Why Is HTTP Adaptive Streaming So Hard?

Sangwook Bae Dahyun Jang KyoungSoo Park

Department of Electrical Engineering, KAIST
Daejeon, South Korea

hoops@ndsl.kaist.edu jangdh93@kaist.ac.kr kyoungsoo@ee.kaist.ac.kr

Abstract

HTTP adaptive streaming is increasingly popular in video delivery. This is mainly because HTTP allows easy deployment while it simplifies content delivery, and chunk-based delivery enables dynamic adaptation of video quality to varying network bandwidth. However, we find that the very nature of chunk delivery on HTTP causes some fundamental problems in efficient bandwidth utilization.

In this work, we investigate why it is so hard to adapt to varying bandwidth with HTTP adaptive streaming. First, we find that the choice of chunk duration greatly affects the bandwidth adaptation logic. Second, we observe that the disparity between the advertised quality of a chunk and real encoding rate confuses the client-side adaptation logic. Third, the dependence on TCP/HTTP leads to suboptimal bandwidth utilization while it makes it challenging to adapt to rapidly-changing bandwidth. We show the evidence of the problems in our controlled experiments with popular HTTP adaptive streaming schemes, and lay out the future requirements for robust bandwidth adaptation in video streaming.

General Terms Algorithm, Design, Performance

Keywords HTTP adaptive streaming, DASH, MPEG Media Transport

1. Introduction

HTTP adaptive streaming (HAS) is increasingly popular in online video delivery. While there are many variations of HAS implementations [5, 9, 29], they all share the similar operational model. In HAS, a large video content is divided into smaller fragments called chunks, and a video client plays it by fetching the chunks in the chronological order from a Web server that serves the video. Typically, each video chunk is encoded into multiple different qualities, which allows the client to choose the proper video rate to adapt to varying network bandwidth in real time. The simplicity and ubiquity of HTTP make it extremely easy to provision, deploy, and operate the service.

Despite many advantages, previous works have reported undesirable behaviors that affect user's quality-of-experience (QoE) [11, 18, 20]. When multiple HAS clients compete on a bottlenecked link, it is reported that the video quality often changes too frequently or some clients would get an unfair bandwidth share while the bandwidth utilization becomes suboptimal. Most of these problems happen because the clients repeatedly go between downloading and pause phases (called ON and OFF periods), which confuses other competing clients about their fair network bandwidth share. To address these problems, some researchers try removing the ON/OFF periods with the help of the server [12, 17], while others develop more accurate bandwidth estimation algorithms [20, 30].

In this work, we report a different class of problems that affect the client-side bandwidth adaptation logic. First, we observe that the choice of chunk duration (e.g., how much time each chunk content should carry) has a great impact on the accuracy of bandwidth adaptation. We find that a short duration could lead to suboptimal bandwidth estimation while a long duration often makes it difficult to adapt to fast-changing bandwidth. Second, we note that the advertised encoding rate of a chunk and its real rate are often drastically different in practice. A lack of such information on the client side produces a completely undesirable behavior in rate selection. Third, the dependence on TCP/HTTP creates a problem in bandwidth utilization and adaptation. We observe that the oscillation of ON/OFF periods makes TCP severely underutilize the full network bandwidth. Also, the rigidity of TCP reliable data transmission prevents fast rate adaptation for varying network bandwidth.

Unlike existing works, we note that these are fundamental problems, directly related to the HAS operational model. The root cause of these problems lies in that (a) the client cannot adapt to abrupt bandwidth change during a chunk download, (b) constant bit rate (CBR) encoding is hard to be accurately enforced on a chunk granularity, (c) HTTP/TCP

APSys 2015, July 27–28, 2015, Tokyo, Japan.
Copyright © 2015 ACM 978-1-4503-3554-6/15/07…$15.00.
http://dx.doi.org/10.1145/http://dx.doi.org/10.1145/2797022.2797031

HAS solution	Player	Version
Smooth streaming	Sliverlight	5.1.30514.0
OSMF	Stobe media player	10.1.102.64
DASH	bitdash	1.2.9

Table 1. Players used in our experiments

are not the ideal transport protocols for efficient bandwidth utilization and fine-grain rate adaptation for video streaming. In this paper, we quantify the problems with three popular HAS implementations such as Smooth Streaming [9], OSMF [8], and bitdash [2], and gauge the gap between HAS and ideal video streaming.

To address these problems, we also consider the future requirements for ideal video streaming. We argue that future video streaming should reflect the difference in the advertised and real encoding rates in bandwidth adaptation. Also, it should allow agility in video rate change to timely respond to abrupt bandwidth change. Finally, it should avoid the OFF period for fair and consistent bandwidth consumption. To realize these requirements, we experiment with an alternative streaming approach called MPEG Media Transport (MMT) [7], and show preliminary results.

