A CASE FOR CACHING MIDDLEBOXES FOR SCALABLE MMT VIDEO DELIVERY

Sangwook Bae, Giyoung Nam, KyoungSoo Park

Department of Electrical Engineering, KAIST
{hoops, giyoung}@ndsl.kaist.ac.kr, kyoungsoo@ee.kaist.ac.kr

ABSTRACT

We present a content-based caching architecture for scalable MMT video delivery. Our architecture places a caching proxy in the path between an MMT server and its clients. Between an MMT server and a caching proxy, we exploit the popular HTTP protocol to reliably synchronize the video contents by their fragments while the caching proxy delivers the MMT asset to the clients via the standard MMTP protocol. This split architecture not only allows distributing the load on the MMT server but also supports MMT clients without any modification on the MMTP protocol. Furthermore, our architecture supports content-based caching with little run-time overhead, eliminating any redundant network transfers.

Index Terms— Content-based caching, MPEG Media Transport, Middlebox

1. INTRODUCTION

Efficient on-line video delivery is increasingly important as the demand for high-quality video streaming is rapidly grows [1]. One prominent approach is HTTP-based adaptive streaming (HAS) [2–4] where video contents are divided into smaller chunks and the clients fetch a stream of video chunks as HTTP requests. The HAS clients dynamically determine the video bit rate of each chunk by carefully estimating the available network bandwidth in real time. Popular HAS protocols include Apple's HLS [2] and Microsoft's Smooth Streaming [4], and MPEG's standardized Dynamic Adaptive Streaming over HTTP (DASH) [3].

Despite its growing popularity, HAS suffers from a number of drawbacks in practice. It is well-known that the rate adaptation in the application layer is often in conflict with TCP's own congestion control [5]. Also, repeated patterns of ON/OFF periods in chunk downloading creates unfair bandwidth allocation across competing HAS clients [6, 7]. Furthermore, the rate adaptation logic in the client often neglects the existence of transparent caching servers, which leads to overestimation of the available bandwidth and wrong choice of video chunk bit rate [8].

Meanwhile, MPEG has standardized an alternative video streaming technology called MPEG Media Transport (MMT) [9]. Unlike DASH, MMT adopts the traditional server-driven streaming model where a server selects a quality of video for each client. It primarily uses UDP, and defines the packet structure to suit the needs of fast adaptation of video quality to varying network environments. MMT is gaining its momentum especially with broadcasting companies [10, 11] that want to control over the on-line broadcasting videos.

While MMT overcomes a few limitations of DASH, the server-driven centralized architecture is generally construed as anathema to the network research community. The key challenge in MMT is the lack of server scalability as the number of clients grows. High computation overhead of rate adaptation for a large number of concurrent video streams often limits the capacity of a single server.

In this work, we address the scalability challenge with the MMT servers. Our basic idea is to exploit a number of MMT caching middleboxes (MMT-CMB) in the path between the clients and the server. Our caching middleboxes play two critical roles. First, they sit near to clients, and act as an MMT server to clients and perform accurate video rate adaptation by closely monitoring the level of network congestion in the last-mile link. MMT-CMBs would reduce the computation load from the server without any modification on the client software. Second, they efficiently use the server-side bandwidth by caching popular video chunks by their content. According to recent measurements [12, 13], up to 27% of popular YouTube contents are almost duplicates. Content-based caching would suppress the redundancy across similar video contents even though they are not identical.

We present the overall architecture of the MMT-CMB system and a detailed protocol in the paths among clients, MMT-CMBs, and MMT servers. We use the popular HTTP protocol between an MMT-CMB and an MMT server to exchange video chunks while the MMT-CMB runs MMT protocol (MMTP) with the client for video streaming. Our preliminary evaluation shows that the MMT-CMB system does reduce the bandwidth consumption at the server for cached contents while it accurately adapts the video rate to varying network bandwidth of the client.

