Low-Latency Delivery of News-Based Video Content

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

ABSTRACT

Nowadays, news-based websites and portals provide significant amounts of multimedia content to accompany news stories and articles. Within this context, HTTP Adaptive Streaming is generally used to deliver video over the best-effort Internet, allowing smooth video playback and a good Quality of Experience (QoE). To stimulate user engagement with the provided content, such as browsing and switching between videos, reducing the video's startup time has become more and more important: while the current median 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. We developed a framework for low-latent delivery of news-related video content, integrating four optimizations either at server-side, client-side, or at the application layer. Using these optimizations, the video's startup time can be reduced significantly, allowing user interaction and fast switching between available content. In this paper, we describe a proof of concept of this framework, using a large dataset of a major Belgian news provider. A dashboard is provided, which allows the user to interact with available video content and assess the gains of the proposed optimizations. Particularly, we demonstrate how the proposed optimizations consistently reduce the video's startup time in different mobile network scenarios. These reductions allow the news provider to improve the user's QoE, reducing the startup time to values well below two seconds in different mobile network scenarios.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; *Content ranking*; *Personalization*; • **Networks** → *Application layer protocols*; *Mobile networks*; *Public Internet*;

KEYWORDS

HTTP Adaptive Streaming, news content, low-latency, segment duration, HTTP/2, content ranking, prefetching, mobile networks

ACM Reference Format:
Jeroen van der Hooft, Dries Pauwels, Cedric De Boom, Stefano Petrangeli, Tim Wauters, and Filip De Turck. 2018. Low-Latency Delivery of News-Based Video Content. In *MMSys'18: 9th ACM Multimedia Systems Conference, June 12–15, 2018, Amsterdam, Netherlands.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3204949.3208110

MMSys'18, June 12–15, 2018, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5192-8/18/06.
https://doi.org/10.1145/3204949.3208110

1 INTRODUCTION

In recent years, news providers have started to produce significant amounts of multimedia content to accompany news stories and articles. As an example, deredactie.be[1], an important Belgian news provider, now offers a large number of video-based news articles, containing individual topics or full news broadcasts. To encourage consumers to use the provided services, facile user interaction is of the utmost importance. In this context, reducing the video's startup time has become more and more important: while videos generally take in the order of seconds to load, research has shown that user waiting times must remain below two seconds to achieve acceptable Quality of Experience (QoE) [1].

Nowadays, HTTP Adaptive Streaming (HAS) is generally used to deliver video content over the best-effort Internet. In HAS, video is encoded at different quality levels and temporally divided into multiple segments with a typical length of 2 to 30 seconds [5]. An HAS client requests these video segments at the most appropriate quality level, based on e.g., the available bandwidth and the amount of buffered content. 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. This approach generally enables smooth video playout, and therefore results in a higher QoE than traditional video streaming techniques.

In previous work, we presented a framework for low-latency delivery of news-related HAS content [7]. This framework 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. Results showed that these optimizations allow 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. In this paper, we present a proof of concept to demonstrate the gains brought by the proposed framework. In the demo, the user is presented with a dashboard which allows to set different parameter and network configurations, and allows the user to start and monitor different video streaming sessions. Applying the considered optimizations on a dataset of user requests to deredactie.be, the demo shows that the video's startup time can be reduced significantly in different network scenarios.

The remainder of this paper is structured as follows. In Section 2, the previously proposed framework is discussed, elaborating on the advantages of each of the optimizations. The experimental setup and proof of concept is presented in Section 3, discussing the most important features of the provided demo dashboard. Final conclusions are presented in Section 4.

[1] http://deredactie.be/cm/vrtnieuws

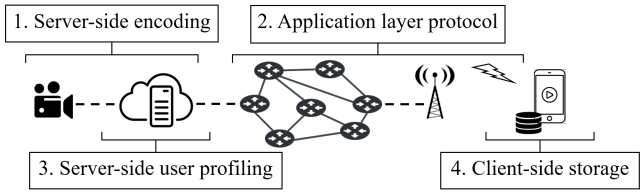

Figure 1: The proposed HAS delivery framework for media-rich content from news providers [7].

