Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2013, Article ID 471683, 16 pages
http://dx.doi.org/10.1155/2013/471683

Research Article

Client-Driven Joint Cache Management and Rate Adaptation for Dynamic Adaptive Streaming over HTTP

Chenghao Liu,[1] Miska M. Hannuksela,[2] and Moncef Gabbouj[1]

[1] *Department of Signal Processing, Tampere University of Technology, 33720 Tampere, Finland*
[2] *Nokia Research Center, 33720 Tampere, Finland*

Correspondence should be addressed to Chenghao Liu; chenghao.liu@tut.fi

Received 9 October 2012; Revised 20 December 2012; Accepted 3 January 2013

Academic Editor: Yifeng He

Due to the fact that proxy-driven proxy cache management and the client-driven streaming solution of Dynamic Adaptive Streaming over HTTP (DASH) are two independent processes, some difficulties and challenges arise in media data management at the proxy cache and rate adaptation at the DASH client. This paper presents a novel client-driven joint proxy cache management and DASH rate adaptation method, named CLICRA, which moves prefetching intelligence from the proxy cache to the client. Based on the philosophy of CLICRA, this paper proposes a rate adaptation algorithm, which selects bitrates for the next media segments to be requested by using the predicted buffered media time in the client. CLICRA is realized by conveying information on the segments that are likely to be fetched subsequently to the proxy cache so that it can use the information for prefetching. Simulation results show that the proposed method outperforms the conventional segment-fetch-time-based rate adaptation and the proxy-driven proxy cache management significantly not only in streaming quality at the client but also in bandwidth and storage usage in proxy caches.

1. Introduction

Recently, Hypertext Transfer Protocol (HTTP) [1] has been widely used for the delivery of real-time multimedia content over the Internet, such as in video streaming applications. Unlike Realtime Transport Protocol (RTP) over User Datagram Protocol (UDP), HTTP typically traverses through firewalls and network address translators (NATs), which makes it attractive for multimedia streaming applications. Kim and Ammar [2] reported that short-term transmission rate variation in HTTP/TCP can be smoothed out through buffering at the receiver side. Recently, standardization projects on dynamic adaptive streaming over HTTP have been carried out, which are briefly reviewed next.

Adaptive HTTP streaming (AHS) was first standardized in Release 9 of the packet-switched streaming (PSS) service [3] by the Third Generation Partnership Project (3GPP). The Moving Picture Experts Group (MPEG) took 3GPP AHS Release 9 as a starting point for its newly published MPEG Dynamic Adaptive Streaming over HTTP (DASH) standard [4]. 3GPP continued to work on adaptive HTTP streaming in communication with MPEG and recently published the 3GP-DASH standard [5]. MPEG DASH and 3GP-DASH have a common core and are therefore collectively referred to as DASH in this paper.

In DASH, a client continuously requests and receives small segments of multimedia content, denoted as media segments. Rate adaptation can be easily supported by requesting media segment encoded at different bitrates. The capability for rate adaptation is one of the key advantages of adaptive HTTP streaming compared to current real-time video services in the Internet, which use a progressive download approach and, hence, may suffer from frequent playback interruptions and suboptimum streaming quality. Nevertheless, DASH standards merely specify the formats for Media Presentation Description and media segments, while client operations, such as rate adaptation, are not normative. However, rate adaptation affects the perceived quality at the client through the selection of the media segments and, hence, the media bitrate to be received as well as the experienced interruption frequency, which also results from the selection of the media segments.

Rate adaptation algorithms for DASH have been proposed in the literature. For example, Liu et al. [6, 7] proposed a rate adaptation algorithm for DASH which uses segment fetch time (SFT) as metrics to decide switching-up/down of media bitrates. SFT denotes a time period from requesting a segment to receiving the last byte of the segment by a DASH client. SFT is compared with media duration contained in a segment to decide whether the media bitrate is higher than, lower than, or approximately equal to the end-to-end bandwidth. However, the effect of caching on rate adaptation for DASH was not studied in [6, 7].

It is common that DASH is operated in networks that include proxy caches capable of caching streamed content. A segment will be cached when a proxy cache first receives and responds to a request for the segment. Such a caching is denoted as passive caching to distinguish with the prefetching. The content provider provides multiple representations for a media clip, and the client may dynamically request different segments from different representations as DASH supports rate adaptation. Hence, segments from multiple representations may be cached by a caching proxy, and the cached segments for each cached representation are likely to form an incomplete representation. This paper considers pull-based strategy as the operating scenario, while content preplacement/push-based strategy is out of the scope of this paper.

A consequence of caching incomplete representations is variation in SFT observed by the client. The reason is that a GET segment requested by a client may be served by a proxy cache or an origin server depending on whether or not the proxy caches (on the path of passing the request towards the origin server) have cached the requested segment. The variation in SFT may further result in a change in buffered media time especially when the caching status of consecutive segments in media presentation order varies, for example, when one segment is cached but the next one is not.

The variations in the SFT and buffered media time caused by the cache misses may be interpreted as congestion or throughput changes in the rate adaptation algorithm. A rate adaptation algorithm can use SFT as in [6, 7] or buffered media time as in [8] to determine switch-up/down to a higher/lower bitrate representation. However, a consequence of caching in DASH is that the representation level may change frequently due to variations in SFT and buffered media time. Frequent changes in the media quality, such as encoded media bitrate and frame rate, may be annoying to users. Furthermore, requesting a high-bitrate encoded segment upon observing a short SFT for fetching cached segments may result in buffer draining.

In addition to the previously discussed impacts of incompletely cached representations on rate adaptation, dynamically varying network throughput can also cause changes in buffered media time. Paper [9] presented bitrate and video quality planning algorithms for mobile streaming to reduce number of buffer underruns and frequency of quality changes. The method presented in [9] uses a GPS-based bandwidth-lookup service to enable client to predict bandwidth using combination of geographical location and past network conditions. In a certain trip, the receiver uses the predicted bandwidth in the potential future route to plan future quality so that it consumes the buffer completely while at the same time avoids buffer underruns during outages and reduces the number of quality changes.

To cope with the problems in the traditional rate adaptation methods for DASH operating over proxy caches, this paper presents a novel joint client-driven prefetching and cache-aware rate adaptation algorithm for DASH. The proposed joint client-driven prefetching and rate adaptation algorithm determines the representation for the next requested media segments and also for the subsequent segments which the client will most probably request subsequently. Then, as part of a regular HTTP GET request, the client also indicates the segments expected to be requested subsequently (hereafter referred to as anticipated segments) so that the proxy cache can prefetch the segments to respond to the future requests of the client. Another important contribution of this paper is a new algorithm of predicting buffered media times which can be used in rate adaptation for DASH operating in networks including proxy caches. The predicted buffered media time is derived from the following two factors. The first factor is the caching status of the requested segment and the anticipated segments. The second factor is the separately measured SFTs from the proxy cache to the client and from the origin server to the proxy cache.

The rest of the paper is organized as follows. Section 2 presents related works and highlights the drawbacks of current proxy-driven prefetching techniques when applied in DASH. A brief overview of the terminology and the architecture of DASH operating in networks including proxy caches is described in Section 3. The proposed signaling method between a DASH client and a proxy cache is presented in Section 4. The proposed joint client-driven prefetching and rate adaptation (CLICRA) method are presented in Section 5. Simulation results are presented in Section 6, while Section 7 concludes the paper.

2. Related Work

Prefetching segments at proxy caches for DASH mainly involves two issues, namely, specifying the time of issuing prefetching requests and selecting segments to be prefetched. To deal with these issues, this paper proposes a novel client-driven proxy prefetching method which moves the intelligence from the proxy cache to the DASH clients. To our knowledge, no prior work in the literature uses client-driven proxy prefetching methods. On the other hand, proxy-driven cache has been proposed, for example, by Chen et al. [10]. The authors developed a proxy-driven active prefetching method to decide when to prefetch uncached segment to overcome the shortcomings of passive caching. In the same work, proxy cache is used to prefetch segments based on the media bitrates and the estimated bandwidth between the "origin" server and the proxy cache to reduce congestion between them and reduce the transmission delay jitter perceived by the proxy cache. The prefetching method in [10] was developed under the assumption that the bandwidth between the proxy cache and the streaming client is large enough to provide smooth

streaming. However, such an assumption may not always be true, for example, in mobile media streaming over wireless networks, which are characterized with a limited and varying bandwidth.