The rest of paper is organized as follows. In Section 2, we provide the brief background of MMT and content-based caching. Section 3 shows our overall design of the MMT-CMB system and a detailed explanation of how MMT-CMB fetches, caches, and delivers the content. In Section 4, we de-

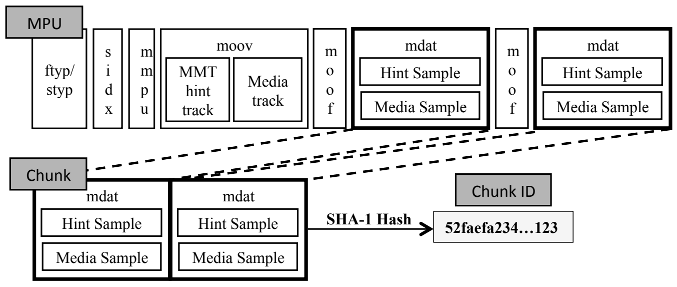

Fig. 1. Chunk generation for MMT content

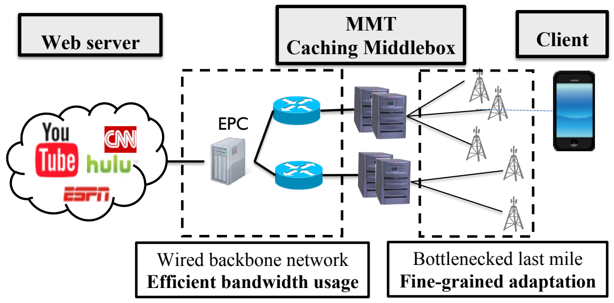

Fig. 2. Overview of MMT-CMB system

scribe our implementation and show preliminary evaluation, and conclude in Section 5.

2. BACKGROUND

In this section, we provide brief background on the MMT file structure and content-based caching.

2.1. Overview of MMT Content Structure

The MMT media format defines a logical structure called *package*, which consists of multiple *assets* and the composition information (CI). An asset represents logically-structured, encoded media data and the CI is used to help deliver the assets to clients. An MMT asset has multiple *media processing units (MPU)*s, which correspond to the chunks in DASH. As shown in figure 1, an MPU consists of a series of 'boxes' such as mmpu, moov, moof, and mdat boxes. The mmpu and moov boxes contain the metadata information about the asset or package while the moof and mdat boxes have the per-frame metadata in an MPU and the actual media content, respectively. When an MPU is being delivered to a client, it could be further fragmented into smaller units called *media fragmented unit(MFU)s*. An MFU contains a frame with its metadata. Since it is encoded independently, it can be dropped to adapt to the network congestion, allowing the server to enforce fine-grained rate adaptation.

2.2. Content-based Caching

Content-based caching caches an object by naming it as its content hash. It typically divides a large content into smaller 'chunks' using fixed or variable-sized chunking methods [14], and uses their content hash as a chunk name, which is essentially a tightly-bound summary of the chunk content. Using the chunk names, one can easily identify the same contents with different URLs (e.g., aliases) or even a set of partially-redundant chunks among similar but different contents [12, 15]. If a sender and a receiver supports content-based caching, the sender can transmit only the chunk names, and the receiver checks with the names if those chunks exist in its cache, and fetches only the cache missed chunks from the server. This would eliminate the redundant data transfer in the path between the sender and the receiver, and even saves the cache storage at the receiver by suppressing the redundancy.

3. DESIGN

In this section, we present the challenges in adopting content-based caching for MMT and describe our solutions. First, we talk about chunk generation issues for content-based caching, and then, we discuss some design issues about a content caching system and its protocols for MMT-CMB.

3.1. Caching Unit and Chunk Name Generation

An MPU would serve as an ideal caching unit for content-based caching. Unlike a variable chunking method like Rabin's fingerprinting, an MPU does not require any run-time overhead in determining the boundary since its boundary is determined at video encoding time by considering the video content. However, we cannot blindly name each MPU by its content hash, since MPUs could include different metadata (e.g., asset ID or MPU sequence number) even though they have the identical content.

To address this problem, we only hash the data in *mdat* box, the video or audio data itself, for chunk name generation. In details, we concatenate all *mdat* boxes in an MPU and run SHA-1 hash for the chunk name since there could be multiple *mdat* boxes in an MPU[1]. This approach would allow caching the same MPUs with different metadata as a single chunk, which would improve the redundancy suppression rate. Figure 1 shows the process our system generates the chunk and its name (chunk ID).

