View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Ghent University Academic Bibliography

An HTTP/2 Push-Based Framework for Low-Latency Adaptive Streaming Through User Profiling

Jeroen van der Hooft, Cedric De Boom, Stefano Petrangeli, Tim Wauters and Filip De Turck
Department of Information Technology, IDLab, Ghent University - imec
jeroen.vanderhooft@ugent.be

***Abstract*—Web portals, such as the one hosted by news providers, have recently started to provide significant amounts of multimedia content. To deliver this content over the best-effort Internet, HTTP Adaptive Streaming (HAS) is generally used, allowing smoother playback and a better Quality of Experience (QoE). To stimulate user engagement with the provided content, reducing the video's startup time has become more and more important: while the current median video load time is in the order of seconds, research has shown that user waiting times must remain below two seconds to achieve an acceptable QoE. In this work, we present a framework for low-latency delivery of news-related video content, integrating four optimizations either at server-side, client-side, or at the application layer. Most importantly, we propose to identify relevant content through user profiling, using proactive delivery and client-side caching to reduce the video startup time. By means of a large data set from a Belgian news provider, we show that the proposed framework can reduce the startup time from 4.6 s to 1.5 s (-74.6%) in a 3G scenario, at the cost of limited network overhead and additional complexity at server- and client-side.**

I. Introduction

In the last decade, web portals, such as the ones hosted by news providers, started to provide significant amounts of multimedia content. As an example, deredactie.be, an important Belgian news provider, now offers a large number of video-based news articles, containing individual topics or full news broadcasts [1]. To stimulate user engagement with the provided content, reducing the video's startup time has become more and more important: while the current median video load time is in the order of seconds, research has shown that user waiting times must remain below two seconds to achieve acceptable Quality of Experience (QoE) [2].

To improve the user's QoE, HTTP Adaptive Streaming (HAS) is generally used to deliver the content over the best-effort Internet. In HAS, video content is encoded at different quality levels and temporally divided into multiple segments with a typical length of 2 to 30 seconds [3]. An HAS client requests these video segments in a dynamic way, changing the preferred quality level whenever required. To this end, the client is equipped with a rate adaptation heuristic that selects the best quality level based on criteria such as the perceived bandwidth and the current buffer filling. The client stores the incoming segments in a buffer, before decoding the sequence in linear order and playing out the video on the user's device.

Recently, further improvements to HAS have been made (e.g., SAND and WebSockets [4], [5]), and new technologies have been introduced (e.g., WebRTC [6]). In this work, we present a framework for low-latency delivery of news-related HAS video. It integrates four complimentary optimizations in the content delivery chain: (i) server-side encoding to provide shorter video segments during the video's startup phase, (ii) the use of HTTP/2's server push at the application layer, (iii) server-side user profiling to identify relevant content for each user and (iv) client-side storage to hold proactively delivered content. Through a relevant use case, we show that the proposed framework can reduce the startup delay of video streaming sessions, at the cost of limited network overhead and additional complexity at server- and client-side.

The remainder of this paper is structured as follows. The proposed framework is presented in Section II, along with related work regarding every optimization. The experimental setup is discussed in Section III, followed by a parameter analysis and a discussion of the most relevant results. In Section IV, conclusions are drawn and future work is discussed.

II. Proposed Framework

The proposed framework integrates four complimentary optimizations in the content delivery chain, as illustrated in Figure 1. First, we consider video encoding at server-side, using a shorter video segment duration to improve playout delay. Second, we focus on the applied application layer protocol, discussing the possibilities of HTTP/2's server push feature. Third, we consider user profiling as a way to predict user interest and interaction. Fourth, client-side storage is considered to hold content which is proactively delivered to the user, once it is deemed of interest by the profiling algorithm.