2. Why Is HTTP Adaptive Streaming Difficult?

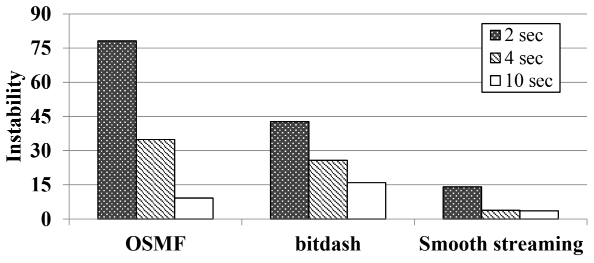
In this section, we present our findings about the difficulty in HTTP adaptive video streaming. First, we discuss the impact of a chunk duration on bandwidth estimation and video quality adaptation, and then describe the problems arising from the disparity of advertised and real video encoding rates. Finally, we reason about why TCP/HTTP leads to suboptimal bandwidth utilization and poor rate adaptation in dynamic network environments. For all experiments, we use a machine with a quad core Intel CPU (i7-2600) with hyper-threading on with 12 GB of physical memory as a server and a client. We use Windows 7 for serving OSMF and Smooth Streaming contents while we run nginx 1.7.8 on Linux (Ubuntu 12.04) for DASH streaming. For limiting the bandwidth, we employ DummyNet [27] on Linux (Ubuntu 12.04) on a machine with the same specification.

2.1 Impact of Chunk Duration

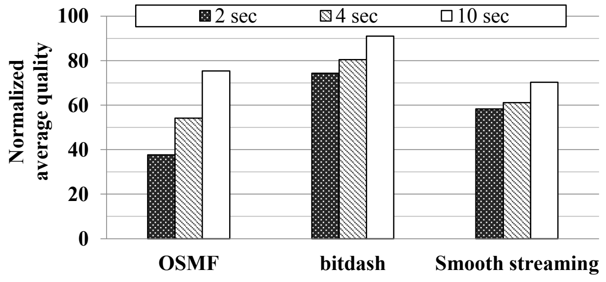
A chunk duration in HAS is typically fixed over the entire streaming period. Publicly recommended chunk durations range from 2 to 10 seconds [6, 10]. Regardless of the actual value, however, we find that the chunk duration often adversely affects the quality of bandwidth estimation and agility in video rate adaptation.

To investigate the impact of a chunk duration on user QoE, we run a few experiments in a controlled environment. We prepare multiple video streams of BigBuckBunny [1] [1], encoded to different bitrates as shown in Table 2. We compare the behavior with three HAS implementations as shown in Table 1. To simulate a highly-dynamic network environment, we configure the available bandwidth to oscillate between 700 Kbps and 4.5 Mbps in every 3 seconds via DummyNet. We measure two popular QoE metrics from [31], instability and average quality of received chunks, and use 2, 4, and 10 seconds as chunk durations. Instability is the sum of the ratios of video quality change between consecutive chunks, and a larger value indicates worse QoE. In contrast, a larger average quality implies better QoE.

[1] In case of DASH streaming, we use the dataset used in [23].



(a) Instability over various HAS implementations



(b) Normalized average quality over various HAS implementations

Figure 1. Impact of chunk duration

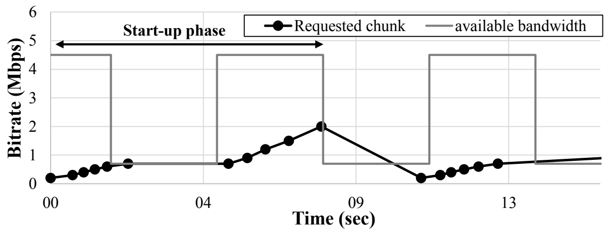


Figure 2. Chunk request behavior with short chunk duration

$$\text{Instability:} \quad \sum_{i=1}^{K-1} \frac{|B_i - B_{i+1}|}{Max(B_i, B_{i+1})} \tag{1}$$

$$\text{AverageQuality:} \quad \frac{1}{K} \sum_{i=1}^{K} B_i \tag{2}$$

K is the total # of received chunks,

B_i is the bitrate of the i-th chunk.

Figure 1 compares instability and normalized average quality over different chunk durations. For comparing with ideal case, we normalize the average quality by the ideal

	Codec	fps	Bitrate (Mbps) @(Resoultion))	Length
OSMF	H.264	24	0.2@(480x360), 0.3@(480x360), 0.4@(480x360), 0.5@(480x360), 0.6@(854x480), 0.7@(854x480), 0.9@(1280x720), 1.2@(1280x720), 1.5@(1280x720), 2.0@(1280x720), 2.5@(1920x1080), 3.0@(1920x1080), 4.0@(1920x1080)	9:57
Smooth Streaming	VC-1			
DASH	H.264			

Table 2. Media statistics for evaluating three HAS method

Chunk	Rebuffering event			Time to converge (sec)		
	OSMF	DASH	SS	OSMF	DASH	SS
2 sec	No	No	No	9	21	55
4 sec	No	No	Yes	25	23	110
10 sec	Yes	Yes	Yes	63	52	304