2 PROPOSED OPTIMIZATIONS FOR LOW-LATENCY DELIVERY

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 (i.e., without the user explicitly requesting it), once it is deemed of interest by the profiling algorithm.

2.1 Server-Side Encoding

As found in previous work, reducing the duration of video segments comes with a number of advantages [8]. Most importantly, shorter segments require a lower download time, allowing the player to start video playout faster. 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. Moreover, since a unique request is required to retrieve each single video segment, 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 type of connection.

While traditional streaming solutions use a fixed segment duration in the order of 2 to 30 seconds, we propose to use different segment durations for the startup and steady-state phase of the video streaming session. This allows us to both reduce the video startup time, and overcome the aforementioned issues once the video is steadily playing. Two approaches are possible: (i) initially start at the lowest segment duration d_1, switching to the highest segment duration d_n once $k = \frac{d_n}{d_1}$ segments have been downloaded, and (ii) initially start at the lowest segment duration d_1, switching to d_2, d_3, ... until a segment duration of d_n is reached. The advantage of the latter is that the buffer is ramped up smoothly, preventing possible freezes when switching from the lowest to the highest segment duration when the buffer level is relatively low.

2.2 Application Layer Optimizations

At the start of an HAS video streaming session, a number of resources need to be retrieved by the client. Using the DASH standard, the client first sends a request for the video's media presentation description (MPD) file, which contains information on the available content (e.g., the video's duration and available quality representations). Based on the contents of this file, the client then proceeds to download the initialization segment and the required video segments one by one. In 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 to push video content from server to client [8]. Pushing video resources allows to eliminate idle RTT cycles, reducing buffering time and improving bandwidth utilization. As such, it allows to further reduce the startup time of video streaming sessions. In the presented proof of concept, the web page, the video player and the video thumbnails have already been retrieved; therefore, the server will only push the initialization segment and the first k segments as soon as a request for a video's MPD is issued.

2.3 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 system setting based on collaborative filtering, users and their consumed items are projected in a low-dimensional vector space [2]. While user vectors are often static, it can be beneficial to explicitly model users in a dynamic fashion, for example by updating the user vector with every consumed item.

In the proposed framework, we want to determine the relevance of a given (video) article to a given user. As traditionally done in recommender systems, we therefore represent each user and article by a low-dimensional vector. To this end, we assume that every video has associated textual metadata, and apply a natural language processing model to represent each of the articles. Based on previous work, the word2vec model is selected [3, 7]. Since this model operates on word-level, each article is represented by the sum of the word vectors it contains. A user is represented by a vector as well, which is initially an all-zeros vector. Each time a new article is requested by a user, the corresponding vector is updated by summation of the user and article vector in an online fashion. This approach allows us to create a unique vector for each user, building a user profile over time. The relevance of an article $\vec{a}$ to a user $\vec{u}$ can then be determined using the cosine similarity:

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

The higher this similarity, the higher the user's preference towards an article. In the presented proof of concept, this metric will be used to rank different news articles by potential user interest.

2.4 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 (i.e., content which will be consumed in the future), using such approach allows to significantly reduce the video session's startup time. 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.

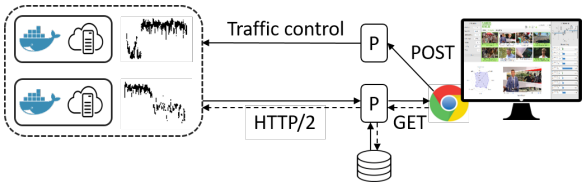

Figure 2: Experimental setup.

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. It is worth noting that HTTP/2 could be used to push additional resources as well, since most browsers allow to store these resources in the browser's cache.

In the proposed framework, recently published content can be prefetched by the client, anticipating future requests. To limit client-side storage and bandwidth overhead, only n videos are considered at each point in time. To determine the most relevant content, three approaches are proposed: (i) rank articles based on the number of requests in the last hour; (ii) rank articles based on recency, i.e., most recent content first, and (iii) rank articles based on the cosine similarity between a user and each of the articles. In the provided proof of concept, each of the these three approaches are showcased.

3 PROOF OF CONCEPT

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 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 [6]. For every request, among others the requested URL, the referrer URL, the server's local time, and the client's hashed IP and cookie ID were logged.

