The analysis of DASH manifest optimizations

Yongjun Wu, Kyle Koceski
Amazon Prime Video
Seattle, USA
yongjuw, kyleko@amazon.com

ABSTRACT

In live video streaming, the size of Dynamic Adaptive Streaming over HTTP (DASH) manifest grows as the number of periods increases and/or the overall time duration of DASH manifest increases. The bigger the manifest size in bytes is, the more computation for manifest generation, manifest storage and parsing and network traffic there will be on service side, the more data to be downloaded, manifest refresh latency, manifest parsing and storage in memory there will be on device side. In this paper, we analyze the techniques and algorithms available to optimize and reduce DASH manifest size with their pros and cons, and limitations of each technique in different scenarios, and propose further optimizations and the adoption of technique(s) in each video streaming scenario according to product requirements, operational cost, system complexity and the requirement of quality of video playback experiences.

CCS CONCEPTS

• Networks • Network services • Cloud computing

KEYWORDS

adaptive bit rate streaming, MPEG-DASH, media timeline, manifest

ACM Reference format:

Yongjun Wu and Kyle Koceski. 2023. The analysis of DASH manifest optimizations. In *Proceedings of ACM Green Multimedia Systems Workshop (GMSys 2023). ACM, Vancouver, Canada, 5 pages.* https://doi.org/10.1145/3593908.3593949

1 Introduction

Video streaming over the internet started back in the 1990s with timely delivery and consumption of large amounts of data being the main challenge. With the increase of internet bandwidth and the tremendous growth of the World Wide Web, now video contents can be delivered efficiently in large segments using HTTP. HTTP streaming has a few main benefits [5]. First, the internet infrastructure has evolved to support HTTP efficiently. Content Delivery Networks (CDN) provide localized edge caches and reduce long-haul traffic to devices. HTTP is firewall friendly and almost all firewalls today are configured to support HTTP outgoing connections. HTTP server technology is a commodity and therefore supporting HTTP streaming for millions of users can scale up easily. Second, with HTTP streaming the client manages the streaming without the need of maintaining a session state on the server. Therefore, provisioning a large number of streaming clients doesn't impose any additional costs on server resources, beyond the standard web use of HTTP, and can be managed by a CDN using standard HTTP optimization techniques. For all the reasons, HTTP streaming has become a popular approach in commercial deployment with a few variants on the market today, such as MPEG DASH [3]. On the other hand, internet video streaming has significant growth in the past several years. Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper [2] shows that globally IP video traffic will be 82% of all IP traffic (both business and consumer) by 2022, up from 75% in 2017. Global IP video traffic will grow four-fold from 2017 to 2022, a Compound Annual Growth Rate (CAGR) of 29%. Internet video traffic will grow fourfold from 2017 to 2022, a CAGR of 33%. Live internet video will account for 17% of Internet video traffic by 2022. Live video will grow 15-fold from 2017 to 2022.

Popular live streaming content providers on the market, such as Amazon Prime Video, NBC, Hulu, Facebook and HBO Go, deploy MPEG-DASH for internet video streaming. Physical partition with separate small segment files is suitable for live streaming where fragments in the future are not available yet at current time. The HTTP requests are simply file based pointing to one individual and independent segment file. In live streaming, there is the interesting challenge about the size of manifest, which might grow linearly as the number of periods increases and/or the overall time duration of DASH manifest increases, in the scenario of long-duration time-shifting or live video recording. It introduces significant overhead for manifest generation, download and network traffic, manifest parsing and manifest storage on both cloud services and more important on devices with limited memory when the DASH manifest size is big.

In this paper, we first present a few scenarios where DASH manifest can grow significantly large in section 2. In section 3, we analyze the techniques and algorithms available to optimize DASH manifest size, and their pros and cons and limitations in different scenarios, and propose further optimizations. In section 4, we

GMSys '23, June 7–10, 2023, Vancouver, BC, Canada

ACM ISBN 979-8-4007-0196-2/23/06.
https://doi.org/10.1145/3593908.3593949

discuss the adoption of technique(s) in different streaming scenarios. In the end, in section 5 we will talk about conclusions and next steps.