A. Server-Side Encoding

The first part of the proposed framework consists of server-side encoding, and more specifically on the segment duration of the provided content. While traditional streaming solutions use a fixed segment duration in the order of 2 to 30 seconds, we will use different segment durations for the startup and steady-state phase of the video streaming session. As found in previous work, reducing the duration of video segments comes with a number of advantages [7]. Most importantly, the short segments require a lower download time, resulting in a reduced playout delay. However, since every segment has to start with an Instantaneous Decoder Refresh (IDR) frame, a higher bit rate is required to achieve the same visual quality compared to segments of higher length. This encoding overhead was analyzed in previous work, showing that a segment duration of at least 500 ms should be used [7]. Moreover, since a unique request is required to retrieve each single video segment,

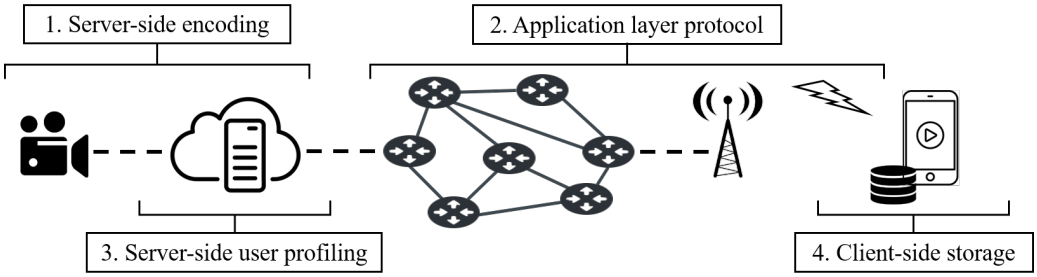

Figure 1: The proposed HAS delivery framework for media-rich content from news providers.

solutions with low segment duration are susceptible to high round-trip times (RTT). This problem mainly arises in mobile networks, where the RTT is in the order of 100 ms, depending on the network carrier and the type of connection.

In the proposed framework, shorter segment durations are only used in the startup phase of the video streaming session: once the first part of video content is retrieved and the buffer is sufficiently filled, the client switches back to a default segment duration of e.g. 10 seconds. This is achieved through appropriate generation of the media presentation description (MPD) file, which contains relevant information on the available content (e.g., the video's duration and available quality representations). Using the MPEG-DASH standard, it is possible to split the video in multiple parts, and define unique characteristics for each of them [8]. As such, the encoding overhead and susceptibility to high RTTs only apply to the startup phase of the video. The former is addressed by using an acceptable minimal segment duration, while the latter is resolved by applying HTTP/2's server push feature at the application layer, as explained below.

B. Application Layer Optimizations

At the start of an HAS video streaming session, a large number of files need to be downloaded. In a stand-alone client, a request is first sent for the video's MPD file. Based on the contents of this file, the client proceeds to download the initialization segment (if any) and from then on, requests video segments one by one. In a web-based context, the HTML page and its required resources need to be fetched as well, including the HAS player, JavaScript sources, CSS files, images, etc. All these resources are requested over HTTP, which among others, allows to traverse firewall and NAT devices, and reuse the existing delivery infrastructure.

In February 2015, the HTTP/2 standard was published as an IETF RFC. Its main purpose is to reduce the latency in web delivery, using request/response multiplexing, stream prioritization and server push. In previous work, we suggested to use the latter in order to push video content from server to client [7]. Pushing video segments back-to-back allows to eliminate idle RTT cycles, reducing buffering time and improving bandwidth utilization. As such, it has the potential to significantly reduce the startup time of video streaming sessions. In related work, Wei et al. explored how HTTP/2's features can be used to improve HAS [9]. By reducing the segment duration from five seconds to one second, they manage to accelerate startup and reduce the camera-to-display delay with about ten seconds.

In the proposed framework, HTTP/2 server push is used to deliver the following sources back-to-back: the HTML source code, the DASH.js reference player, the MPD, the initialization segment and the first k video segments, corresponding to the first ten seconds of the the video stream. Although not applicable in the evaluation setup in Section III, it is worth noting that additional sources, such as images, scripts and CSS, can be retrieved separately.