SS: Smooth Streaming

Table 3. Effect of long chunk duration to three HAS protocol

rate selection. In general, both instability and average quality improve as the chunk duration increases. Especially, bitdash and OSMF clients show 2.7 and 8.7 times improvement in instability respectively when the chunk duration is switched from 2 to 10 seconds. This is mainly because a long chunk duration enables more accurate bandwidth estimation. A short chunk duration produces a smaller chunk, and the chunk download finishes more quickly than a longer chunk. This would miss the opportunity to accurately probe the dynamically-changing network bandwidth, which leads to suboptimal bandwidth estimation. Smooth streaming shows a reasonably stable rate selection behavior regardless of chunk duration since its bandwidth estimation logic is more conservative. However, compared with the ideal rate selection (e.g., 2.5 Mbps), its bandwidth utilization is lower by 30 to 42%.

Figure 2 shows the client-side rate selection behavior with OSMF in more detail when the chunk duration is fixed to 2 seconds. We observe that the client mostly chooses low-quality chunks even after the start-up phase because the downloading higher-quality chunk starts when the available bandwidth is low. At around the 11-th chunk, the client attempts to download a higher-quality chunk, but its download period overlaps with a low bandwidth period. So, the client moves to the lowest quality in the next round to offset the deficit from the previous download. A short chunk duration hinders accurate bandwidth estimation, and increases the oscillation in bitrate choice, which degrades the stability and average quality.

The next question is whether a long chunk duration is always desirable in HAS. While a long duration allows more accurate bandwidth estimation, it greatly reduces the agility in rate adaptation. Even when the available bandwidth abruptly changes, a long chunk duration makes it difficult to react to it until the chunk download finishes. Note that timely response to fast-changing bandwidth is the key to minimizing rebuffering events that significantly undermine user QoE. To see the impact of a long chunk duration, we simulate a dynamic network environment by starting with 3 Mbps as available bandwidth and dropping it to 0.7 Mbps in the middle of streaming. Table 3 shows the results with three different chunk durations. We observe that the largest chunk duration produces a rebuffering event in all clients. Moreover, the time to adapt to the new bandwidth becomes much larger in a longer chunk duration. This is because they need to reduce the video quality to refill the drained buffer and to probe the accurate bandwidth.

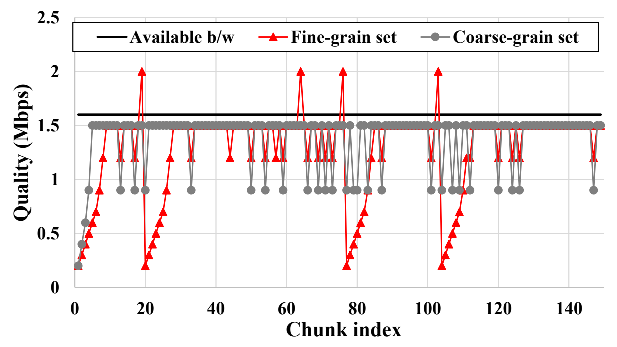
In summary, neither short nor long chunk duration always produces good user QoE. A proper choice would highly depend on the characteristics of network environments, which is often hard to predict. A short duration could incorrectly estimate the network bandwidth and incur more overhead from frequent HTTP requests. A long duration could be too slow to adapt to sudden bandwidth drop, and could increase the rebuffering events.

We think this is a fundamental problem in HAS since the core difficulty is attributed to chunk-based downloading and the adoption of inflexible transport protocols. The fact that the HAS client cannot switch to a different video quality during a chunk download is the source of the problem. An ideal approach would be to have a long chunk duration for accurate estimation of available bandwidth while providing the flexibility that a client can switch to a different video quality at any time when it senses abrupt bandwidth change. One can abort the current chunk download and switch to a different quality via HTTP range queries. However, this would make the rate adaptation highly complex since the client now needs to know which byte offset would seamlessly link to the last frame of the previously-interrupted chunk.

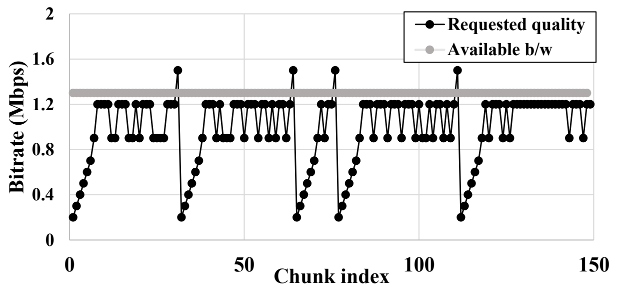
2.2 Impact of Bitrate Disparity

It is commonly known that a lager encoding set size produces better QoE in HAS. An encoding set size is referred to the number of different encoding qualities to serve a given video content. A larger encoding set size allows a client to choose the video quality from a bigger pool of bitrates, which enables more fine-grain rate adaptation that effectively utilizes the available bandwidth. Also, it would prevent sudden bitrate oscillation, improving the QoE for the end users.