3.2. MMT Caching System

The MMT caching middlebox (MMT-CMB) system consists of three entities: a Web server that understands the MMT-CMB protocol (explained in Section 3.3), an MMT-CMB and a client. Figure 2 shows the overview of the MMT-CMB system design. The Web server can be any popular server as long as it understands the custom HTTP request headers (in Table 1) from an MMT-CMB. The MMT-CMB enforces content-based caching for efficient bandwidth usage, and dynamically adjusts the video rate to cope with the congestion in the last mile.

[1]The actual chunk name is calculated over 'payloadized' mdat sequences. In this way, (a) the middlebox reduces an undesirable delay since it can deliver each MFU promptly without waiting for the full MPU. (b) Payloadized mdat sequences would fix the location of the hint samples, which would produce the same name even if those hint samples are stored in different locations at a storage.

Format	Purpose
X-MPURange: N-M	Request MPU IDs from Nth MPU to Mth MPU
X-MPURequest: N	Request full content of Nth MPU
X-MPUHeaderRequest: N	Request only metadata part of Nth MPU
X-HTMLRequest	Request HTML file of Package
X-CIRequest	Request CI file of Package

Table 1. MMT-CMB protocol overview

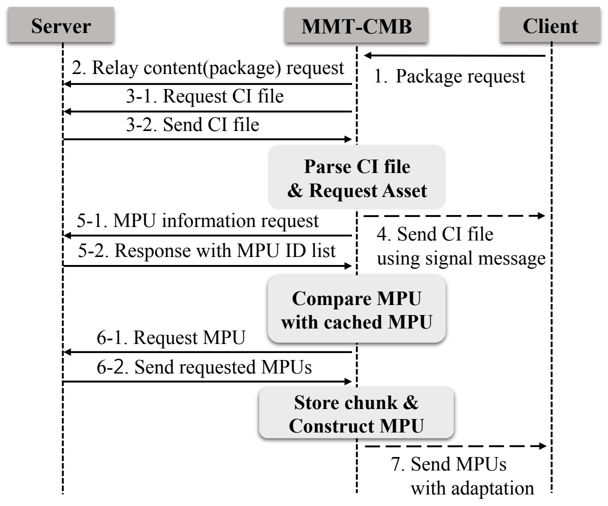
Server
MMT-CMB
Client
1. Package request
2. Relay content(package) request
3-1. Request CI file
3-2. Send CI file
Parse CI file & Request Asset
4. Send CI file using signal message
5-1. MPU information request
5-2. Response with MPU ID list
Compare MPU with cached MPU
6-1. Request MPU
6-2. Send requested MPUs
Store chunk & Construct MPU
7. Send MPUs with adaptation

Fig. 3. MMT-CMB work flow: Dashed arrows are MMTP and bold arrows are MMT-CMB protocol (HTTP)

The goal of the MMT-CMB is (a) to optimize the physical bandwidth usage in the path between the server and the MMT-CMB and (b) to adapt the video rate to available bandwidth per each client. For the former, the MMT-CMB focuses on content-based caching on the MPU delivery to increase the effective bandwidth and to reduce the load of MMT servers. An MMT-CMB fetches original MPUs (with the highest video quality) from the Web server only if they are a cache miss. The MMT-CMB fetches the highest quality MPUs from the server, but adapts its video quality to the network environments towards the clients. Our assumption is that the path between the Web server and the MMT-CMB has more bandwidth than that of the clients, but content-based caching would minimize any redundant MPU transfers.

For serving the media to clients, we locate the MMT-CMB close to the clients, and MMT-CMB can enforce a fine-grained rate control over the client-side network environments. The MMT-CMB adjusts the quality of MPUs to the available bandwidth, and fairly distributes the bandwidth to competing clients. Then, the MMT-CMB delivers the MPUs to the clients through MMTP. While the overall architecture is similar to that of commercial CDNs [16], it is specilaized for MMT media delivery.

3.3. Protocols for Caching Middlebox

We define the MMT-CMB protocol for delivering MMT media from a server to a MMT-CMB. It augments HTTP with a few custom headers to exchange MPU requests, MPU IDs (chunk IDs), and metadata for delivering MMT packages. Table 1 shows the format and usage of custom headers that are defined for content-based caching.