C. Server-Side User Profiling

A third optimization consists of server-side user profiling. Its purpose is to build a profile for all platform users, determining their preferences towards certain news content. Traditionally, user profiling is about representing the users of a system in such a way that similar users share similar representations. In a recommender systems setting based on collaborative filtering, users and their consumed items are projected in a low-dimensional vector space [10]. The problem with this approach is that the user vectors are often static, which is not always ideal in dynamic scenarios in which many items are consumed one after the other, such as songs, videos and news content. It has recently been shown in literature, and real-life scenarios at Netflix and Spotify, that it is often beneficial to explicitly consider the time aspect by modeling users in a dynamic fashion, for example by updating the user vector with every consumed item [11], [12], [13].

In the proposed framework we will determine whether or not a given (video) article will likely be consumed by a given user. For this purpose we will, as traditionally done in recommender systems, represent each user and article by a low-dimensional vector. The preference of a user $\vec{u}$ for a certain article $\vec{a}$ can then be determined based on the cosine similarity:

$$\cos(\vec{u},\vec{a}) = \frac{\vec{u}\cdot\vec{a}}{\|\vec{u}\|_2\|\vec{a}\|_2}. \quad (1)$$

The higher the similarity, the higher the user's preference towards an article. We will assume that every video has associated textual metadata, and therefore, to represent articles, we will investigate three state-of-the-art natural language processing (NLP) models: Latent Dirichlet Allocation (LDA), word2vec and paragraph2vec [14], [15], [16]. Both LDA and paragraph2vec model documents explicitly, while word2vec operates on word-level. So, for word2vec, we will represent

each article by the sum of word vectors it contains. To represent the users of our system, we will also associate a vector with each of them. This vector is initially an all-zeros vector. Every time a new article is requested by a user, his vector is immediately updated by summation of the user and article vector. To predict then whether a user will read a certain article, the cosine similarity of the article vector and the user vector is determined. If a certain threshold θ is exceeded, we deem the article relevant for the user. The impact of the applied model and the parameter θ is evaluated in Section III.

D. Client-Side Storage

A fourth and final component of the proposed framework consists of client-side storage, which is used to enable proactive delivery of relevant video content. If the right content is sent, using such approach allows to significantly reduce the video session's startup time [17]. Depending on the use case scenario, multiple options for content delivery and storage are possible. In a stand-alone application, dedicated storage on the local device can be used. Based on server recommendations, the application can retrieve content in the background. In web-based applications, control over client-side storage is less evident. Recent versions of browsers such as Google Chrome allow to prefetch web pages which are referred to in the current page. Pages are prerendered in a hidden tab, and moved to the foreground upon request. Most browsers now also support HTTP/2, storing pushed resources in the browser's cache.

In the proposed framework, HTTP/2's server push is used to push relevant content in the background upon connection to the server. To avoid impeding the QoE in video streaming sessions, content will be pushed only if non-video news articles are requested, and once the page is completely loaded. If the client requests a video article which was deemed of interest by the server, content is directly retrieved from the browser's cache, thus reducing startup latency. It is worth noting that proactive delivery of non-relevant content results in network overhead, since bandwidth is wasted on content which may never be consumed. In Section III, both results for the startup time and the network overhead will be discussed.

III. Evaluation and Discussion

Deredactie.be is one of the major news websites in Belgium, hosted by the Flemish Radio and Television Broadcasting Organization (VRT). In recent years, its focus has shifted largely from simple text-based articles towards multimedia-rich news reports. Because of this, the website is an excellent use case for the proposed delivery framework. In collaboration with VRT, Van Canneyt et al. were able to collect a data set containing approximately 300 million website requests, issued between April 2015 and January 2016 [18]. For every request to the website, among others the requested URL, the referrer URL, the server's and client's local time, and the client's hashed IP and cookie ID were logged. These data allow to gain interesting insights, as illustrated in previous work [17].