2 Scenarios

Users might start watching live events from the beginning or scrub back to a certain time location and start watching from there, besides watching on the live head. To enable such customer experiences, content providers have to generate a full long-duration DASH manifest from the beginning to the live head, and players have to download and refresh the DASH manifest every N seconds, where N is often an integer multiplication of one segment duration, such as 4 or 6, to get the latest updates. There are two kinds of DASH manifests.

- Single period: The DASH manifest might have single period without discontinuities, when content insertion, such as video advertisement insertion, happens before video encoders, i.e. burnt-in live feed. The single-period DASH manifest might have single or multiple audio streams in different languages and/or audio codec formats. As analyzed in [1, 7], the size of the full-duration DASH manifest grows linearly when media timeline is explicitly represented using SegmentTemplate and SegmentTimeline definitions in DASH specification [6], as the overall duration of DASH manifest increases and the number of audio streams increases, due to the representation of audio stream media timeline.
- Multiple periods: The DASH manifest might have multiple periods with discontinuities, for content insertion, such as dynamic video advertisement insertion (DAI), to happen after video encoders, i.e. dynamic video advertisement. The size of the full-duration DASH manifest grows linearly when media timeline is explicitly represented using SegmentTemplate and SegmentTimeline, as the number of periods increases, the number of audio streams increases and/or the overall duration of DASH manifest increases.

As the size of the full-duration DASH manifest grows, the overall video playback performance and metrices will be impacted and eventually customer experiences will be compromised, if the full-duration DASH manifest has to be refreshed and updated every N seconds, consuming network bandwidth, computations and memory on both service and device sides. At the same time, there will be more power and energy consumption.

Another scenario is the video recording just after broadcast live, where video fragments are coming directly from live streaming and its DASH manifest would have the same format as the scenario of a full long-duration manifest in live streaming. Though the video recording scenario doesn't need to refresh manifest every N seconds but is only downloaded once similar to video on demand scenario, the time to first frame (TTFF) of its video playback and energy consumption could be still impacted by a big manifest size.

3 The analysis of techniques and algorithms

There are four techniques and algorithms available to optimize and reduce the size of DASH manifest in a long duration, for both the first manifest download in the beginning and manifest refreshes/updates during live video playback. There are pros and cons of each algorithm and technique to compress and optimize the DASH manifest size with lossless representations.

In this section, we summarize and analyze each known algorithm and technology, Pattern Template Manifest (PTM) [1] from Amazon Prime Video still proprietary not in DASH specification yet, Compacted Manifest available from Elemental Media Package defined as part of DASH specification [6], Patch Manifest (PM) from Hulu also defined as part of DASH specification, and XLink in discussion in DASH community. Each technology is independent of each other and can be done in parallel or one on top of the other, providing incremental benefits in different scenarios.

3.1 Pattern Template Manifest (PTM)

PTM is an Amazon Prime Video proprietary algorithm, not adopted in DASH standard specification yet. It compresses the audio media timeline in each audio stream by a pattern within a period, since the audio fragment duration repeats in a pattern derived as analyzed in [1, 7], different from that in video stream which can be compressed by the run-length encoding representation already defined in DASH specification. It compresses the DASH manifest into constant size without linear growth, independent of the time duration of the overall DASH manifest, when there is one single period, or a small number of periods. It works well for a DASH manifest with a limited number of periods but doesn't work well if there are lots of periods, since each period boundary breaks the continuity of pattern in audio streams and there are lots of duplications and redundancies in syntaxes and values across periods.