Figure 3 presents an overview of how the protocol works[2]. First, in order to work as server, the MMT-CMB asks for a CI file and an HTML file from the server. After parsing the CI and HTML files, the MMT-CMB retrieves the name of the asset and requests MPU IDs for the MPUs in the asset. On receiving the MPU IDs from the server, the MMT-CMB determines whether the chunk with the ID is cached at its cache storage. In case of a cache hit, it requests the MPU metadata consisting of non-mdat boxes (e.g, moof and moov boxes), and reconstructs the MPU with the cached mbox data. On a cache miss, the MMT-CMB requests the entire MPU and store the MPU content (e.g., only the mdat part) in cache storage along with its MPU ID. While MMT-CMB sends MPUs to client, the client also informs the MMT-CMB of transmission information such an available bitrate or a drop ratio.

3.4. Adaptive Streaming of MMT in Caching Middlebox

MMT-CMB is located close to the clients and monitors the congestion in the last mile which often suffers from bandwidth bottleneck when multiple clients compete to link. With network abstraction for media (NAM) feedback message in MMT, MMT-CMB can be notified the available bandwidth and loss ratio from the client. Moreover, MMT-CMB also can utilize QoS information for media transmission from transport characteristic (ADC) in MMT.

To eliminate buffering event in video playback, MMT-CMB adjust the media with awareness of available bitrate and video encoding rate by dropping MFUs in MPU. Dropped MFUs are selected from end of MFU in MPU for reducing decoding error. The rate-adapted MPUs are delivered through MMTP with encoded bitrate speed and this delivery approach eliminates OFF period which is pointed out the OFF periods as the root cause of QoE problem [5, 6].

4. IMPLEMENTATION AND EVALUATION

We implement an MMT-CMB and an MMT server in 12K lines of C code. Our evaluation shows (a) the caching efficiency when serving the three identical video contents with different names, and (b) fine-grained rate adaptation control at the last mile. We use a machine with a quad core Intel CPU (i7 2600) with hyper-threading on with 8GB of physical memory as MMT-CMB.

Caching effectiveness of MMT-CMB First, to evaluate the caching efficiency of an MMT-CMB, we prepare three MMT packages that have different asset IDs but have the same content, encoded at the rate of 2,694 kbps. A client requests each content sequentially for simulating two cases; (a) an aliased content delivery, requested with a different URL but has the same package and asset IDs and (b) a duplicated content with a different binary due to different package and asset IDs. Figure 4 shows that caching works well for the second and

[2]One can implement the same mechanism using custom web services based on XML-RPC, JSON-RPC, RESTful API, etc.

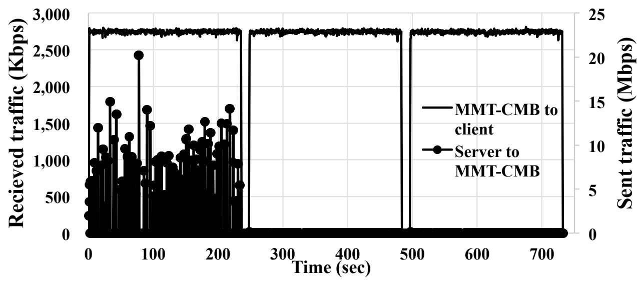

Fig. 4. Measured traffic in MMT-CMB

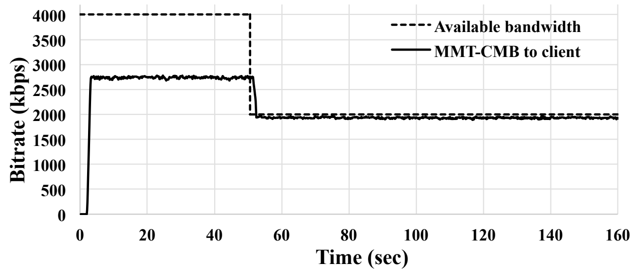

Fig. 5. Adaptation in MMT-CMB

third requests. For the first request, we do see some fetching from the server. One notable aspect is the client-side delivery throughput, which is set slightly higher than the video bit rate. This is because there is per-packet overhead in delivering the contents over the network. For cached contents, the MMT-CMB should get uncached metadata and MPU IDs from the server. However, the amount of extra traffic is negligible, taking up only 0.64% of the video content with the benefit of redundancy suppression.