To verify this claim, we run an experiment with the same video content (e.g., Big Buck Bunny) with different encoding set sizes. We prepare a coarse-grain encoding set with 7 distinct bitrates (0.2, 0.4, 0.6, 0.9, 1.5, 2.5, 4 Mbps) and a fine-grain set that adds 6 more bitrates (0.3, 0.5, 0.7, 1.2, 2.0, 3.0 Mbps) to the coarse-grain set. We download the video



(a) OSMF client at 1.6 Mbps bandwidth limit



(b) OSMF client with fine-grain set at 1.3 Mbps bandwidth limit

Figure 3. Counterexample of common sense about granularity of encoding set

with an OSMF client and record the stream of chunk bitrates requested by the client. The available bandwidth is fixed to 1.6 Mbps throughout the experiment.

Figure 3(a) shows the results. Surprisingly, we find that the coarse-grain encoding set provides better bandwidth adaptation than that of the fine-grain set. Chunk bitrates for the fine-grain encoding set oscillate more widely, often picking the chunks in much lower quality than desired. Also, the average video quality of the fine-grain set is lower (1.28 Mbps) than that of the coarse-grain set (1.37 Mbps). Figure 3(b) shows the similar behavior of the fine-grain set when the bandwidth is set to 1.3 Mbps. We expected that the client would choose the chunks encoded in 1.2 Mbps, but the actual selection varies widely between 0.3 and 1.5 Mbps.

After close investigation, we find out that the advertised bitrates by the manifest file and the actual bitrates of the chosen chunks are often drastically different. We find that the actual bitrate varies greatly, from 92% to 193% of the advertised bitrate as shown in Table 4. To see whether this is a corner case, we also check with four other contents from the public DASH video dataset [23], but we observe the similar trend. We see that the actual encoding rate goes as small as 11% or as large as 321% of the advertised rate. We go on to check with the commercial video contents of 13 hours of HLS traffic by Btv [3], a popular online streaming

Big Buck Bunny used for OSMF [2]				
Advertised Bitrate (Kbps)	400	1200	2500	4000
Real to adv. ratio (%)	93 - 172	95 - 152	92 - 181	117 - 193
Big Buck Bunny used for Smooth Streaming [3]				
Advertised Bitrate (Kbps)	400	1200	2500	4000
Real to adv. ratio (%)	59 - 192	80 - 173	67 - 192	64 - 195
Big Buck Bunny in DASH dataset				
Advertised Bitrate (Kbps)	378	1207	2410	3936
Real to adv. ratio (%)	11 - 119	39 - 321	11 - 114	25 -210
Elephants Dream in DASH dataset				
Advertised Bitrate (Kbps)	379	1197	2394	4066
Real to adv. ratio (%)	14 - 139	13 -131	20 - 130	8 - 207
Tears of Steel in DASH dataset				
Advertised Bitrate (Kbps)	504	1506	2416	4011
Real to adv. ratio (%)	13 - 230	13 - 231	17 - 231	16 - 228
HLS traffic from Btv Mobile				
Advertised Bitrate (Kbps)	-	1000	-	-
Real to adv. ratio (%)	-	57 - 158	-	-

Table 4. Real to advertised encoding bitrate ratios

service in South Korea. We confirm that the actual bitrates do not match the advertised rates in many cases, taking up 57% to 157% of the advertised rates.

This bitrate disparity has crucial impact on client-side rate selection since the client has to resort to the bitrate information exposed by the manifest file. If the actual bitrate is larger than the advertised bitrate, the client must choose a lower quality than the available bandwidth in the next round to offset the buffer drain. The situation could be worse for buffer-aware rate adaptation algorithms [13,30] that consider buffer occupancy for rate selection. If the actual rate is lower than the advertised rate, such an algorithm would choose a higher rate in the next round only to find that the available bandwidth is lower than the requested bitrate. Then, it would choose a lower quality to compensate the buffer loss, which aggravates the oscillation.

Why is there disparity between the advertised and real encoding rates even if the video chunks are encoded in constant bit rate (CBR)? The short answer is that CBR encoding is hard to be enforced on a small chunk granularity. Uniform CBR encoding is challenging without significant quality degradation or redundant video frames since some video frames could be much busier than others. However, most manifest files in HAS advertise only the nominal encoding rate averaged over the entire content while the real rate could vary on a chunk basis. The lack of the information on the client side confuses the rate adaptation logic.