In general, proxy-driven prefetching methods have limitations to be used in DASH because of the following reasons. The proxy-driven prefetching and client-driven rate adaptation for DASH cannot operate well together, since intelligence of prefetching and rate adaptations are located in two different entities, that is, proxy cache and client, respectively. Proxy-driven prefetching methods typically prefetch future segments of a specific media bitrate in a certain window without being aware of whether or not a switch-up/down will occur at a DASH client, or a proxy cache has to estimate clients' rate adaptation behavior. However, such estimation is very difficult as DASH clients do not use a uniform rate adaptation algorithm. When switching-up/down representation occurs in a DASH client, prefetched segments of the previously cached representation may become useless for the DASH client, and the segments of new representations have to be prefetched. Moreover, proxy-driven prefetching method causes a dramatic increase of the computational and memory demands for deciding when and which segments needed to be prefetched. A general philosophy in DASH is to solve similar scalability issues by locating the intelligence functionality, such as rate adaptation and scheduling of requesting media segments, at the DASH client. For segment fetching, either a serial or a parallel segment fetching method can be used in DASH, by requesting and receiving media segments in sequential [6] and in parallel fashion [11], respectively. The parallel segment fetching method also prefetches future segments. This, however, differs from the client-driven prefetching method as follows. In the parallel segment fetching method, subsequent segments are prefetched to the client regardless of whether or not the subsequent segments are cached by a proxy cache. In the proposed method, segments which are not cached by proxy caches are fetched and cached but not sent immediately to the streaming client. The client-driven prefetching method is superior to parallel segment prefetching method in two aspects. First, unnecessary congestion between a proxy cache and DASH clients can be avoided by prefetching and caching in proxy caches instead of the client. Second, the client-driven prefetching method prefetches and caches segments which are not cached by the proxy cache.

3. Terminology and Architecture of DASH

In DASH, Media Presentation Description (MPD) provides the necessary information for a client to establish an adaptive dynamic streaming over HTTP. MPD contains information describing media presentation, such as an HTTP-URL of each segment to facilitate HTTP GET requests for segments. DASH introduced concept of representation, which is one of the alternative choices of the media content or a subset thereof typically differing by the encoding choice, for example, by bitrate, resolution, language, codec, and so forth. A representation is identified by a representation ID.

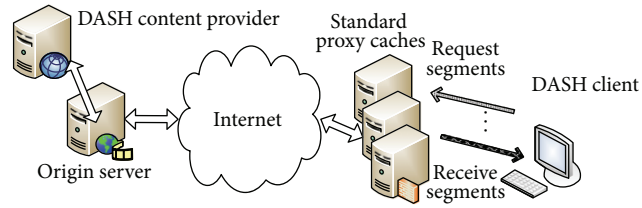


Figure 1: Typical DASH system [11].

A segment contains certain duration of media data starting from a specific presentation time and metadata to decode and present the included media content. Such duration is referred to as media segment duration or segment duration, while the staring presentation time of a segment is denoted as segment start time. A segment number is used to specify the start time and the segment duration for a segment. A segment is identified by a URI and can be requested by a HTTP GET request.

Figure 1 shows a typical media distribution architecture using DASH. The depicted end-to-end DASH system consists of a DASH content provider, an origin server, standard proxy caches, and a number of DASH clients. DASH enables rate adaptation by dynamically requesting media segments from different representations to match a varying network bandwidth. A representation level is used to specify the level of media bitrate, wherein a higher bitrate is associated with a higher representation level, and vice versa. When a DASH client switches up/down the representation level, coding dependencies within the representation have to be taken into account. In DASH, a general concept named Stream Access Point (SAP) is introduced to provide a codec-independent solution for accessing or switching representations. As SAP is specified as a position in a representation that enables playback of a media stream to be started using only the information contained in a representation data starting from that position onwards (preceded by the initialization segment, if any).

A DASH client first accesses an MPD to obtain information of available representations and the URI or URI template for each segment. The client continuously requests and receives segments from a server. DASH clients request media segments from different representations for reacting to varying network resources. Rate adaptation decisions typically take place each time before requesting a segment. If DASH operates in a network including a cache or many caches, a client's GET segment request is first redirected to a cache. If the cache has the requested segment, the request will be served by the cache and the segment fetching time is reduced. Otherwise, the request will be forwarded to the origin server.

4. Signaling Method for Joint Client-Driven Prefetching and Rate Adaptation

Figure 2 shows parts of DASH system architecture, which includes a DASH client and proxy caches. DASH system architecture is shown in Figure 1. The proxy cache can be

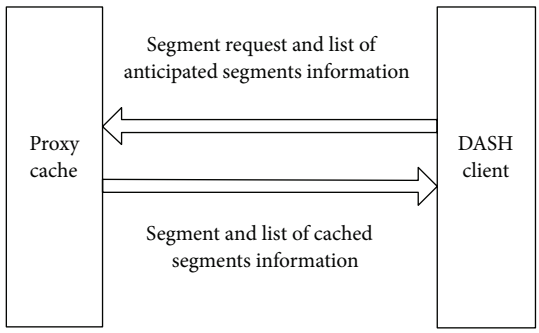


Figure 2: System architecture for DASH assisted with CLICRA-aware cache.

a conventional proxy cache or a proxy cache with segment prefetching functionality as proposed in this paper, which is capable of processing the signaling sent from the DASH client and optionally creating signaling to the DASH client.

The signaling method includes two signaling directions from a DASH client to a proxy cache and signaling from a proxy cache to a DASH client as presented in Sections 4.1 and 4.2, respectively. The signaling method is specific for DASH operating in networks including proxy caches. It should be noted that the signaling presented in Section 4.1 provides the main functionality of the proposed client-driven joint rate adaptation and prefetching. To achieve an enhanced performance of rate adaptation, the signaling presented in Section 4.2 can be further deployed.

4.1. Signaling of Anticipated Segment Information. In the client to proxy cache signaling, the DASH client informs the proxy cache that the DASH client is likely to request one or multiple subsequent segments from a specific representation in the future, referred subsequently to as anticipated segments. For example, the anticipated segment information may be conveyed to the proxy cache by using a new HTTP header named anticipated segment header, when the client sends segment request. The signaling could be received by a CLICRA-aware proxy cache or a conventional proxy cache. If the proxy cache is aware of the signaling and wishes to deploy the conveyed anticipated segment information to improve its efficiency, then it may take advantage of the anticipated segment information to prefetch the anticipated segments, if they have not already been cached by the proxy cache. Otherwise, if the proxy cache is aware of the signaling and does not wish to use the information, or if the proxy cache is not aware of the signaling, then it should ignore the information carried in the header and can operate conventionally, that is, as if the proxy cache did not receive the additional header. Consequently, the anticipated segment informational signaling does not bring any negative effect to proxy caches and could improve proxy cache performance in case it is aware of the signaling.

Note that the signaling is expected to be received by the proxy cache which is closest to the client. To achieve this goal, the anticipated segment header could be included in the connection header of HTTP; hence, the anticipated segment signaling will be only received by the closest proxy cache and will not be received by any other proxy caches nor the origin server.

When compared to proxy-driven prefetching mechanisms, the signaling method releases the processing load at the proxy cache and avoids the problem of scalability when the number of clients connected to the proxy cache increases.

4.2. Signaling of Caching Information. In addition, the DASH client indicates to the proxy cache that caching information of one or multiple subsequent segments of one or multiple representations is requested by the DASH client for the purpose of rate adaptation. The caching information includes the caching status, the time instant of starting to fetch a segment, and the segment fetch time for fetching a segment from the origin server to the proxy cache if the segment is cached. The previous information can be obtained by the proxy cache without intensive computational processing. Hence, the proposed signaling does not affect the efficiency of the proxy cache.

In the proxy cache to client signaling, the proxy cache may inform the DASH client about the caching information of the specified subsequent segments of the representations.

Similar to Section 4.1, the caching information request and the caching information itself can be signaled between the client and the proxy cache using two new headers, called caching request and caching, respectively. The caching request signaling may also be received by the proxy cache, which may or may not be aware of the signaling. In the former case, the proxy cache may inform the client about the caching information together with the responding segment. In the later case, the proxy cache ignores the header as discussed in Section 4.1.

5. Joint Client-Driven Prefetching and Rate Adaptation

This section proposes a novel joint client-driven prefetching and rate adaptation (CLICRA), which moves the processing intelligence from a proxy cache to DASH clients unlike the traditional proxy-driven prefetching methods.

The proposed CLICRA method selects the media bitrate for the current segment to be requested and the anticipated segments. By indicating the anticipated segments to the proxy cache, it can prefetch the anticipated segments which are neither cached nor being fetched, which are hereafter referred to as uncached anticipated segments. Hence, the client-driven joint rate adaptation and proxy cache prefetching are realized by using the anticipated segment information signaling presented in Section 4.1 for DASH operating in networks including proxy caches.

To specify the representation of the segment to be requested and of the anticipated segments, we propose a method of predicting buffered media time for DASH operating in networks including proxy caches. Our method takes into account incompletely cached representations and its impact on the buffered media time changing trend. The

proposed buffered media time prediction method uses the signaling method presented in Section 4.2.

The following sections are organized as follows. A buffered media time prediction method for DASH operating in networks including proxy caches is proposed in Section 5.1. Section 5.2 presents the proposed rate adaptation method, which switches up/down to higher/lower bitrate encoded representation based on the predicted buffered media times.

5.1. Buffered Media Time Prediction Method for Enhanced Rate Adaptation. The DASH client estimates the future trend of the buffered media time by taking into account the fact that the SFTs for fetching a segment from the proxy cache and fetching from the origin server are different. As presented in [6], SFT refers to the time duration from requesting a segment to receiving the last byte of the segment at a DASH client. Hence, the differentiated SFTs can be estimated according to the caching status of selected subsequent segments determined by the client. The proposed buffered media time prediction method enables the rate adaptation algorithm to adapt media bitrate in advance and to avoid unnecessary switch-up/down representation level when DASH operates in networks including proxy caches. The main symbols appearing in rest of the paper are summarized in Table 1 for ease of reference.