In a first step, we select the most appropriate NLP model for this use case. To this end, all requested Dutch articles in the data set are retrieved and parsed, in order to extract the title, summary and content of each article. Dutch stopwords are eliminated and the resulting lower-case text is used as input to train the three models. In our setup, all vectors in the system are 100-dimensional. It is worth noting that articles containing video, include at least a title and a brief summary; therefore, user profiling can be applied both on text- and video-based news content. To evaluate the prediction accuracy of the resulting models, we replay all article publications and requests issued by a given set of users within the considered ten-month period. Similar to articles, each user is represented by a 100-dimensional vector, which is updated every time an article is requested. When a new article is published, the cosine similarity of the article vector and the considered user vectors is determined. If a certain threshold θ is exceeded, the article is deemed relevant to the user, which is confirmed or dismissed by analyzing future requests: when a published article is requested, a hit occurs when this article was previously deemed of interest, and a miss otherwise.

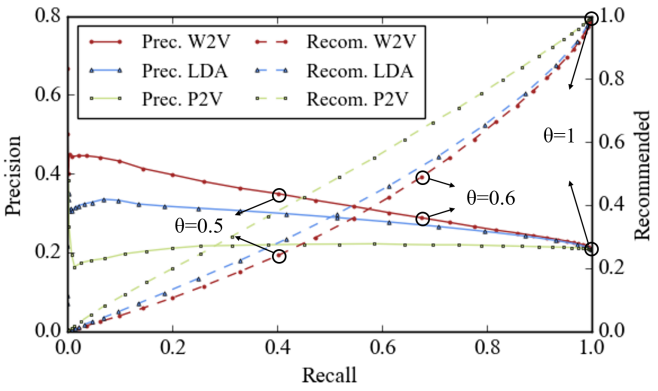

Figure 2: Precision and recall curves for word2vec (W2V), LDA and paragraph2vec (P2V), with varying cosine similarity threshold $\theta \in [0,1]$. The right axis shows the relative amount of recommended articles.

We selected all requests issued by the top 30 video consumers, consisting of 26,539 text-based and 28,060 video-based article requests. Figure 2 shows the obtained precision and recall values for these users, for the three different models and for all cosine similarity threshold values $\theta = k/40, k \in \{0,\ldots,40\}$. No content is recommended for a value of $\theta = 0$ (a user and article vector should be exactly aligned), while all content is recommended for $\theta = 1$. Interpolating the results, the highest precision and recall curve is obtained for word2vec, followed by LDA and paragraph2vec. From the graph, it also follows that less content needs to be recommended to improve the recall: for $\theta = 0.6$, for instance, 48.8% of content is recommended, resulting in a recall of 67.4%. When we compare this to LDA and paragraph2vec, a similar recall requires us to respectively recommend 55.4% and 66.3% of content. From these results, we conclude that word2vec is the most appropriate model for our use case.

In a second step, we evaluate the proposed framework under realistic network conditions. To this end, a 3G network setup

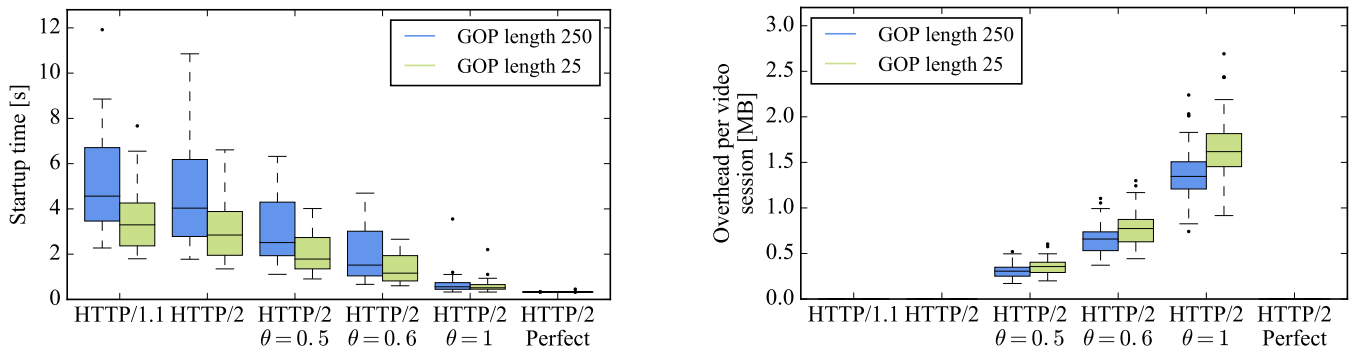

Figure 3: Resulting startup time (left) and network overhead per video session (right), for HTTP/1.1, HTTP/2 with server push, and HTTP/2 with user profiling and 16 MB of storage size (with word2vec and threshold θ, or with perfect prediction).