We note that [19] considers the bitrate disparity problem, but our focus here is that the client-side adaptation is the source of the complexity to address the problem. For correct operation, the client would require chunk size information for all video qualities. However, fetching such metadata for all video chunks would be an overhead since the number of different video qualities could be large, and most of the per-chunk metadata downloading would be wasted. Also, when it comes to live streaming, such additional metadata download would incur extra latency. These overheads imply

[2] Encoded by Adobe Media Encoder 8.2.0

[3] Encoded by Microsoft Expression 4 (version 4.0.1460.0)

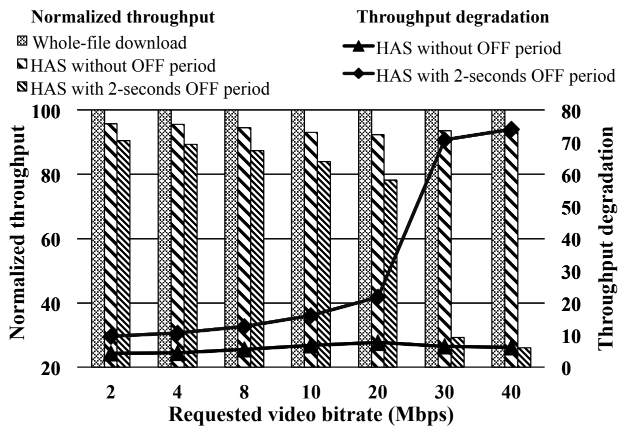


Figure 4. Normalized throughput and throughput degradation of three download methods over various available network bandwidth environments

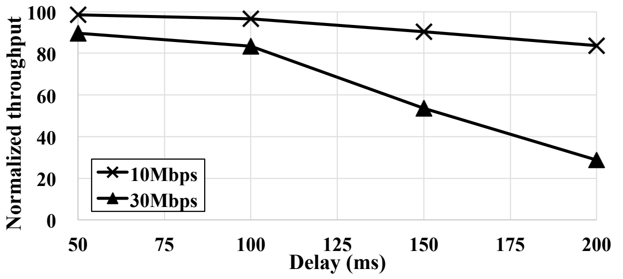


Figure 5. Normalized throughput of HAS downloading 10Mbps (HD) and 30Mbps (UHD) chunks with 2 seconds gap over various delay environment

that handling the bitrate disparity problem at client side is complex and inefficient.

2.3 Impact of Transport Protocols

In contrast to traditional streaming approaches that use UDP (RTP on top of UDP or DCCP [21]) as the transport layer protocol, HAS adopts TCP and HTTP for adaptive media streaming. It is well-known that the choice of TCP/HTTP trades fast bandwidth adaptation for operational simplicity and easy deployment. That is, TCP is known to be inflexible to quickly adapt to varying network bandwidth due to its reliable data transfer requirement. So, we will not focus on the lack of agility from TCP retransmission. Our question here is if TCP/HTTP is still a good choice if the available bandwidth is stable over time. How much overhead does chunk-based downloading incur in typical video streaming?

To answer this question, we run an experiment in a controlled environment. A client downloads a 400-second video whose chunk size is set to 4 seconds. So, there are 100 video chunks, and we make sure that each chunk encoded to the same bitrate has the exactly same size to avoid the adverse effect from variable bitrate encoding. We also make the available bandwidth stable over time so that there is no need to adapt to changing bandwidth. We fill the client-side buffer before each experiment so that the client can skip the start phase that fills up the buffer with lower-quality chunks than desired for the bandwidth. Our expectation is that the client downloads the video chunks of the same quality from start to end. We test three scenarios. First, the client downloads the video chunks after two seconds of delay between each download. This simulates the well-known OFF period in HAS. Second, the client downloads the chunks back to back without any idle period between the downloads. This scenario measures the HTTP transaction overhead. Third, the client downloads the entire file with one HTTP request. This is similar to traditional streaming, but it uses TCP as the transport layer protocol. This serves as the base case for performance comparison. For all experiments, we use a persistent connection to download all chunks in a single TCP connection.

Figure 4 shows the relative throughputs normalized by the base case performance over various available network bandwidths. We mark the throughput of the base case as 100%. Overall, we see a significant overhead in using HTTP/TCP along with chunk-based downloading. Having the two-second OFF period between chunk downloads degrades 10% to 74% of the base case throughputs, and even without the OFF period, the HTTP transaction alone reduces the performance by 4% to 8%. Having an OFF period is detrimental to the throughput since it initiates TCP slow start for every chunk download. As the bandwidth-delay product (BDP) gets larger, the throughput degradation is more noticeable since the TCP connection severely underutilizes the pipe due to repeated TCP slow start. Moreover, the problem may get worse in the near future. Given that the higher-quality video encoding such as 4K requires 30 to 40 Mbps [4] of bandwidth for streaming, the TCP overhead in HAS will remain a significant problem. While [15] also discusses the TCP behavior on video streaming, we quantify the impact of the OFF period in various network environments.

Figure 5 shows the normalized throughputs of an HAS client with an OFF period over various delays between a client and a server. We choose 10 and 30 Mbps as available bandwidths required for HD and UHD videos. As the delay grows, the performance significantly degrades from the base case since BDP increases, but even at 50 ms, the UHD video shows as much as 10% performance reduction from the base case throughput.