Buffered media time can be predicted by adding the differences of the media segment duration and SFTs of the subsequent segments to the current buffered media time. In several candidate representations with rid in the range of $[\mathrm{rid}_c - \gamma, \mathrm{rid}_c + \gamma]$, subsequent segments with segment numbers in the range of $[\mathrm{snum}_c, \mathrm{snum}_c + w]$ can be predicted as

$$\overline{B}_{t_{\mathrm{rec}}(\mathrm{rid},\mathrm{snum})} = \overline{B}_{t_{\mathrm{rec}}(\mathrm{rep},\mathrm{snum}-1)} + \mathrm{MSD} - \overline{\mathrm{SFT}}\,(\mathrm{rid},\mathrm{snum}), \tag{1}$$

where $\overline{B}_{t_{\mathrm{rec}}(\mathrm{rid},\mathrm{snum})}$ denotes the predicted buffered media time when the snum-th segment is to be received by the client, and MSD and $\overline{\mathrm{SFT}}\,(\mathrm{rid},\mathrm{snum})$ denote the predefined media segment duration and the predicted SFT for fetching the segment, respectively.

In addition to the predicted buffered media times, variation of predicted buffered media time ($\overline{BV}$) is estimated after receiving the $(\mathrm{snum}_c + w)$th segment compared to the current buffered media time. $\overline{BV}$ with representation identifier rid and subsequent segments window size w is calculated as

$$\overline{BV}\,(\mathrm{rid}, w) = \sum_{\mathrm{sid}=\mathrm{snum}_c}^{\mathrm{snum}_c+w} \left(\mathrm{MSD}\,(\mathrm{rid},\mathrm{sid}) - \overline{\mathrm{SFT}}\,(\mathrm{rid},\mathrm{sid})\right). \tag{2}$$

To solve (1) and (2), the SFTs of subsequent segments within a sliding window are predicted according to the caching status of subsequent segments at the corresponding time to issue the requests.

5.1.1. Segment Fetch Time Prediction. SFT is predicted using one of the following three equations depending on the caching status of the segment. The predicted SFT is denoted as $\overline{\mathrm{SFT}}\,(\mathrm{rid},\mathrm{snum})$, which is represented as representation ID, that is, rid, since the SFT is proportional to the encoded bitrate of representation.

Table 1: Definition of the main symbols.

$\overline{B}$	Predicted buffered media time
$\overline{B}_{t_{\mathrm{rec}}(\mathrm{rid}_c,\mathrm{snum})}$	$\overline{B}$ at time instants $t_{\mathrm{rec}}\,(\mathrm{rid}_c,\ \mathrm{snum})$
$\overline{BV}$	Variation of predicted buffered media time
br	Media bitrate
CS	Caching status
MSD	Predefined media segment duration
(rid, snum)	Representation ID and segment number to identify a segment
t_{cache}	Time to cache a segment
t_{rec}	Time to receive a segment
t_{req}	Time to request a segment
$\theta_B^{\uparrow}$, $\theta_B^{\downarrow}$	Upper threshold, lower threshold of predicted buffered media time
$\theta_{BV}^{\uparrow}$, $\theta_{BV}^{\downarrow}$	Upper threshold, lower threshold of variation of predicted buffered media time
snum, snum_c	Segment number, current segment number to be requested by client
SFT	Segment fetch time
$\overline{\mathrm{SFT}}$	Estimated SFT for a subsequent segment
$\widetilde{\mathrm{SFT}}^{c}$	SFT measured by client
$\widetilde{\mathrm{SFT}}^{p}$	SFT measured by proxy cache
$\widetilde{\mathrm{SFT}}_{p2c}$	SFT measurement from proxy cache to client
$\widetilde{\mathrm{SFT}}_{s2p}$	SFT measurement from the origin server to proxy cache
$\widetilde{\mathrm{SFT}}'_{p2c}$	The recent sample of $\widetilde{\mathrm{SFT}}_{p2c}$
$\widetilde{\mathrm{SFT}}'_{s2p}$	The recent sample of $\widetilde{\mathrm{SFT}}_{s2p}$
$p2c$	Proxy cache to the client
$s2p$	Origin sever to the proxy cache
rid, rid_c, rid_p	Representation ID, The current representation ID, the previous representation ID
w, w_f, and w_p,	Window size of the subsequent, prefetching and previous segments
γ	Window size of the representation ID

Case 1. A segment is cached by a proxy cache, and $\overline{\mathrm{SFT}}\,(\mathrm{rid},\mathrm{snum})$ can be set to measured SFT from the proxy cache to DASH client $\widetilde{\mathrm{SFT}}_{p2c}$, see (Appendix C.1) as

$$\overline{\mathrm{SFT}}\,(\mathrm{rid},\mathrm{snum}) = \widetilde{\mathrm{SFT}}_{p2c}\,(\mathrm{rid}). \tag{3}$$

Case 2. A segment is being fetched by a proxy cache, and SFT is predicted as the sum of the time spent to fetch remaining part of the segment and the time required to receive the segment from the cache to the client. The time spent to fetch remaining part of the segment is the time span between the time of caching by the proxy cache (t_{cache} discussed in

Section 6.2) and requesting by the client (t_{req} discussed in Section 6.2). Consider

$$\overline{\text{SFT}}\,(\text{rid}, \text{snum}) = t_{\text{cache}}\,(\text{rid}, \text{snum}) - t_{\text{req}}\,(\text{rid}, \text{snum}) + \widetilde{\text{SFT}}_{p2c}\,(\text{rid})\,. \tag{4}$$

To be compliant with the response cacheability of the cache operation as specified in HTTP/1.1 [12], only a fully cached segment is sent to a DASH client responding to HTTP GET segment request.

Case 3. A segment is neither cached nor being fetched by the proxy cache, and SFT is predicted as the sum of the estimated SFTs from the origin server to the proxy cache $\widetilde{\text{SFT}}_{s2p}$, see (Appendix C.2) and $\widetilde{\text{SFT}}_{p2c}$, as

$$\overline{\text{SFT}}\,(\text{rid}, \text{snum}) = \widetilde{\text{SFT}}_{s2p}\,(\text{rid}) + \widetilde{\text{SFT}}_{p2c}\,(\text{rid})\,. \tag{5}$$

From (1)-(2) and (3)–(5), $\overline{B}_{t_{\text{rec}}(\text{rid},\text{snum})}$ and $\overline{BV}\,(\text{rid}, w)$ can be predicted. Since $\overline{\text{SFT}}(\text{rid}, \text{snum})$ depends on the caching status, the next section presents a method of updating and predicting the caching status of a segment when the segment is to be requested.

5.1.2. Caching Status Update and Prediction. This section presents a method of updating the caching status to the caching status conveyed from the proxy cache and predicting future caching status of a segment when the segment is to be requested. In Section 4.2, the proposed signaling method signals the caching status (CS) of subsequent segments at the time instant when the proxy cache responds to the client request for the current requested segment. Hence, the predicted caching status of each segment in a certain sliding window is estimated consecutively by the DASH client based on the informed caching information as described later.

Figure 3 shows an example of consecutively estimating the caching status of segments from #n to #n + 2 when the DASH client issues the requests for the corresponding segments. Specifically, the figure depicts one initial signaled caching status and three sets of caching status at time instants $t\_\text{req}(n)$, $t\_\text{req}(n+1)$, and $t\_\text{req}(n+2)$, that is, times to issue the requests for segment #n, #n + 1, and #n + 2, respectively. In Figure 3, the cached segments, being fetched segment and uncached segment, are represented with light blue, green, and orange blocks, respectively.

As shown in Figure 3, steps 1, 2, and 3 are used to determine the segment caching statuses at $t\_\text{req}(n)$, $t\_\text{req}(n+1)$, and $t\_\text{req}(n+2)$, respectively. Each step includes processes of requesting a segment, identifying the earliest uncached anticipated segments and updating the caching status in a sliding window. The client indicates the earliest uncached segment information to the proxy cache which can prefetch them. In turn, the client can update the caching status, for example, updating the earliest uncached segments to "being fetched" segment according to the receiving caching information. This example sets the sliding window to 2.

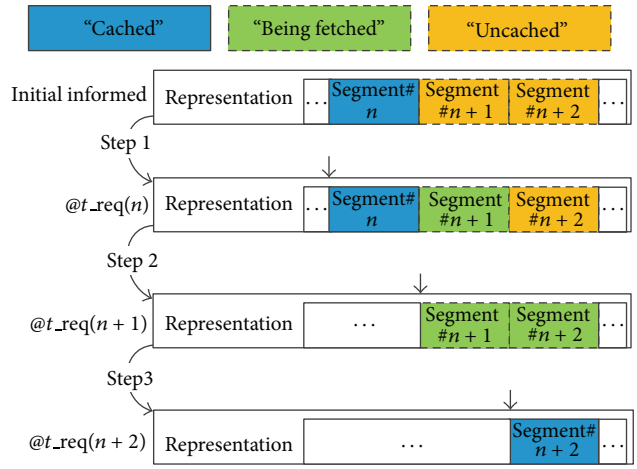


Figure 3: Example of estimating each segment caching status when the segment is to be requested.