3.2 Compacted Manifest (CM)

Elemental CM version 1 [4] can consolidate the redundant syntaxes and values at AdaptationSet level rather than repeating at each Representation in an AdaptationSet. It can reduce the size of DASH manifest when there exist duplication and redundancy across Representations in a period in a DASH manifest, especially in the scenario where there are lots of periods in a manifest and the amount of duplication and redundancy removal is significant compared to the overall manifest size. It doesn't help much when there is only one single period in a long duration in the scenario of burn-in ads, or a small number of periods only.

Currently Elemental CM version 1 [4] only reduces duplication and redundancy up to AdaptationSet, as shown and investigated in the sample, Pattern-Compact.mpd. It was developed and optimized based on an old version of DASH specification [3]. Instead, the new definitions in the latest version of DASH specification [6] can further remove the duplication and redundancy, especially the big data block of 'pssh' inside ContentProtection element. CM version 2 with duplication removal on ContentProtection element can

reduce DASH manifest size significantly, on top of the existing CM version 1, for the scenario of many periods, i.e. DAI. CM version1 and 2 deployed in media services will benefit both burn-in ads and DAI scenario, though the main benefit is for DAI scenario.

In fact, if there are a significant number of periods in a DASH manifest as shown in the manifest sample of original. mpd with 79 periods, the redundancy and duplication of a few other syntaxes and values across periods can be further eliminated by syntaxes and values at Media Presentation Description (MPD) level and using a unique ID to reference them, rather than duplications at Adaptation and Period levels. Specifically, a DASH manifest has a very limited set of bit rate ladders for audio and video streams in general. For example, Amazon Prime Video uses at most two video bit rate ladders in a DASH manifest, one for main content encoding and the other for video advertisement encoding. The bit rate ladder for video advertisement encoding could be exactly the same as that for main content or different. PV keeps using the same encoder settings and ladders for audio streams in both main content and ads. The syntaxes and values about bit rate ladders stored at MPD level, rather than duplicate and redundant Representations and AdaptationSet in each period can further significantly reduce DASH manifest size as shown in the prototype manifest of Pattern-Compact-pssh-ladder.mpd. Such definitions about duplication and redundancy removal for bit rate ladders and more are not part of DASH standard yet. It can be proposed and contributed to DASH standard body as CM version 3 in the future.

3.3 Patch Manifest (PM)

In the scenario of a live streaming including live events, live events on linear, and linear channels with dynamic manifest refreshes, manifest updates can happen frequently and create a considerable bandwidth and processing overhead. In the vast majority of cases, a small minority of elements in the manifest is added or removed. Hulu invented and defined Patch Manifest (PT) to optimize the updates and refreshes of long-duration DASH manifests by delivering the delta patches only in very small size in bytes, instead of the full manifests. PM is already part of the latest DASH specification [6].

However, the very first manifest requested by DASH players could still be big, containing all the information from the beginning to the live head, which could have impacts on TTFF, if other techniques and algorithms are not used to reduce the full manifest size but only PT is applied. Moreover, in the fallback scenario, where PT request fails for some reason, players will have to request the full DASH manifest too.

On the other hand, the service of DASH manifest origin will have to produce two copies of manifests on two end points, one full manifest with updates from the beginning to the live head, and the other delta patch manifest with updates, and CDNs will have to cache two copies too. When there is no manifest personalization existing in some scenarios, such as burn-in advertisement, all users share the same two copies, one full manifest and the other delta patch manifest, both with updates. However, when manifest personalization is present in some scenarios, such as dynamic video advertisement insertion into DASH manifest, each user would need two separate copies. It certainly adds more complexity and challenges, and increases the scaling requirement when the number of concurrent users is large at scale.

Overall, the support of PT in DASH need significant efforts and changes across media services and caching on CDN, fine tuning in download heuristics and manifest parsing and construction in DASH players on devices.