is emulated using MiniNet, where 30 clients are connected to an HTTP/1.1- and HTTP/2-enabled Jetty server. Traffic control is applied to set the network latency to 100 ms and shape the available bandwidth of each client according to 3G traces provided by Riiser et al. [19]. Clients use the Google Chrome browser in headless mode to start a video streaming session, using the reference DASH.js player. The open-source code of the Jetty server was slightly modified, allowing it to push the first ten seconds of a given video upon request. To allow seamless connection over HTTP/2, a Node.js proxy is provided for each client. This proxy can store pushed content locally, adopting a FIFO logic so that content which was published least recently, is removed first. Because of the time complexity, it is not feasible to replay all 28,060 streaming sessions under a large number of different configurations. Therefore, 30 sessions were selected at random for each user, resulting in 900 measurements for the video's startup time.

Deredactie.be provides its video content at a frame rate of 25 FPS, a resolution of 640×360 and a segment duration of 10 s. For all 9,559 video articles published within the time of logging, the first ten seconds of the embedded video were retrieved at lowest quality. This content was re-encoded using AVC/H.264 with the same frame rate and resolution, but with a segment duration of both 1 and 10 s. To allow each segment to be decoded independently, every segment starts with an IDR frame, and the Group of Pictures (GOP) length is set to 25 and 250 respectively. To realize the same visual quality and target bitrate as the original content, the Constant Rate Factor (CRF) rate control in the x264 encoder is enabled, with a CRF value of 25. This results in an average video bit rate of 361 and 307 kb/s for a segment duration of 1 and 10 s respectively.

Figure 3 shows the boxplots of the average startup time and resulting network overhead for the 30 considered users. The current setup of deredactie.be is depicted on the left, using HTTP/1.1 with a GOP length of 250, i.e. a segment duration of 10 s. For this configuration, the median startup time equals 4.6 s, with outliers as high as 11.9 s. Using a GOP length of 25 allows to reduce the median startup time to 3.3 s, a reduction of 27.8% compared to this configuration. Applying HTTP/2 with server push allows to further reduce the median startup time to 2.8 s (−37.7%) for a GOP length of 25. Since both approaches are reactive, no bandwidth overhead is introduced for these configurations.

Applying all optimizations, considering user profiling and client-side storage with a GOP length of 25 and requests over HTTP/2 in case of missing content, further reductions are possible. Using word2vec with 16 MB of storage size, the median startup time can be reduced to 1.8 s (−60.9%) for $\theta = 0.5$, to 1.2 s (−74.6%) for $\theta = 0.6$ and 0.5 s (−88.7%) for $\theta = 1$. This however comes at the cost of a bandwidth overhead: for $\theta = 0.5$, $\theta = 0.6$ and $\theta = 1$, a median overhead of respectively 0.36, 0.77 and 1.62 MB is observed per video session. Given an average encoding bit rate of 361 kb/s, this translates to roughly 8, 17 and 36 s of wasted content per video session. In general, there is a trade-off between precision and recall: given suitable storage, pushing more content results in more available content and thus a lower startup time, but also results in higher consumption of network bandwidth.

As a reference, results are also shown for theoretically perfect user profiling and prediction. In this case, all content can be retrieved locally, reducing the median startup time to 0.3 s (−92.8%) for a GOP length of 25. Since predictions are perfect, no bandwidth is wasted for this configuration. It is worth noting that all discussed reductions are statistically significant (Wilcoxon signed rank test, $p < 0.01$).