The following steps describe the processes of updating and predicting for this example.

Step 1. Updating of the CS at the time to issue the request for the segment #n, that is, $@t_{\text{req}}(n)$ based on the initial caching status signaled from the proxy cache. As the segment #n+1 is the earliest uncached anticipated segment, its CS is updated from "uncached" to "being fetched" $@t_{\text{req}}(n)$. And the CSs of other segments are same as the initial CSs.

Step 2. Updating of the CS$@t_{\text{req}}(n+1)$ based on CS$@\,t_{\text{req}}(n)$. Let us assume that segment #n + 1 is not fully fetched to the proxy cache when segment #n is received by the DASH client. So, CS of segments #n+1 is "being fetched." The CS of segment #n+2 is updated from "uncached" to "being fetched" $@t_{\text{req}}(n+1)$ as similar to updating the CS of segments #n + 1 in step 1.

Step 3. Updating of the CS$@t_{\text{req}}(n+2)$, that is, time instant to issue the request for segment #n + 2. Assume that segment #n + 2 is cached when segment #n + 1 is received by the DASH client. Thus, segments #n + 2 caching status changes to "cached" from "being fetched".

For the detailed procedure of updating and predicting caching status and equations of determining t_{cache} and t_{req}, please refer Appendix B.

5.2. Joint Rate Adaptation Algorithm for Requested Segment and Anticipated Segments. The main objectives of rate adaptation for DASH consist of the following three aspects. First, the rate adaptation aims to provide the maximum possible media bitrate. Second, it tries to avoid playback interruptions. And third, it should reduce the bitrate adaptation frequency and the average magnitude of the bitrate change for each adaptation to provide a stable playback quality. The first two targets require the rate adaptation to adapt media bitrates quickly to the varying network resources, but the last target requires it to react conservatively to the long-term network resource variations.

To achieve the aforementioned objectives, this paper proposes a novel technique which selects an identical encoded

media bitrate of a representation for the current requested segment and the anticipated segments. The proposed CLICRA examines the predicted buffered media times at several future time instants in the adjacent representations and selects the encoded media bitrates of the representation so that the predicted buffered media times are in a safety level.

The predicted buffered media times based rate adaptation method can ignore short-term SFT variations similarly to the current buffered media time-based method by using a conservative threshold to invoke a rate adaptation. With the knowledge of predicted buffered media times, the DASH client no longer needs to wait until the buffered media time reaches a certain threshold which will invoke a rate adaptation.

Rate adaption consists of switching-up, switching-down, or keeping a representation unchanged.

Switching-down (switching-up) representation occurs when the network capacity is lower (larger) than the threshold required to deliver the current bitrate encoded representation and enables the delivery of one of the lower (higher) bitrate encoded representations. If the network capacity is lower (larger) than the media bitrate, then the buffer filling rate is lower (higher) than the buffer consumption rate. So, a lower (higher) network capacity can be detected by checking if one (multiple) predicted buffered media time(s) in the current representation is (are) lower (higher) than the predefined threshold (and in a higher bitrate encoded representations are higher than a defined switch-up threshold). Here, the predicted buffered media times consist of multiple predictions in future times when the segment is to be received by the client and denoted as $\overline{B}_{t_{\text{rec}}(\text{rid,snum})}$. An identical lower (higher) bitrate encoded representation, also denoted as lower (higher) representation level, is selected as the representation for the requested segment and for the anticipated segments so that the predicted buffered media times in the selected lower (higher) representation level are higher than the switching-down (switching-up) threshold.

In all other cases, the representation is kept unchanged.

6. Simulation Results

In the performed simulations, the CLICRA method was operated at the DASH client, and only the signaling functionality was operated at the proxy cache. The proposed CLICRA method for DASH was implemented in ns2 [13]. Two combinations of rate adaptation and proxy cache were also implemented in ns2 to compare it with the proposed method. In the first combination, called TraRA in this paper, a traditional rate adaptation algorithm for DASH similar to [6] was operated with a conventional proxy cache without prefetching. In the second combination, called TraCaMaRA in this paper, the same traditional rate adaptation was operated with a proxy-driven prefetching method similar to the active prefetching method [10].

In many DASH services, it is expected that a large number of DASH clients are streaming at the same time and therefore compete for a limited bandwidth. A network simulator, such as ns2, can be considered a good choice for evaluating prefetching and rate adaptation methods for a large number of clients.

6.1. Network Topology and Settings. The network topology used in the simulation is shown in Figure 4. One origin server, two cascaded proxy caches, a network element, and n DASH clients were included in the topology. The two layers of proxy caches were equipped with the caching functionality, but the network element, which could be for example be a router collecting multiple digital subscriber line (DSL) connections, does not provide caching functionality. The same media segments will be sequentially delivered from the origin server to proxy caches #1 and #2, where they were cached to serve future requests. The same media segment may be delivered from proxy cache #2 to the network element and to DASH clients multiple times. Therefore, in case many clients start streaming at the same time, the bandwidth consumption from proxy cache #2 to the network element is expected to be large. Bandwidths from the network element to clients, that is, links #4, were different for different clients as described later.

In our simulation, a group of five DASH clients were simulated wherein the bandwidths of link #4 were set to 0.4 Mbits/s, 0.7 Mbits/s, 1.0 Mbits/s, 1.3 Mbits/s, and 1.6 Mbits/s, respectively. To evaluate DASH performance under dynamically changing shared network resources and varying caching ratios at proxy caches, DASH clients were set to start media streaming progressively at intervals of 200 s from 0 s to 800 s. The simulated DASH client numbers for each group assigned to a certain bandwidth were set to 2, 4, 6, 4, and 2 at [0s, 50s], [200s, 250s], [400s, 450s], [600s, 650s], and [800s, 850s], respectively. So, the total number of simulated DASH clients was 90. All clients requested the same media clip, and the playback duration of the media clip was 600 s.

To achieve adaptive HTTP streaming, the streaming server provides ten sets of representations for clients to adapt the media bitrates wherein the media bitrates include 64, 128, 192, 256, 384, 512, 640, 896, 1152, and 1408 Kbits/s for representation levels 0 to 9, respectively. The bitrate difference between adjacent representations increases as a function of the bitrate for observing a roughly equivalent quality improvement. In the simulations, segment duration, the initial buffering time, and ε were set to 5 s, 10 s, and 30 s, respectively. w, w_p, and w_f were set to 10, 6, and 5, respectively. $\theta_B^{\uparrow}$, $\theta_B^{\downarrow}$, $\theta_{BV}^{\uparrow}$, and $\theta_{BV}^{\downarrow}$ were set to 60 s, 20 s, −30, and −50, respectively.

To simulate the dynamic nature of the Internet, two exponential (Exp) traffic senders were connected each to one proxy cache. The Exp traffic senders #2 and #3 generate on/off traffic. During the "on" periods, packets were generated at constant burst rates of 5 Mbits/s and 10 Mbits/s by the Exp senders #2 and #3, respectively. During the "off" periods, no traffic is generated. Burst times and idle times were taken from exponential distributions. In the simulation, the average on- and off-times were set to 2 s. Exp receiver #2 is connected to the second proxy cache, while receiver #3 is connected to the network element. The origin server also includes an internal exponential traffic generator, which generates

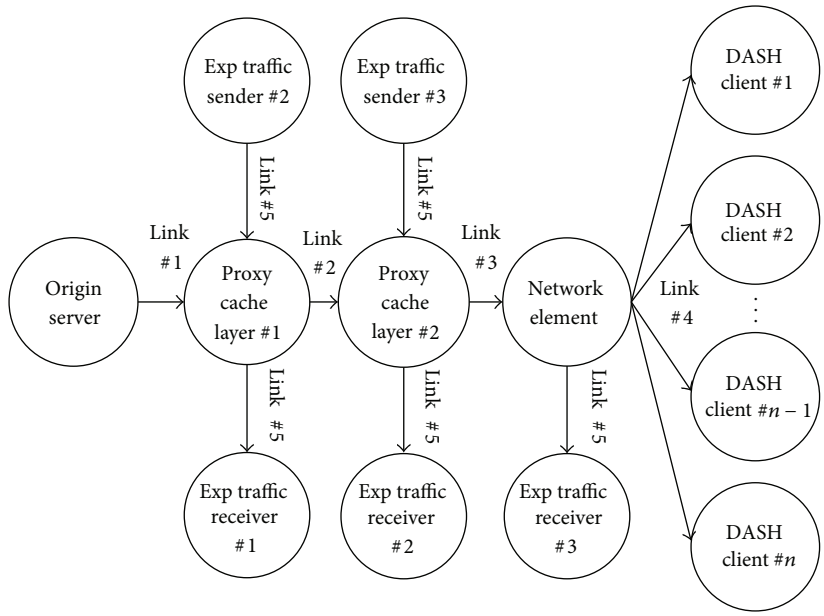


Figure 4: Network topology.

Table 2: Two sets of simulation settings.

