existing SVC-based heuristics for small buffer sizes. The experimental results have shown that even with small buffer sizes, the combination of SVC Cursor with parallel scheduling is not only able to overcome the issues in high delay networks, but is even capable of achieving higher quality with less frequent switches than AVC-based HAS. For a buffer size of 3 segments, parallel and pipelined scheduling improve the quality with about 14%, while reducing the number of switches with a factor 4 compared to the sequential scheduling. Hence, the combination of our novel SVC Cursor heuristic in combination with parallel download scheduling is able to outperform the state of the art heuristics by alleviating the drawbacks of SVC-based HAS while retaining the higher decision granularity and other advantages.

ACKNOWLEDGMENT

Niels Bouten is funded by a Ph.D. grant of the Agency for Innovation by Science and Technology (IWT). This research was partially performed within the iMinds MISTRAL project (under grant agreement no. 10838). Alcatel-Lucent was partially funded by IWT project 110112.

REFERENCES

[1] J. Famaey, S. Latré, N. Bouten, W. Van de Meerssche, B. De Vleeschauwer, W. Van Leekwijck, and F. De Turck, "On the merits of SVC-based HTTP adaptive streaming," in *Proceedings of the seventh IFIP/IEEE International Symposium on Integrated Network Management*, may 2013.

[2] R. Huysegems, B. De Vleeschauwer, T. Wu, and W. Van Leekwijck, "SVC-based HTTP adaptive streaming," *Bell Labs Technical Journal*, vol. 16, no. 4, pp. 25–41, 2012.

[3] Microsoft, "Smooth streaming: The official microsoft IIS site," http://www.iis.net/download/SmoothStreaming.

[4] R. Pantos and W. May, "HTTP Live Streaming," 2012. [Online]. Available: http://tools.ietf.org/html/draft-pantos-http-live-streaming-10

[5] Adobe, "HTTP dynamic streaming: Flexible delivery of on-demand and live video streaming," http://www.adobe.com/products/httpdynamicstreaming/.

[6] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 133–144.

[7] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," in *IEEE Transactions on Circuits and Systems for Video Technology In Circuits and Systems for Video Technology*, 2007, pp. 1103–1120.

[8] H. Choi, J. Nam, D. Sim, and I. Bajic, "Scalable video coding based on high efficiency video coding (hevc)," in *Communications, Computers and Signal Processing (PacRim), 2011*, aug. 2011, pp. 346 –351.

[9] C. Liu, I. Bouazizi, and M. Gabbouj, "Parallel adaptive HTTP media streaming," in *Proceedings of 20th International Conference on Computer Communications and Networks*, 31 2011-aug. 4 2011, pp. 1 –6.

[10] N. Bouten, J. Famaey, S. Latré, R. Huysegems, B. De Vleeschauwer, W. Van Leekwijck, and F. De Turck, "QoE optimization through in-network quality adaptation for HTTP adaptive streaming," in *8th International Conference on Network and Service Management (CNSM)*. IEEE, 2012, pp. 336–342.

[11] N. Bouten, S. Latré, W. Meerssche, B. Vleeschauwer, K. Schepper, W. Leekwijck, and F. Turck, "A multicast-enabled delivery framework for QoE assurance of over-the-top services in multimedia access networks," *Journal of Network and Systems Management*, pp. 1–30, 2013.

[12] N. Bouten, S. Latré, W. Van De Meerssche, K. De Schepper, B. De Vleeschauwer, W. Van Leekwijck, and F. De Turck, "An autonomic delivery framework for HTTP adaptive streaming in multicast-enabled multimedia access networks," in *Fifth IFIP/IEEE Workshop on Distributed Autonomous Network Management Systems (DANMS), 2012*. IEEE, 2012, pp. 1248–1253.

[13] S. Akhshabi, A. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," *ACM MMSys*, vol. 11, pp. 157–168, 2011.

[14] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj, "Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 288 – 311, 2012.

[15] V. Adzic, H. Kalva, and B. Furht, "Optimized adaptive HTTP streaming for mobile devices," in *SPIE Optical Engineering+ Applications*. International Society for Optics and Photonics, 2011, pp. 81 350T–81 350T.

[16] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE," *Technical report, Carnegie Mellon University*, 2012.

[17] T. Schierl, Y. Sanchez de la Fuente, R. Globisch, C. Hellge, and T. Wiegand, "Priority-based media delivery using SVC with RTP and HTTP streaming," *Multimedia Tools and Applications*, vol. 55, pp. 227–246, 2011.

[18] T. Andelin, V. Chetty, D. Harbaugh, S. Warnick, and D. Zappala, "Quality selection for dynamic adaptive streaming over HTTP with scalable video coding," in *Proceedings of the 3rd Multimedia Systems Conference*. ACM, 2012, pp. 149–154.

[19] D. Kaspar, K. Evensen, P. Engelstad, and A. Hansen, "Using HTTP pipelining to improve progressive download over multiple heterogeneous interfaces," in *IEEE International Conference on Communications (ICC), 2010*. IEEE, 2010, pp. 1–5.