IV. Conclusions

In this work, we presented a framework for low-latency delivery of news-related video content. Its main components include server-side encoding, HTTP/2's server push and user profiling for proactive content delivery and storage. Through a relevant use case, we showed that the proposed framework allows us to significantly reduce the startup delay of video streaming sessions, at the cost of limited network overhead and additional complexity at server- and client-side. As an example, applying word2vec-based user profiling with 16 MB of storage size and a GOP length of 25, the median startup time is reduced from 4.6 s to 1.5 s (−74.6%) in a 3G scenario, while the bandwidth overhead per video session is limited to 0.77 MB. In the future, we will focus on use cases where content is plainly indexed, making user profiling inherently suitable (e.g., video portals).

ACKNOWLEDGMENTS

Jeroen van der Hooft is funded by grant of the Agency for Innovation by Science and Technology in Flanders (VLAIO). Cedric De Boom is funded by grant of the Research Foundation - Flanders (FWO). This research was performed partially within the imec PRO-FLOW project (150223).

REFERENCES

[1] VRT. (2018) DeRedactie.be. [Online]. Available: http://deredactie.be/cm/vrtnieuws/

[2] S. Egger, T. Hoßfeld, R. Schatz, and M. Fiedler, "Waiting Times in Quality of Experience for Web-Based Services," in *International Workshop on Quality of Multimedia Experience*, 2012.

[3] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hossfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *Communications Surveys Tutorials, IEEE*, vol. 17, no. 1, pp. 469–492, 2015.

[4] ISO/ICE, "Dynamic Sdaptive Streaming over HTTP (DASH) - Part 5: Server and Network Assisted DASH (SAND)," 2017.

[5] ——, "Dynamic Sdaptive Streaming over HTTP (DASH) - Part 6: DASH with Server Push and WebSockets," 2017.

[6] W3C/IETF, "Web Real-Time Communication (WebRTC)," https://webrtc.org, online; accessed 12 January 2018.

[7] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, T. Bostoen, and F. De Turck, "An HTTP/2 Push-Based Approach for Low-Latency Live Streaming with Super-Short Segments," *Journal of Network and Systems Management*, 2017.

[8] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE Multimedia*, vol. 18, no. 4, 2011.

[9] S. Wei and V. Swaminathan, "Low Latency Live Video Streaming over HTTP 2.0," in *Proceedings of the Network and Operating System Support on Digital Audio and Video Workshop*. ACM, 2014, pp. 37:37–37:42.

[10] Y. Koren, R. Bell, and C. Volonsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[11] C. De Boom, R. Agrawal, S. Hansen, E. Kumar, R. Yon, C. Chen, T. Demeester, and B. Dhoedt, "Large-Scale User Modeling with Recurrent Neural Networks for Music Discovery on Multiple Time Scales," *Multimedia Tools and Applications*, 2017.

[12] J. Basilico and Y. Raimond, "DéJà Vu: The Importance of Time and Causality in Recommender Systems," in *Proceedings of the Conference on Recommender Systems*. ACM, 2017, pp. 342–342.

[13] T. Donkers, B. Loepp, and J. Ziegler, "Sequential User-based Recurrent Neural Network Recommendations," in *Proceedings of the Conference on Recommender Systems*. ACM, 2017, pp. 152–160.

[14] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," vol. 2013, 2013.

[16] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," 2014, pp. 1188–1196.

[17] J. van der Hooft, S. Petrangeli, T. Wauters, R. Rahman, N. Verzijp, R. Huysegems, T. Bostoen, and F. De Turck, "Analysis of a Large Multimeda-Rich Web Portal for the Validation of Personal Delivery Networks," in *Proceedings of the Sysposium on Integrated Network and Service Management*, 2017, pp. 714–719.

[18] S. Van Canneyt, B. Dhoedt, S. Schockaert, and T. Demeester, "Knowledge Extraction and Popularity Modeling Using Social Media," 2016.

[19] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video Streaming Using a Location-Based Bandwidth-Lookup Service for Bitrate Planning," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 8, no. 3, pp. 24:1–24:19, 2012.