Setup	Receive window size	Link parameters	Links #1 and #2			Link #3			Link #5
			Setting #1	Setting #2	Setting #3	Setting #1	Setting #2	Setting #3	Setting #1, #2, #3
Setup 1	1000 Kbytes	Bandwidth (Mbits/s)	10	10	20	20	40	40	10
		Link delay (ms)	100	100	100	10	10	10	10
Setup 2	60 Kbytes	Bandwidth (Mbits/s)	20	20	20	40	40	40	10
		Link delay (ms)	50	100	150	10	10	10	10

packets with a burst rate of 5 Mbits/s during the "on" periods and sends the packets to the Exp traffic receiver #1. The internal exponential traffic generator is used to simulate a scenario where the origin server not only serves DASH clients but also serves other clients such as web browser clients.

The maximum receive window size a receiver can advertise is 65535 bytes in the 16-bit TCP header. In such a case, the achievable maximum TCP throughput is limited to the ratio of the receive window size and RTT. For larger window sizes to accommodate high speed transmission paths, RFC 1323 [14] specifies the window scaling that allows a receiver to advertise a window size larger than 65535 bytes. To evaluate the proposed method in different receive window sizes and RTTs, two different receive window sizes of 60 KBytes and 1000 Kbytes were tested.

For both window sizes, three different sets of link bandwidths and delays were tested as shown in Table 2. A recent measurement on the Internet reported in [15] was considered to set the link delays. The bandwidths were set according to the number of DASH clients and the media bitrates of provided representations. With the set bandwidth, the clients undergo different levels of network congestion and experience different media bitrates. For each setting in Table 2 simulations were run three times to smooth out the impacts of random starting times of different clients.

6.2. Simulation Results and Analysis. The proposed method was evaluated with respect to the perceived streaming experience at the DASH clients and the network resource usage from the origin server to the proxy cache. The perceived streaming experience at the DASH client was evaluated with respect to the convergence in the representation level indicating the media bitrates, the achievable media bitrate, and the playback interruption frequency. The network resource usage was measured in terms of the amount of media traffic from the origin server to the proxy cache layer #2, and the cache hit ratio at the proxy cache layer #2.

6.2.1. Simulation Results for Setup 1. Simulation results were reported for the three different bandwidth combinations with the given delays in the links #1, #2, and #3. The receive window size was set to 1000 Kbytes; hence, the throughput is mainly controlled by the bandwidth of the links.

Table 3 shows the buffer underflow counts for the proposed CLICRA method and the traditional methods TraCaMaRA and TraRA, with bandwidth settings #1, #2, and #3. In the bandwidth settings #1 and #3, buffer underflow occurred at most 7 times for the proposed CLICRA method in all 90 clients, each of which streamed 600 s duration of a media clip. In the bandwidth setting #2, buffer underflow

Table 3: Buffer underflow count.

Bandwidth	CLICRA	TraCaMaRA	TraRA
Setting #1	4	86	77
Setting #2	10	279	345
Setting #3	7	95	181

occurred 10 times for the proposed CLICRA. However, for the TraCaMaRA and TraRA, buffer underflow occurred 77–345 times in all three bandwidth settings.

To evaluate convergence in the representation levels, Figure 5 shows points indicating switching frequency and amplitude with bandwidth setting #1, wherein the x axis denotes the mean of the absolute difference in the representation levels between adjacent segments, called switching amplitude, and the y axis denotes the ratio of the count of representation level switches divided by the count of all received segments. Thus, a point in the bottom-left corner of the figure indicates a superior convergence compared to a point in the top-right corner. Figure 5 also shows the average values of each method as specifically labeled points. Average values of switching frequency and amplitude of each method were reported in Table 4. It shows that the average switching frequency and amplitude were lower than 0.1 and 0.1 for the proposed method but were in the range of 0.2–0.5 and 0.2–0.8, respectively, for the traditional methods.

Simulation results show that the proposed CLICRA method not only provides significantly lower switching frequencies but also lower switching amplitudes compared to the traditional TraCaMaRA and TraRA methods as its corresponding points were clustered in the bottom-left corner for the proposed method; however, those of the competing methods were widely spread and appear at the top-right corner.

The reasons of relatively high buffer underflow count and high switching representation frequency and amplitude in TraCaMaRA can be described as follows. The traditional proxy-driven proxy segments prefetching method prefetches subsequent segments without being aware of the representations of the subsequent segments at DASH clients. The prefetched segments will become useless in case clients switch to a new representation level. Therefore, unnecessary segments may be prefetched, which may result in congestion between the origin server and DASH clients and further result in buffer underflow in a large number of clients. TraRA and TraCaMaRA employ an SFT-based rate adaptation method, which compares the ratio of media segment duration divided by SFT with a rate adaptation threshold to decide whether or not to switch-up representation levels. The incompletely cached representations in proxy caches caused a large variation in terms of SFT measured at a DASH client in fetching cached segments from proxy caches and fetching uncached segments from the origin server. TraRA may switch-up a representation level after observing a smaller SFT for fetching the cached segments from proxy cache and without checking if the subsequent segments in a higher representation were cached by a proxy cache or not. Hence, after switching-up the representation level, a DASH client took larger SFT to fetch

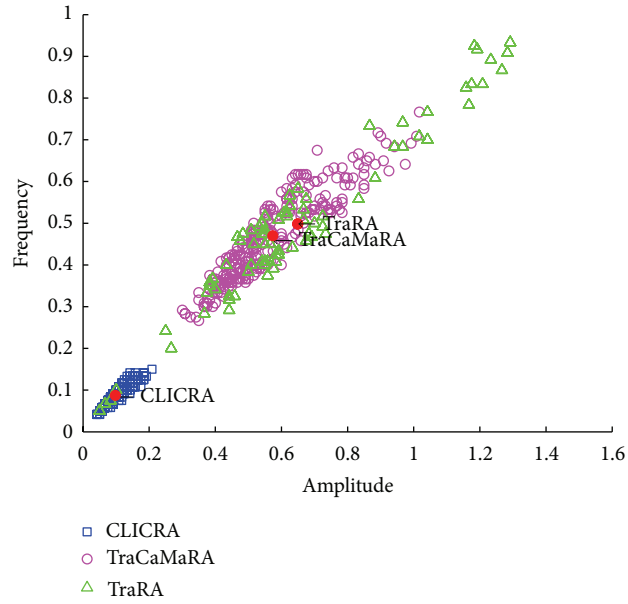


Figure 5: Switch frequency and amplitude.

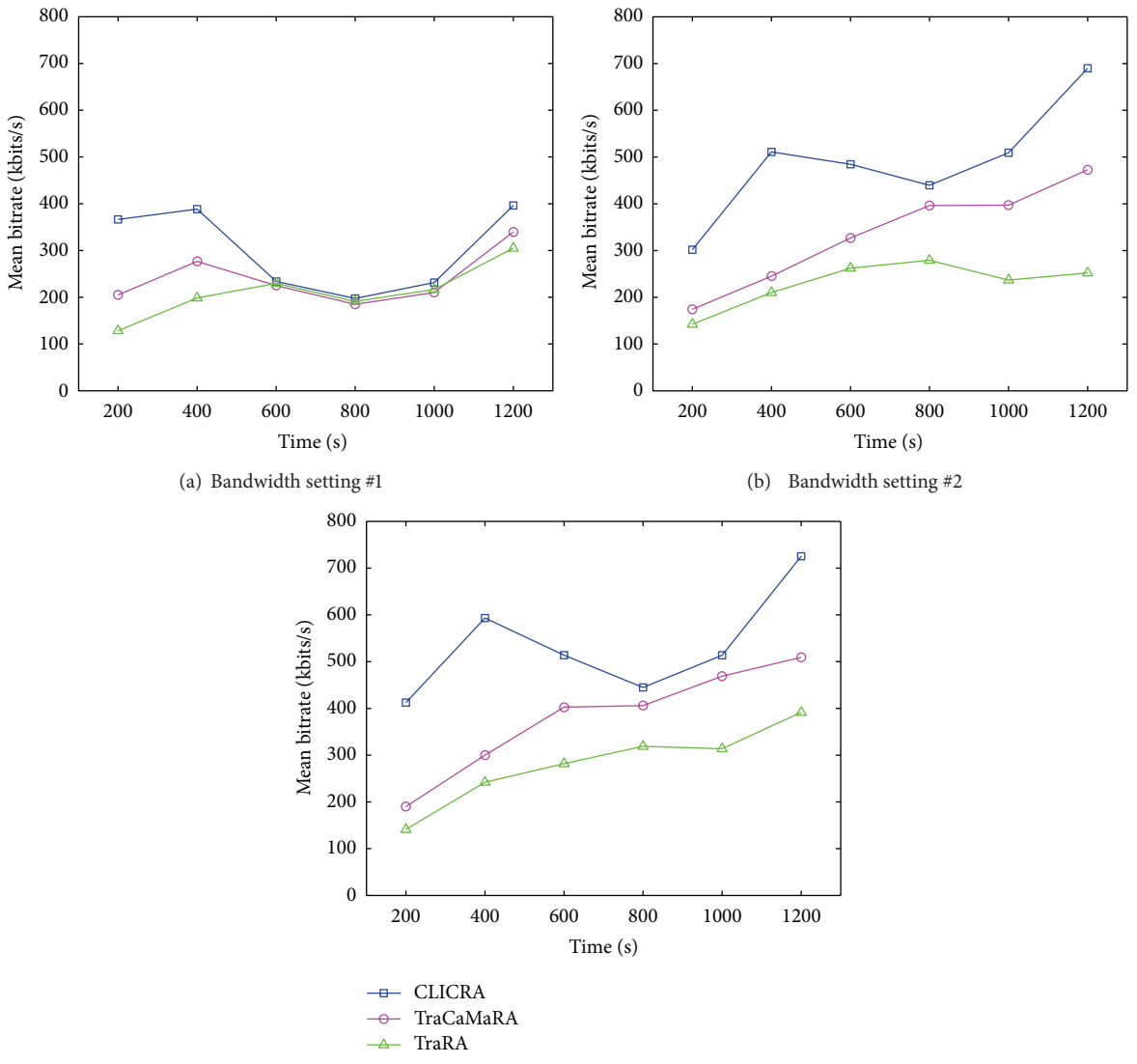
subsequent segments of a higher representation level if such segments were not cached by proxy caches. In case a large number of DASH clients make such incorrect switching-up decisions, the network between the origin server and the proxy cache may suffer from congestion which further results in buffer draining and switching-down representation levels at a large number of DASH clients. Such a repetitive cycle of switching-up/down representation levels resulted in high switching frequencies and amplitudes.