3. Requirements for Future Media Streaming

We summarize three key capabilities for future adaptive streaming; (a) fast and fine-grain rate adaptation, (b) media transmission without the OFF period, and (c) reflecting the real encoding information into rate selection. Unfortunately, as the experiments in Section 2 imply, we find that these requirements are often in conflict with the current HAS operational model. Table 5 shows the requirements for desirable rate adaptation and the current limitations on the client side. Client-driven rate adaptation could lead to suboptimal

Requirements	Limitations on client-driven streaming	Possible solutions in server-driven streaming
Agile and fine-grain rate adaptation	• Chunk-nature operating model • Inflexible transport protocol	• Easy to implement with characteristics of MMT • DCCP-based transmission
Transmission without the OFF period	• Chunk-nature operating model • Client cannot control the sending rate	• Server controls the sending rate
Reflecting the encoding information	• Extra delay for metadata requests • Waste of network resource	• Server has media encoding information
Avoid overheads from transport protocol	• Possible slow start at every request • Transaction overhead at every request	• No additional slow-start • No additional transactions

Table 5. Comparison of two streaming systems

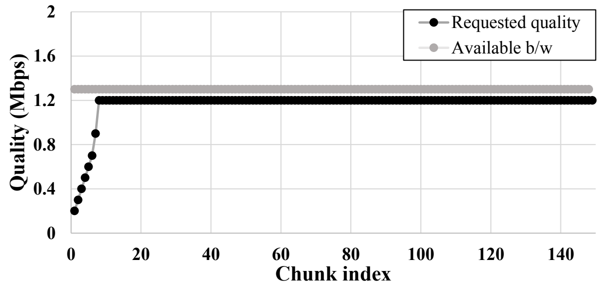
rate selection due to a lack of real encoding information, and the reliable chunk delivery model limits efficient network resource utilization. Moreover, the chunk-based rate selection often prevents agile adaptation and can generate ON-OFF periods and HTTP transaction overheads for chunk requests.

In this section, we open our mind to address these deficiencies in alternative streaming approaches and discuss the trade-offs. Table 5 lays out the advantages of server-driven streaming in achieving the requirements.

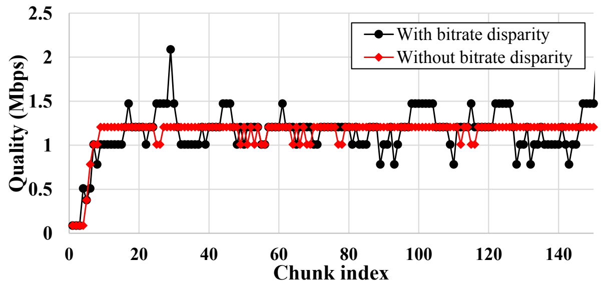
Revisiting server-driven streaming with MMT Is server-driven rate adaptation really a bad idea? One argument for client-driven video delivery is that it is more scalable since it would significantly reduce the load on the server side. But the proliferation of multicore CPUs and high memory capacity on the server side and the rapid development of cloud-based streaming service technology could alleviate the issue. On the other hand, UDP-based server-driven video delivery could face a serious deployment issue especially due to wide adoption of network middleboxes. However, we argue that at least the streaming services in controlled environments like Telco-driven IPTV or video-on-demand services could benefit more from the server-driven approach for better resource utilization.

We briefly discuss a new video streaming standard called MPEG Media Transport (MMT) [7], which addresses some of the challenges in existing server-driven streaming protocols (e,g. RTP/RTSP) such as identifying the multiple sources, and overhead from mismatch between the storage and delivery media formats [24, 25]. An MMT media also consists of multiple video fragments called media processing units (MPUs), and the MPUs are composed of smaller media fragmented units (MFUs) that are decoded independently (e.g., Network Abstraction Layer (NAL) unit in H.264). The key difference from HAS is that the MMT server mainly delivers the media on UDP with a standardized packet structure defined by the MMT Protocol (MMTP). The MMTP packet structure allows efficient packetizing and fine-grain adaptation because it is agnostic to the media codec and it encodes only one MFU in each packet.

Utilizing media bitrate information In Section 2.2, we have shown that reflecting the real encoding rate to rate selection is critical for improving the user QoE. One solution



(a) Quality selection of OSMF client without the bitrate disparity



(b) Quality selection of DASH client

Figure 6. Impact of managing bitrate disparity in fine-grain encoding set which is used in Section 2.2

is to provide the real video encoding rate to the client. The challenge here is that if the encoding set size is large, delivering such metadata not only wastes network bandwidth but complicates the client side logic. In addition, in live streaming, downloading such metadata could produce extra latency, which might interfere with real-time streaming. In contrast, a server-driven approach could utilize the video encoding information more efficiently since the server already has the information.