Adaptation at the last mile With the same MMT packages, we observe the behavior when the available bandwidth varies over time. During video delivery, we change the link bandwidth between the MMT-CMB and the client to 2 Mbps using DummyNet [17]. Figure 5 shows that the MMT-CMB changes the sending rate with the rate-adapted media. That is, the MMT-CMB gets notified of the reduced available bandwidth from the client and adjusts the MPU video quality by dropping some frames. We also observe that it reacts to the link bandwidth change fast and accurately due to its location. We fin that the client can watch the content without any buffering during the experiments.

5. CONCLUSION

MMT is an emerging streaming technology that gains a momentum with broadcasting companies. Despite its strengths, however, it suffers from a lack of scalability from a centralized server architecture. In this work, we alleviate the problem by presenting a caching architecture for MMT. Our caching system places a caching middlebox between the client and the MMT server. The middelbox distributes the load from the server by caching the MPUs by their content names and adapts the video quality more accurately to the varying client-side available bandwidth. We have shown the overall architecture and its protocol, and implemented the system. Our evaluation shows that our system works effectively, meeting the dual-goals of the caching middlebox.

6. ACKNOWLEDGMENTS

This research was supported by Samsung Electronics (Project name: Efficient Media Delivery Technology for Cloud-based Media Delivery Systems, number: IO140305-01361-01).

7. REFERENCES

[1] Cisco, Inc., "Cisco Visual Networking Index: Forecast and Methodology, 2012-2017," 2012.

[2] "HTTP Live Streaming," `https://developer.apple.com/streaming/`.

[3] Iraj Sodagar, "The mpeg-dash standard for multimedia streaming over the internet.," *IEEE Multimedia*, , no. 18, pp. 62–67, 2011.

[4] "Smooth Streaming," `http://www.iis.net/downloads/microsoft/smooth-streaming`.

[5] Saamer Akhshabi, Lakshmi Anantakrishnan, Ali C Begen, and Constantine 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.

[6] Junchen Jiang, Vyas Sekar, and Hui Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies (CoNEXT)*, 2012.

[7] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari, "Confused, timid, and unstable: picking a video streaming rate is hard," in *Proceedings of the 2012 ACM conference on Internet measurement conference (IMC)*, 2012.

[8] Danny H Lee, Constantine Dovrolis, and Ali C Begen, "Caching in http adaptive streaming: Friend or foe?," in *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop (NOSSDAV)*. ACM, 2014.

[9] "ISO/IEC 23008-1:2014, Information technology - Coding of audiovisual objects - Part 12: ISO base media file format," Tech. Rep., 2012.

[10] "NHK MMT UHD system demo," `http://www.nhk.or.jp/strl/english/aboutstrl1/r1-1-7.htm`.

[11] "SKT MMT based True Realtime video streaming solution," `http://www.telecomlead.com/telecom-equipment/sk-telecom-samsung-develop-real-time-mobile/-streaming-technology-53422`.

[12] Shinae Woo, Eunyoung Jeong, Shinjo Park, Jongmin Lee, Sunghwan Ihm, and KyoungSoo Park, "Comparison of Caching Strategies in Modern Cellular Backhaul Networks," in *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2013.

[13] Xiao Wu, Alexander Hauptmann, and ChongWah Ngo, "Practical elimination of near-duplicates from web video search," in *Proceedings of the ACM international conference on Multimedia (MM)*, 2007.

[14] Michael Rabin, *Fingerprinting by random polynomials*, Center for Research in Computing Techn., Aiken Computation Laboratory, Univ., 1981.

[15] Athicha Muthitacharoen, Benjie Chen, and David Mazières, "A low-bandwidth network file system," in *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2001.

[16] Akamai Technologies, Inc., ," `https://www.akamai.com/`.

[17] Luigi Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 27, no. 1, pp. 31–41, 1997.