In contrast, the proposed method predicts the SFT for subsequent segments based on the caching statuses of subsequent segments in a certain sliding window and the separately measured SFTs from the origin server to the proxy cache and from the proxy cache to DASH clients. Thus, incorrect switching up/down representation levels can be reduced. Furthermore, the proposed client-driven prefetching method specifies the anticipated segments at DASH clients and indicates the earliest uncached anticipated segments to the proxy cache so that it can prefetch the indicated anticipated segment, hence, reducing prefetching of unnecessary segments as well as decreasing congestion and buffer underflow counts.

Figure 6 shows the average received media bitrates over time, where the x and y axes denote the simulation time t_0 (s) and the average received media bitrate for each client in the interval $[(t_0 - 200), t_0]$, respectively. The proposed CLICRA method shows higher achievable media bitrates compared to the traditional TraRA method. Furthermore, Table 5 shows the number of transmitted bytes from the origin server to the proxy cache. In most of the cases, CLICRA consumed less bandwidth from the origin server to the proxy cache compared to TraCaMaRA. It can be observed that more traffic was delivered from the origin server to the proxy cache by CLICRA compared to TraCaMaRA with bandwidth setting #3. The possible reason is that CLICRA clients quickly

Table 4: Switch frequency and amplitude for receive window 1000 Kbytes.

Bandwidth	Frequency			Amplitude		
	CLICRA	TraCaMaRA	TraRA	CLICRA	TraCaMaRA	TraRA
Setting #1	0.0867	0.4695	0.4984	0.0978	0.5736	0.6491
Setting #2	0.0743	0.2201	0.4959	0.0825	0.3142	0.7176
Setting #3	0.0734	0.2060	0.4611	0.0805	0.2832	0.6653



(a) Bandwidth setting #1

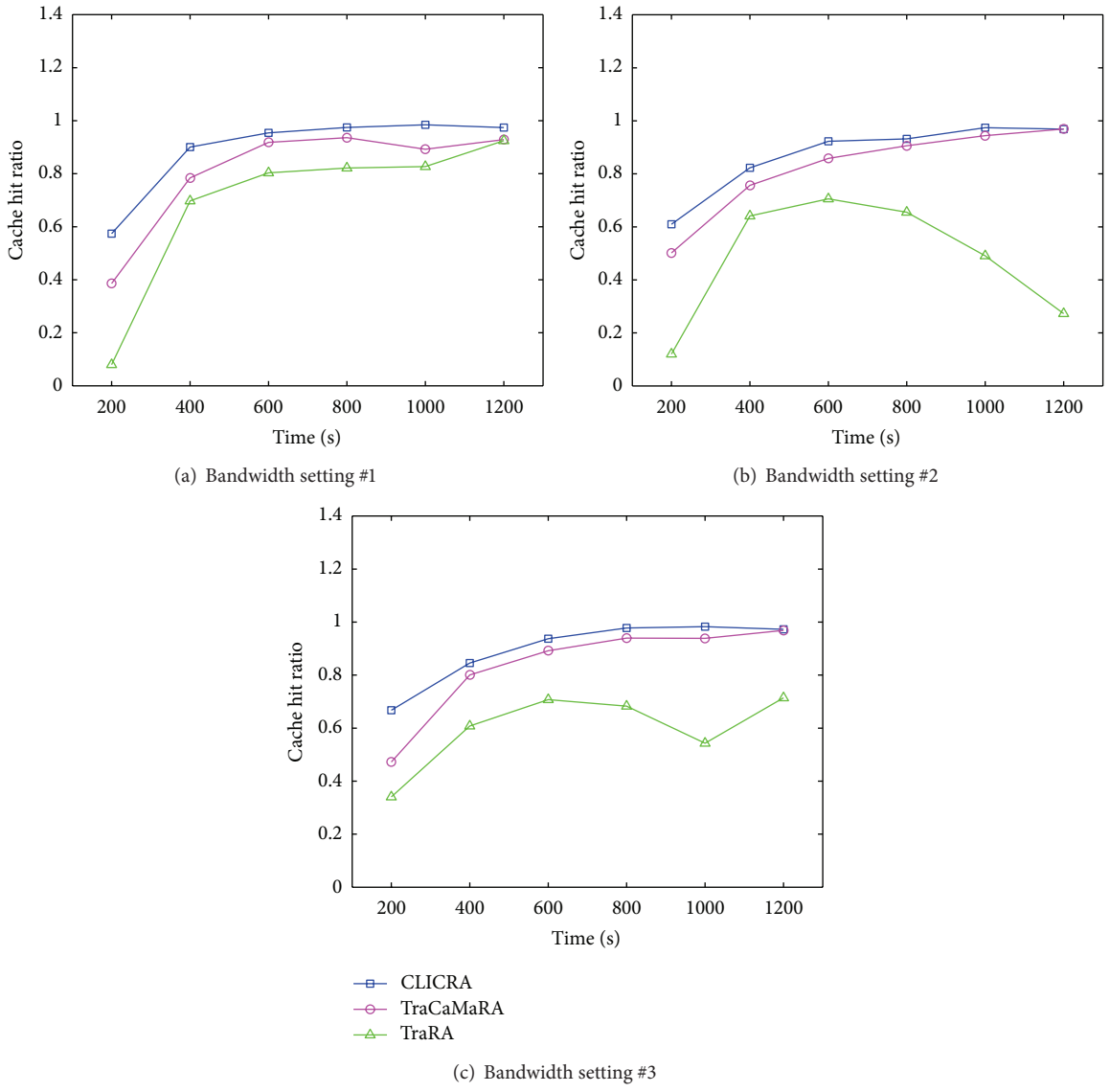
(b) Bandwidth setting #2

(c) Bandwidth setting #3

Figure 6: Received mean bitrate over simulation time for receive window 1000 Kbytes.

switch-up to higher bitrates encoded representations and the TraCaMaRA remains at relatively lower bitrates encoded representations, which causes the results of delivered traffics in CLICRA and TraCaMaRA. Such a result shows that the proposed CLICRA method outperforms TraCaMaRA in terms of fully utilizing the available bandwidth to provide a higher playback quality to the clients.

Figure 7 shows the average cache hit ratio at the proxy cache #2 for clients' requests over time, where the x and y axes denote the simulation time t_0 and the average hit ratio for each client in the interval $[(t_0 - 200), t_0]$, respectively. The cache hit ratios of CLICRA were much larger than those of TraRA. In the beginning, CLICRA outperforms TraCaMaRA. Since 600 s, TraCaMaRA shows similar cache hit ratio with the proposed CLICRA method, but the high cache hit ratio of TraCaMaRA does not demonstrate its superiority due to the high buffer underflow count, the high switching frequency,



(a) Bandwidth setting #1

(b) Bandwidth setting #2

(c) Bandwidth setting #3

Figure 7: Average cache hit ratio of the clients over simulation time for receive window 1000 Kbytes.

Table 5: Transmitted bytes (Gbytes) from the origin server to the proxy cache.