3.4 XLink

DASH specification defines the XLink for enabling remote elements. The remote period variant of remote elements can be used to delay the resolution of ad opportunities. In particular a remote period with @xlink:actuate="onRequest" can be inserted into a manifest as an ad break placeholder. Then a DASH client can perform resolution as the portion of the media timeline containing the remote period is approached during the playout. The response for the resolution can contain one or more periods to represent one or more ad slots within the ad break. Generally, the delayed or just-in-time resolution of ad opportunities can reduce the initial number of periods in a manifest to a certain extent, since one ad break could have more than one ad slot and period. The main scenario is for the optimizations of ad delivery, metrics and revenue under Server-Guided Ad Insertion (SGAI) architecture [8], rather than DASH manifest size.

Remote elements have to be further studied for interoperability. The usage of remote elements within a dynamic live manifest has not been studied sufficiently yet to understand restrictions and constraints that such a manifest would impose on remote elements. Moreover, a DASH client will have to build the capability of just-in-time resolution of ad opportunities with requests at scale to some services which have to return the just-in-time personalized ads and their manifests for the resolutions on DASH clients. The system complexity might increase significantly.

4 Experiments and discussions

In this section, we will present the experiment results in the two major scenarios, the scenario of single period with one or multiple audio streams in a long duration, and the other scenario of many periods with one or multiple audio streams, and analyze how effective the techniques would be in each scenario.

4.1 Single period in a long duration

In the sample manifest, Elemental-2.3-hours.mpd [9], from some live sports game in Amazon Prime Video, it has one audio stream in Advanced Audio Coding (AAC) stereo format in the duration of 2.3 hours, which has the size of 121 K bytes. After we apply the PTM technique, the manifest of Pattern.mpd [9] has the size of 24 K bytes, a reduction of 80.2%, since PTM compressed audio media

timeline representation in constant size by a pattern, without linear growth. We can further reduce the manifest size by applying CM version 2 by removing the redundant and duplication ContentProtection element. We reduce the manifest size from 24 K bytes to 8 K bytes for the manifest of Pattern-pssh.mpd. No matter how long the overall duration grows in the given sample DASH manifest, the size will stay at the size of 24 K bytes without CM version 2, and 8 K bytes with CM version 2, for the full long-duration manifest download in both the first download in the beginning and manifest refreshes for updates in the middle of video playback. If we apply PM on top of PTM and CM version 2, the reduction of DASH manifest size would be marginal, with additional costs and complexity on both services and clients, though PT is able to reduce the manifest size during refreshes from 8 K bytes to a couple of kilobytes or less.

Since there is only one single period in this scenario, there isn't any duplication and redundancy in the audio and video bit rate ladder representations. CM version 3 is not applicable in this scenario.

4.2 Many periods in a long duration

In this section, we perform experiments on a DASH manifest with many periods (79 periods specifically) and two audio streams in AAC stereo and Dolby Digital Plus 5.1 formats, in the duration of 6.1 hours, and examine the efficiency of each technique. The manifest came from another live sports game in Amazon Prime Video.

- The original DASH manifest, original.mpd [9], has the size of 2,884 K bytes where there are 79 periods without PTM and CM applied.
- Pattern-only.mpd [9] has the size of 2,282 K bytes after PTM is applied, for the same representation of MPD. The reason why there is only 20.9% reduction in manifest size rather than 80.2% in the example in section 4.1 is that there are 79 periods and frequent period boundaries in the media timeline, which cause many discontinuities in the pattern template compression and at the same time syntaxes and values in different periods and representations have to be repeated and duplicated 79 times.
- Pattern-Compact.mpd [9] has the size of 762 K bytes after both PTM and CM version 1 applied. CM version 1 removes some duplications and redundancy up to AdaptationSet level, but not all of them. It reduces the size by 66.6% on top of Pattern-only.mpd.
- Pattern-Compact-pssh.mpd [9] has the size of 365 K bytes after removing all the ContentProtection elements. Pattern-Compact-pssh.mpd is a simulation by manually removing all ContentProtection elements, instead of a DASH manifest generated by a cloud service. It reduces the size by 52.1% on top of Pattern-Compact.mpd.
- Pattern-Compact-pssh-ladder.mpd [9] has the size of 279 K bytes after removing all the video bit rate ladder information in video AdaptationSet's. Pattern-Compact-pssh-ladder.mpd is also a simulation by manually removing all duplicate video Representation information about bit rate ladders. It reduces the size by 23.6% on top of Pattern-Compact-pssh.mpd.