From the given dataset, all users and article requests were extracted. The HTML and XML sources of the requested articles were retrieved, and relevant information such as the title, summary, content and embedded video was extracted. In the demo, a snapshot of deredactie.be is used: the audience is shown a list of the twenty most recently published video articles as of Sunday 30 August 2015, 7.30 PM, complete with poster, title and time of publication. Article requests prior to this point in time are used to build a user profile for different users in the dataset. To this end, the title, summary and text content from each article is extracted, Dutch stopwords are eliminated and the resulting lower-case text is used as input to train a word2vec-based model. Users and articles are both represented by a 100-dimensional vector, allowing us to determine e.g. the cosine similarity between a given user and article. From the set of users, a representative user was selected which mainly has shown interest in Het Journaal (a news program) and articles concerning weather and climate. Ordering the list of videos according to popularity, recency and cosine similarity, will show that the proposed framework is able to capture the user's preference towards these subjects.

By default, deredactie.be provides its video content at a frame rate of 25 FPS, resolutions ranging between 384×216 to 640×360 and a segment duration of 10 s. This content was re-encoded using AVC/H.264 with the same frame rate and resolution, but with a segment duration of 1, 2, 5 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 20. This results in an average video bit rate of 824 and 740 kb/s for the lowest resolution at a segment duration of 1 and 10 s respectively, indicating that there is indeed an encoding overhead for lower segment durations.

The experimental setup used in the demo is shown in Figure 2. Docker containers are running in the background, hosting an HTTP/1.1- and HTTP/2-enabled Jetty server. This server provides the required video content, and is able to parse URL queries requesting the pushing of video resources. Linux' Traffic control (tc) is applied on the Docker interface, so that the available bandwidth of the client can be shaped according to both 3G and 4G traces collected in real mobile networks [4, 8], and the latency can be set to 120 and 60 ms respectively. In the foreground, a dashboard is provided to consume the video content and assess the startup times in different configurations. This dashboard runs in the Google Chrome browser, which is connected to two Node.js proxies. A first proxy is used to issue POST requests, setting the required configurations and for instance (re)starting bandwidth shaping between client and server. A second proxy intercepts and forwards all GET requests to the server, unless a "cached" parameter in the URL's query is set to true: in this case, the proxy retrieves the resources from the local file system and serves it to the client. This way, the startup time can be compared in different scenarios, where content is either retrieved over HTTP/1.1 or HTTP/2, or is assumed prefetched by the client. It is worth noting that these proxies are only required in the demo, but would not be needed in a real-life scenario, where content can be stored in the browser's cache or on the local device.

The provided demo dashboard is shown in Figure 3. The following elements can be distinguished:

(1) **Configuration panel,** with following settings:
- The type of network (3G or 4G);
- The content order (popularity, recency or user profiling);
- The segment duration scheme (constant 10 s, switching from 1 to 10 s, or gradually going from 1 to 2, 5 and 10 s);
- The application layer (HTTP/1.1 or HTTP/2);
- Prefetching (true or false);
- The prefetching approach (prefetch most relevant content based on popularity, recency or user profiling);
- The amount of articles to prefetch (4-8-12-16).

(2) **Video content panel.** Here, the twenty most recently published articles are presented for browsing. Articles which have been watched by the monitored user are indicated with an eye symbol, prefetched articles with a green background;

(3) **Radar chart,** showing the cosine similarity between the user vector and the category vector (i.e., the sum of article vectors, for all articles published within a given news category);

(4) **Embedded DASH.js video player.** The reference player for the DASH standard, which has been slightly modified to enable different segment durations during the streaming