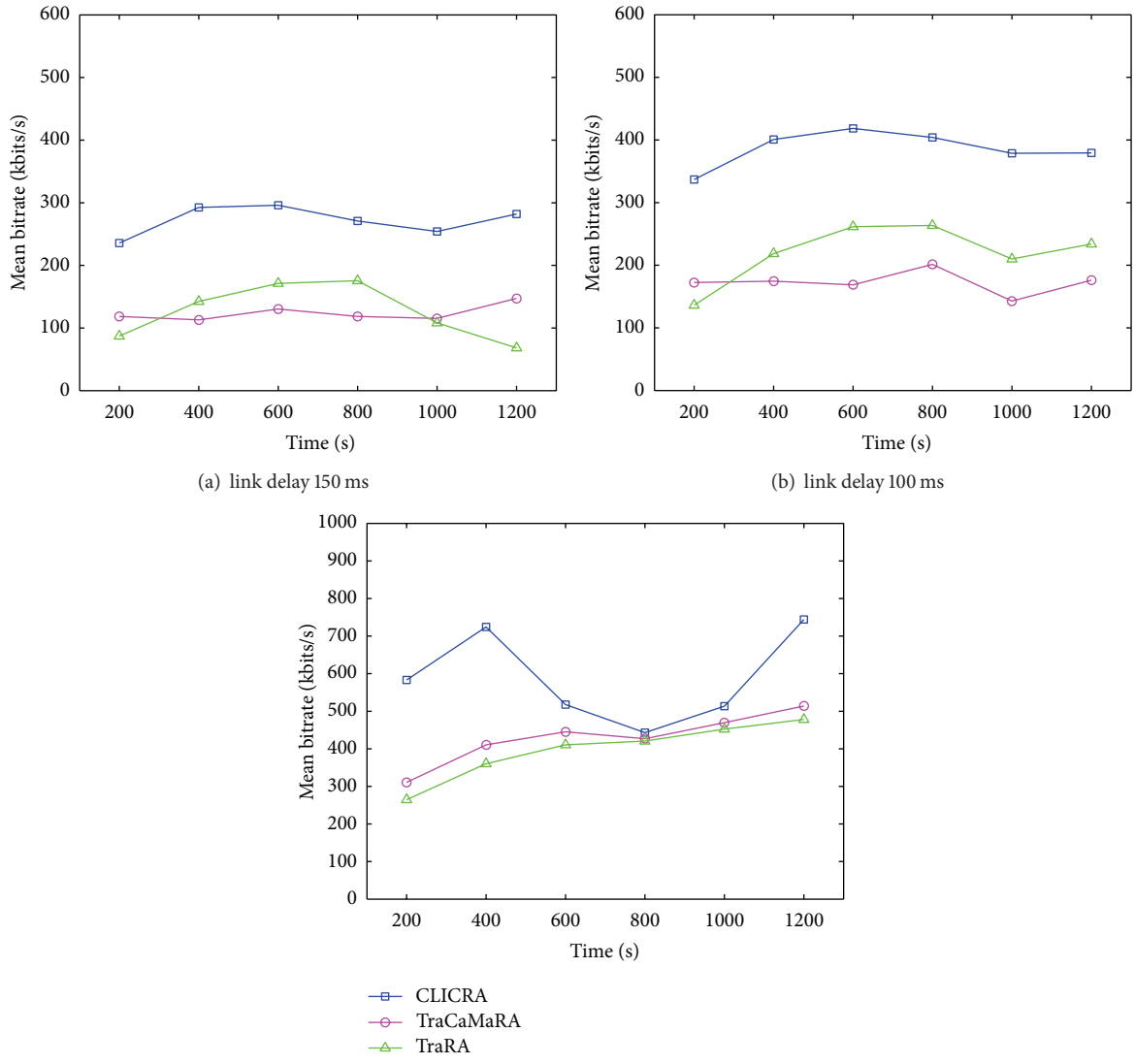
Bandwidth	CLICRA	TraCaMaRA	TraRA
Setting #1	132.0	159.3	118.6
Setting #2	216.4	287.1	169.1
Setting #3	251.7	234.9	199.9

and the large bandwidth consumption caused by TraCaMaRA.

6.2.2. Simulation Results for Setup 2. Simulation results were reported for the three different delay combinations with the given bandwidth in the links #1, #2, and #3. The receive window size was set to 60 Kbytes; hence, the throughput is affected by delays of the links in addition to bandwidths. The results were reported in a similar fashion as in the previous section. Tables 6, 7, and 8 show the buffer underflow count, the number of transmitted bytes from the origin server to the proxy cache layer #2, and switching frequency and amplitude with the three different link delays. Figures 8 and 9 show the average received media bitrates over the simulation time and the average cache hit ratio at proxy cache #2 for clients' requests, respectively. The simulation results show that the proposed method is significantly superior to the traditional methods, similarly to the results of the first simulation setup. The reasons were the same as discussed in the simulation results for the first setup.

7. Conclusions

This paper showed that when streaming management in a proxy cache and rate adaptation in streaming clients



(a) link delay 150 ms

(b) link delay 100 ms

(c) link delay 50 ms

Figure 8: Received mean bitrate over simulation time for receive window 60 Kbytes.

Table 6: Buffer under flow count for receive window 60 Kbytes.

Link delay	CLICRA	TraCaMaRA	TraRA
150 ms	91	872	668
100 ms	44	741	371
50 ms	4	219	22

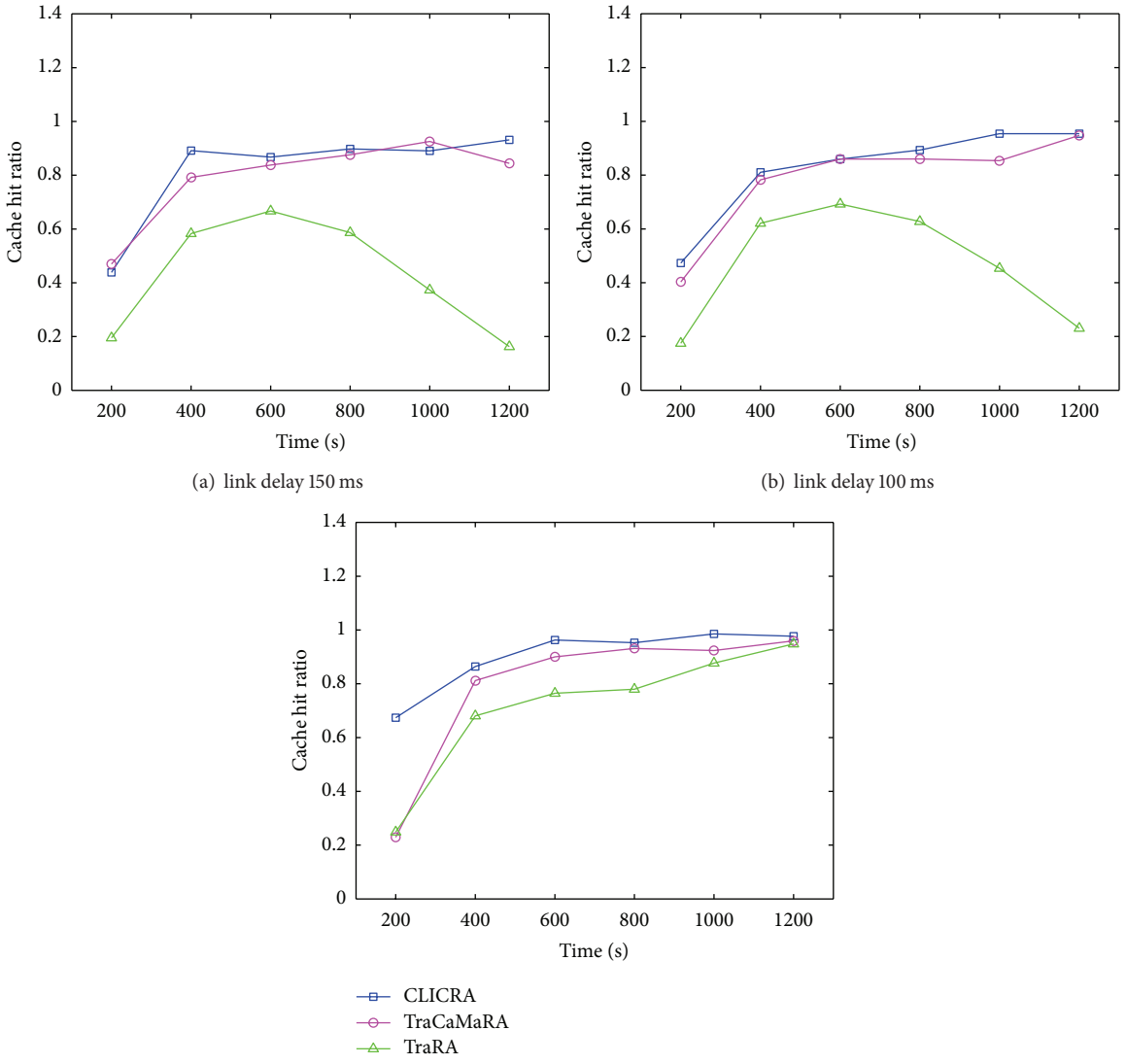
Table 7: Transmitted bytes (Gbytes) from the origin server to proxy cache.

Link delay	CLICRA	TraCaMaRA	TraRA
150 ms	118.9	121.0	102.2
100 ms	180.3	180.5	162.0
50 ms	271.6	353.8	219.7

using dynamic adaptive streaming over HTTP (DASH) are performed independently, network resources are not used efficiently and the streaming experience at DASH clients is degraded. To address these problems, this paper proposed a signaling method from a streaming client to a proxy cache to manage prefetching and caching of segments and from the proxy cache to the streaming client to feedback the caching status and the segment fetch time from the origin server to the proxy cache. Based on the signaling, the proposed client-driven joint proxy cache management and rate adaptation (CLICRA) method estimated the deliverable media bitrate between the origin server and the proxy cache and separately between the proxy cache and the streaming client. Hence, the fetching time and the buffered media duration can be predicted for subsequent segments according to the caching status and the estimated deliverable media bitrates. The proposed CLICRA method not only specifies the media bitrate of the representation level for the requested segments but also for the anticipated segments, which the client will most probably request shortly. The information

Table 8: Switch frequency and amplitude for receive window 60 Kbytes.