To show the benefit of utilizing the media encoding information, we run experiments with two HAS clients (OSMF and DASH) at a fixed bandwidth of 1.3 Mbps. To simulate an ideal environment, we enforce the same bitrates for advertised and real encoding rates over all chunks. Figure 6 shows the results, which confirms that removing the bitrate disparity substantially improves the QoE by avoiding undesirable oscillation. Figure 6(a) shows that the OSMF client chooses the proper rate all the time, in contrast to the behavior in Fig-

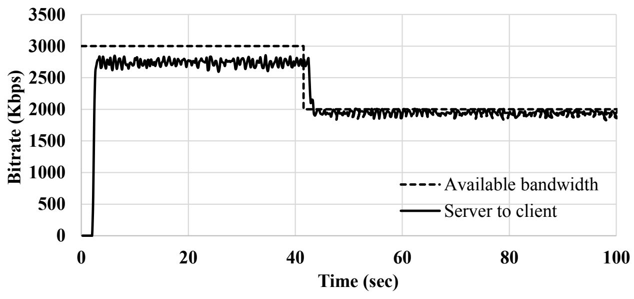


Figure 7. Agile adaptation by MMT with the video of 2,694Kbps bitrate as encoding rate

ure 3(b). Figure 6(b) shows that the bitrate mismatch leads to higher instability (12.3) even in DASH, 3.3 times larger than that of having the correct knowledge of real bitrate (3.7).

Agile adaptation & streaming w/o the OFF period As shown earlier, the chunk-based downloading in HTTP makes it difficult to adapt to sudden bandwidth change, and removing the OFF period is often hard without the help of the server or a middlebox. Prior works try mitigating the effect of the OFF period in chunk rate selection by scheduling chunk requests [20], or using a feedback control [14, 30, 32] rather than removing the OFF periods.

In contrast, a server-driven approach with MMT would make it easy for agile adaptation of video quality. Because MFUs in an MMT medium can be easily dropped by the server to fast respond to sudden network congestion without interrupting video playback, the server can adjust the video quality closely to varying bandwidth. On the other hand, certain information such as the level of buffer occupancy and real-time download throughput is known only to the client. Therefore, we can maximize the QoE by having the client notify such status information to the server, and by having the server dynamically change the sending rate and video quality by dropping the MFUs (or frames).

The server-driven approach also provides the benefit of controlling the sending rate, and the server can remove the OFF period easily by limiting the rate to the highest bitrate of the media. Moreover, the server could use a UDP-based congestion control algorithm such as TCP-friendly rate control (TFRC) [16]. TFRC provides the fairness among other flows while it allows smooth switch in the sending rates even when the available bandwidth abruptly changes.

To show the potential benefit of a server-driven approach with MMT for agile adaptation, we implement a basic MMT server and a client that periodically reports the measured throughput to the server. The MMT server continuously sends the media without the OFF period and adapts its sending rate to the available bandwidth. It also reflects the real encoding rate to its sending rate. Figure 7 shows the behavior of our preliminary system when the available bandwidth drops from 3 Mbps to 2 Mbps. We observe that the server adapts the rate fast to sudden bandwidth drop. Actually, even if some MFU could are lost due to congestion, it will not affect the playback much because they could be decoded separately. We do see the saw-tooth behavior in bandwidth utilization because each GoP (Group of Pictures) has a different size even with CBR encoding, but the level of oscillation is much smaller than those in Figure 3 because of server-side adaptation.

Fine-grain adaptation For fine-grain rate adaptation, one might consider scalable video coding (SVC) [22, 26, 28]. SVC encodes a video into a new bitstream that has several layers of subset bitstreams. By removing each layer, the server can generate multiple bitstreams of a lower spatial or temporal resolution from one bitstream. In addition, more fine-grain encoding rates can be presented to the client by selectively dropping the B-or P-frames that do not affect the media playback. While SVC is promising, it could be more easily integrated into the server side rather than the client-based HAS model.

4. Conclusion and Future work

HAS is widely used for online video delivery due to its easy deployment and simplicity. However, we have identified some of the fundamental problems that are directly related to the HAS operational model. These include suboptimal bandwidth estimation tied to the choice of chunk duration, poor bitrate selection due to lack of the real encoding rate, and inefficient network bandwidth utilization due to dependence of rigid transport protocols. To address these problems, we lay out requirements for future streaming, and revisit the idea of server-driven media streaming with MPEG Media Transport. Barring the deployment barrier, we find that the server-driven architecture is attractive in solving the problems. In the future, we plan to fully implement our system and compare it with HAS-based approaches in various situations.

5. Acknowledgments

We would like to thank the anonymous reviewers of APSys 2015 and our shepherd, Jianfeng Zhan, for their insightful comments and suggestions to improve the paper. This research was supported by Samsung Electronics [IO150406-02858-01, Development of efficient real-time media delivery] and the ICT R&D program of MSIP/IITP, Republic of Korea [14-911-05-001, Development of an NFV-inspired networked switch and an operating system for multi-middlebox services].