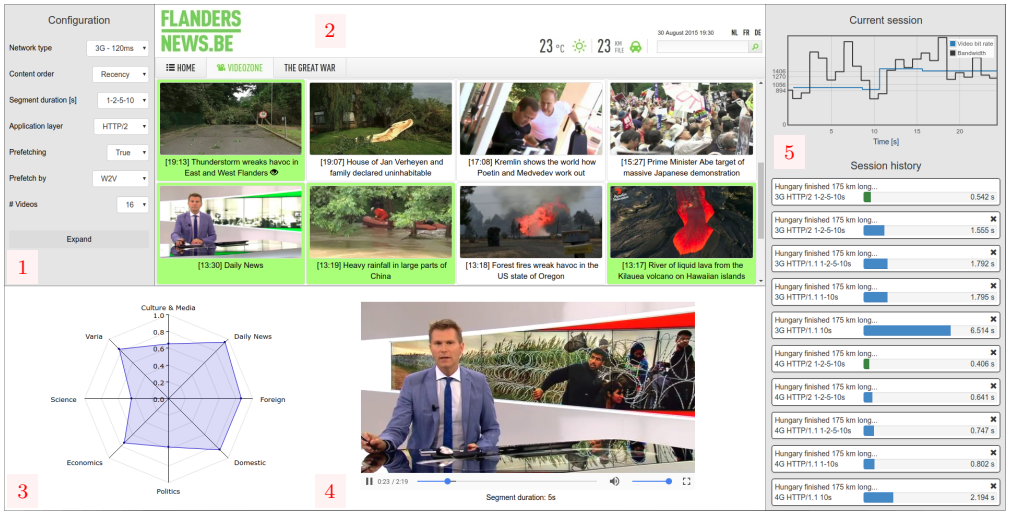

Figure 3: The demo dashboard.

session. Loading is visualized by a loading spinner, and audio has been made available to make stalling more clear. The selected segment duration is shown beneath the video, and can e.g. been seen changing from 1 to 10 s at startup;

(5) **Results panel.** At the top, the available bandwidth and selected quality level are shown for the current video session. This graph changes dynamically over time, giving an indication of the real-time experiment. At the bottom, the startup time of all video sessions is reported, along with the most important configurations. A progress bar is used to visualize the loading time during the startup phase.

Users are allowed to freely interact with the demo, assessing the system's performance by altering the different configuration parameters. It is worth noting that the results in Figure 3 reflect the gains which can be achieved by the proposed framework. For instance, using 3G as network type, the startup time for the default configuration of deredactie.be (i.e. 10 s segment duration, HTTP/1.1 without prefetching), the startup time of one of the presented videos is 6.5 s. Reducing the segment duration to 1 s, this interval can be reduced to 1.8 s, and additionally applying HTTP/2 server push, to 1.6 s. When prefetching is enabled and available content is selected, the startup time goes down to values below 0.6 s. Given the available quality representations, startup times are significantly lower when a 4G network is used. For instance, the startup time is 2.2 s for the default configuration, and 0.6 s for a segment duration of 1 s with HTTP/2 server push. In absolute numbers, reductions in a 4G scenario are lower than for 3G, but in relative numbers, are quite similar. The higher throughput does however allow to consider uses cases with more resource requirements, such as 360° video delivery for virtual reality. This will be the subject of future work.

4 CONCLUSIONS

In this paper, we presented a proof of concept of a framework for low-latency delivery of news-related video content. This framework incorporates four complimentary optimizations, which reside either at server-side, client-side, or at the application layer. The resulting demonstration, which is based on the dataset of a major Belgian news provider, allows us to show that significant reductions to the video startup time can be achieved by each of the proposed optimizations. These reductions allow the news provider to significantly improve the user's Quality of Experience, encouraging low-latency user interaction with the provided video content. In future work, we will apply the considered optimizations in other use case scenarios, such as 360° video delivery for virtual reality.

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 PROFLOW project (150223) and the FWO "Optimized source coding for multiple terminals in self-organising networks" project (G025615N).

REFERENCES

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

[2] Y. Koren, R. Bell, and C. Volonsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. 2013 (2013).

[4] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen. 2012. Video Streaming Using a Location-Based Bandwidth-Lookup Service for Bitrate Planning. *ACM Transactions on Multimedia Computing, Communications and Applications* 8, 3 (2012), 24:1–24:19.

[5] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia. 2015. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys Tutorials* 17, 1 (2015), 469–492.

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

[7] J. van der Hooft, C. De Boom, S. Petrangeli, T. Wauters, and F. De Turck. 2018. An HTTP/2 Push-Based Framework for Low-Latency Adaptive Streaming Through User Profiling. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*. Accepted for publication.

[8] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. 2016. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters* 20, 11 (2016), 2177–2180.