We could further eliminate duplicate and redundant information across different periods, such as Pattern element at MPD level rather than Representation or AdaptationSet level. However, that will only provide some more incremental and marginal reduction, beyond Pattern-Compact-pssh-ladder.mpd. When we have to download and refresh the full long-duration DASH manifest, the size will be in the range of 200-300 K bytes for the sample manifest with 79 periods in the duration of 6.1 hours after both PTM and CM with all duplication and redundancy removal are applied. In order to further reduce the manifest size, PM has to be used which is applicable for manifest refresh and update only, not the first download in the very beginning and any fallback scenarios. With PM after the full long-duration DASH manifest is downloaded and available in a client, only the delta patches in the size of a couple of kilobytes will be downloaded by the client during manifest refreshes.

Overall it has to balance among product requirements about the number of periods and the full duration in a DASH manifest which decide the worst-case size of a full DASH manifest, user network conditions, operational costs and system complexity. Then we can decide whether PM should be used due to the dual-manifest structure introduced by PM, especially in the scenario with highly personalized manifest, such as dynamic video advertisement insertion, or PTM and CM together should be sufficient, which keeps using the same end point and one single copy of manifest.

4.3 Reduce initial number of periods by XLink

We are not aware of any services capable of XLink, remote periods and just-in-time resolution of ad opportunities yet as of now. We have to discuss the impacts of XLink on DASH manifest size conceptually. When the number of periods further increases because multiple ad slots and periods are present in one ad break, as shown in section 4.2 the DASH manifest size could be doubled or even more. The delayed resolution can reduce the size of full DASH manifest by consolidating multiple ad periods in one ad break into one period only. The amount of reduction and optimization would depend on both the average number of ad slots per ad break, and the ratio between ad break periods and other main content periods.

5 Conclusions and next steps

In this paper, DASH manifest sizes in different scenarios with a long duration, i.e. single period and many periods with one or many audio streams, are analyzed. We discussed different techniques available to reduce the manifest size, i.e. PTM, CM, PM ad XLink, and presented their pros and cons, and limitations. We also proposed potential additional optimizations in the context of CM, not existing in DASH specification yet.

We did some experiments and simulations with the three techniques (PTM, CM and PM) and provided some guidance in different scenarios about which technique(s) should apply to achieve the optimized decision and balance among operational costs and overhead, system complexity and quality of video playback experiences, and also save computation power and energy. Since XLink needs further study and has no deployment adoption yet, we only analyze its impact on DASH manifest size conceptually.

Next step is to discuss the results and potential additional optimizations in DASH community and drive the adoption in DASH specification, and share the analysis, experiments and guidance in DASH Industry Forum.

REFERENCES

[1] Yongjun Wu, Nicolas Weil, 2019, Pattern Template Manifest for live DASH streaming, DASH-IF F2F Meeting #20, https://github.com/yongjuw-git/documents/blob/main/PatternTemplateManifest_DASH-IF_F2F_May2019.pptx
[2] https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html
[3] ISO/IEC 23009-1 Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats, 2019
[4] Compacted DASH manifests - AWS Elemental MediaPackage (amazon.com)
[5] Iraj Sodagar, 2011, The MPEG-DASH Standard for Multimedia Streaming Over the Internet, Page: 62 - 67, IEEE Multimedia, Vol. 18 , Issue 4 , April 2011.
[6] ISO/IEC 23009-1 Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats, 2022
[7] Higher order manifest data compression, US11063997B1 - Higher order manifest data compression - Google Patents
[8] https://dashif.org/docs/IOP-Guidelines/DASH-IF-IOP-Part5-v5.0.0.pdf
[9] https://github.com/yongjuw-git/manifests