References

[1] BigBuckBunny. `http://www.bigbuckbunny.org/`.

[2] bitdash mpeg-dash player. `http://www.bitmovin.net/bitdash-mpeg-dash-player/`.

[3] Btv Mobile. `http://www.skbtv.co.kr/btvmobile/`.

[4] H.265/HEVC Ratification and 4K Video Streaming. `http://blogs.iis.net/alexzam/h-265-hevc-ratification-and-4k-video-streaming/`.

[5] HTTP Live Streaming. `https://developer.apple.com/streaming/`.

[6] IIS Smooth Streaming Technical Overview. http://www.microsoft.com/en-us/download/details.aspx?id=17678.

[7] ISO/IEC 23008-1:2014, Information technology - High efficiency coding and media delivery in heterogeneous environments - Part 1: MPEG media transport (MMT) .

[8] Open Source Media Framework. http://blogs.adobe.com/osmf/.

[9] Smooth Streaming. http://www.iis.net/downloads/microsoft/smooth-streaming.

[10] Technical Note TN2224. https://developer.apple.com/library/ios/technotes/tn2224/_index.html.

[11] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis. What happens when http adaptive streaming players compete for bandwidth? In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop (NOSSDAV)*, 2012.

[12] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen. Server-based traffic shaping for stabilizing oscillating adaptive streaming players. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop (NOSSDAV)*, 2013.

[13] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo. Elastic: a client-side controller for dynamic adaptive streaming over http (dash). In *Proceedings of the international Packet Video Workshop (PV)*, 2013.

[14] L. De Cicco, S. Mascolo, and V. Palmisano. Feedback control for adaptive live video streaming. In *Proceedings of the ACM Multimedia Systems Conference (MMSys)*, 2011.

[15] J. Esteban, S. A. Benno, A. Beck, Y. Guo, V. Hilt, and I. Rimac. Interactions between http adaptive streaming and tcp. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop (NOSSDAV)*, 2012.

[16] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *ACM SIGCOMM Computer Communication Review (CCR)*, 2000.

[17] R. Houdaille and S. Gouache. Shaping http adaptive streams for a better user experience. In *Proceedings of the ACM Multimedia Systems Conference (MMSys)*, 2012.

[18] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, timid, and unstable: picking a video streaming rate is hard. In *Proceedings of the ACM conference on Internet measurement conference (IMC)*, 2012.

[19] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *ACM SIGCOMM Computer Communication Review (CCR)*, 2014.

[20] J. Jiang, V. Sekar, and H. Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the international conference on Emerging networking experiments and technologies (CoNEXT)*, 2012.

[21] E. Kohler, M. Handley, and S. Floyd. Designing dccp: Congestion control without reliability. In *ACM SIGCOMM Computer Communication Review (CCR)*, 2006.

[22] C. Kreuzberger, D. Posch, and H. Hellwagner. A scalable video coding dataset and toolchain for dynamic adaptive streaming over http. In *Proceedings of the ACM Multimedia Systems Conference (MMSys)*, 2015.

[23] S. Lederer, C. Müller, and C. Timmerer. Dynamic adaptive streaming over http dataset. In *Proceedings of the ACM Multimedia Systems Conference (MMSys)*, 2012.

[24] Y. Lim. Mmt, new alternative to mpeg-2 ts and rtp. In *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2013.

[25] Y. Lim, K. Park, J. Y. Lee, S. Aoki, and G. Fernando. Mmt: An emerging mpeg standard for multimedia delivery over the internet. *MultiMedia, IEEE*, 2013.

[26] C. Muller, D. Renzi, S. Lederer, S. Battista, and C. Timmerer. Using scalable video coding for dynamic adaptive streaming over http in mobile environments. In *Proceedings of the IEEE European Signal Processing Conference (EUSIPCO)*, 2012.

[27] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. In *ACM SIGCOMM Computer Communication Review (CCR)*, 1997.

[28] Y. Sánchez de la Fuente, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, and Y. Le Louédec. idash: improved dynamic adaptive streaming over http using scalable video coding. In *Proceedings of the ACM Multimedia Systems Conference (MMSys)*, 2011.

[29] I. Sodagar. The mpeg-dash standard for multimedia streaming over the internet. *IEEE Multimedia*, (18):62–67, 2011.

[30] G. Tian and Y. Liu. Towards agile and smooth video adaptation in dynamic http streaming. In *Proceedings of the international conference on Emerging networking experiments and technologies (CoNEXT)*, 2012.

[31] X. Yin, V. Sekar, and B. Sinopoli. Toward a principled framework to design dynamic adaptive streaming algorithms over http. In *Proceedings of the ACM Workshop on Hot Topics in Networks (Hotnets)*, 2014.

[32] C. Zhou, X. Zhang, L. Huo, and Z. Guo. A control-theoretic approach to rate adaptation for dynamic http streaming. In *Visual Communications and Image Processing (VCIP), 2012 IEEE*, pages 1–6. IEEE, 2012.