Link delay	Frequency			Amplitude		
	CLICRA	TraCaMaRA	TraRA	CLICRA	TraCaMaRA	TraRA
150 ms	0.1117	0.4712	0.4548	0.1388	0.6982	0.6502
100 ms	0.0906	0.3902	0.4892	0.1040	0.5810	0.6892
50 ms	0.0700	0.1596	0.2273	0.0744	0.2092	0.3024



(a) link delay 150 ms

(b) link delay 100 ms

(c) link delay 50 ms

Figure 9: Average cache hit ratio of the clients over simulation time for receive window 60 Kbytes.

of the anticipated segments is passed on to the proxy cache so that it can prefetch the anticipated segments. Therefore, the proposed CLICRA method efficiently uses the limited network resources between the origin server and the proxy cache and avoids congestion. CLICRA shows improved performance with respect to buffer underflow count, switching frequency, and achievable media bitrate at the clients and bandwidth consumption from the origin server to the proxy cache and cache hit ratio at the proxy cache compared to competing techniques.

Appendices

The Appendix is organized as follows. The detailed rate adaptation algorithm is presented in Appendix A, which describes the detailed realization of determining joint media bitrates for the requested segment and the anticipated segment presented in Section 5.2. Appendix B presents the caching status update and prediction which is summarized in Section 5.1.2. Finally, Appendix C proposes a method of separately measuring SFT which is used in Section 5.1.1.

A. Rate Adaptation

Rate adaptation actions (summarized in Section 5.2) consist of switching-up or down the representation level and keeping the representation level unchanged.

A.1. Switching-Up Representation Level. Two conditions are used to decide whether or not to switch-up to a higher bitrate encoded representation.

The first condition is to check whether or not the network capacity is large enough to deliver the *current* level of representation in time so that the buffered media time will remain in a safety level against playback interruption. The condition is that all predicted buffered media times at time instants $t_{\text{rec}}\ (\text{rid}_c, \text{snum})$, denoted as $\overline{B}_{t_{\text{rec}}(\text{rid}_c,\text{snum})}$, are higher than a certain threshold $\theta_B^{\uparrow}$ under the assumption that the DASH client sets rid to current representation ID rid_c; that is, the following inequality must hold:

$$\forall \text{snum} \in [\text{snum}_c, \text{snum}_c + w] : \overline{B}_{t_{\text{rec}}(\text{rid}_c,\text{snum})} > \theta_B^{\uparrow}. \tag{A.1}$$

The second condition is to check if the network capacity is large enough so that the *higher* level of representation is delivered in time and the buffered media time will remain in a safety level. This condition is to measure if network capacity is large enough so that the variation of the buffered media time is higher than a certain threshold compared to the current buffered media time. It is that all $\overline{B}_{t_{\text{rec}}(\text{rid},\text{snum})}$ must be higher than a certain threshold and variation of predicted buffer media time $\overline{BV}\ (\text{rid}, \text{snum}_c + w)$ must be higher than a certain threshold under the assumption that the DASH client switches-up to a higher bitrate encoded representation. In other words, if there is a higher level of rid within $[\text{rid}_c + 1, \text{rid}_c + \gamma]$, the following two inequalities must hold:

$$\exists\ \text{rid} \in [\text{rid}_c + 1, \text{rid}_c + \gamma] : \left\{\forall \text{snum} \in [\text{snum}_c, \text{snum}_c + w] : \overline{B}_{t_{\text{rec}}(\text{rid},\text{snum})} > \theta_B^{\uparrow}\right\}, \left\{\overline{BV}\left(\text{rep}, \text{seg}_{\text{rec}} + w\right) > \theta_{BV}^{\uparrow}\right\}, \tag{A.2}$$

where γ is window size of rid as defined in Table 1. rid is a variable instead of constant, which is different to the first condition where rid is constantly set to rid_c.

The proposed decision of switch-up to a higher bitrate encoded representation is made conservatively but quickly to provide stable and high playback quality as much as possible. The conservative decision is made by using multiple conditions as described earlier, and the quick decision is made by using predicted buffered media times for DASH operating in networks including proxy caches.

A.2. Switching-Down Representation Level. The decision of switching-down to a low bitrate encoded representation is made when the network capacity is decreased, and the *current* level of representation cannot be delivered in time. The late delivery results in that buffered media time will decrease to an unsafe level causing playback interruption. Late delivery may lead to variation of buffered media time compared to the current buffered media time which is lower than a certain threshold. The following condition must hold to switch down to a low bitrate encoded representation:

$$\forall \text{snum} \in [\text{snum}_c, \text{snum}_c + w] : \overline{B}_{t_{\text{rec}}(\text{rid}_c,\text{snum})} < \theta_B^{\downarrow} \text{ or } \overline{BV}\left(\text{rid}_c, \text{snum}_c + w\right) < \theta_{BV}^{\downarrow}. \tag{A.3}$$

The new representation level is set as the highest rid value within the range of $[\text{rid}_c - 1, \text{rid}_c - \gamma]$ if the following holds:

$$\text{rid} = \underset{\text{rid}\in[\text{rid}_c-\gamma,\text{rid}_c-1]}{\arg\max} \left\{\overline{B}_{t_{\text{rec}}(\text{rid},\text{snum})} \geq \theta_B^{\downarrow}\right\}. \tag{A.4}$$

Otherwise, rid of the requesting segment is decided as

$$\text{rid} = \underset{\text{rid}\in[0,\text{rid}_c-\gamma-1]}{\arg\max} \left\{br_{\text{rid}} < br_{\text{rid}_c} \frac{\text{MSD}}{\text{SFT}}\right\}, \tag{A.5}$$

where br_{rid} and br_{rid_c} denote the media bitrates with representation ID equals to rid and rid_c.

B. Caching Status Update and Prediction Process

The caching status update and consecutive prediction method (summarized in Section 5.1.2) includes the following processes.

Process 1. Caching statuses of subsequent segments is updated as those signaled from the proxy cache. rid is set to $\text{rid}_c - \gamma$, snum is set to snum_c, and $t_{\text{req}}\ (\text{rid}, \text{snum})$ is set to the current time.

Process 2. Emulate issuing a request for snum(th) segment at $t_{\text{req}}\ (\text{rid}, \text{snum})$ and the time to receive the segment, that is, $t_{\text{rec}}\ (\text{rid}, \text{snum})$, which can be predicted as

$$t_{\text{rec}}(\text{rid}, \text{snum}) = t_{\text{req}}(\text{rid}, \text{snum}) + \overline{\text{SFT}}(\text{rid}, \text{snum}). \tag{B.1}$$

Then, $t_{\text{req}}\ (\text{rid}, \text{snum} + 1)$ is set to $t_{\text{rec}}\ (\text{rid}, \text{snum})$. The scheduling method presented in [6] is deployed to request and receive segment sequentially. But, the idling period between two consecutive requests is set as in [11].

Process 3. Emulate signaling the earliest uncached "anticipated segment" information to the proxy cache if there is an uncached segment in several "anticipated segments" in a certain sliding window (w_f). The earliest one is conveyed to the proxy cache so that the latter can prefetch it. Therefore, if an earliest uncached "anticipated segment" information is conveyed to the proxy cache, then the client updates the caching status of the uncached "anticipated segment" as "being fetched" and estimates the time instant of caching the "being fetched" segments, that is, t_{cache}, which is predicted as

$$t_{\text{cache}}(\text{rid}, \text{snum}) = t_{\text{req}}(\text{rid}, \text{snum}) + \widetilde{\text{SFT}}_{s2p}(\text{rid}). \tag{B.2}$$

Process 4. Update the caching status of the "anticipated segments" from "being fetched" to "cached" if the time to request the next segment, that is, $t_{\text{req}}\ (\text{rid}, \text{snum}+1)$, is larger than $t_{\text{cache}}\ (\text{rid}, \text{snum})$.
Process 5. Increase snum and go to step 2 if $\text{snum} \le \text{snum}_c + w$.
Process 6. Increase rid and go to step 1 if $\text{rid} \le \text{rid}_c + \gamma$.

Processes 5 and 6 are used to consecutively check subsequent segments specified by snum and rid which are in the range of $[\text{snum}_c, \text{snum}_c + w]$ and $[\text{rid}_c - \gamma, \text{rid}_c + \gamma]$, respectively.

C. Measured SFTs

The measured SFTs from the origin server to the proxy cache and from the proxy cache to the DASH client, that is, $\widetilde{\text{SFT}}_{s2p}\ (\text{rid})$ and $\widetilde{\text{SFT}}_{p2c}\ (\text{rid})$, are used in the consecutive SFT prediction in Section 5.1.1. This appendix explains how $\widetilde{\text{SFT}}_{s2p}\ (\text{rid})$ and $\widetilde{\text{SFT}}_{p2c}\ (\text{rid})$ are measured by the DASH client.

C.1. SFT Measurement from the Proxy Cache to the DASH Client. The $\widetilde{\text{SFT}}_{p2c}\ (\text{rid})$ can be specified as SFT measured by the DASH client minus SFT from the origin server to the proxy cache, which depends on the caching status of the segment.

Each time after receiving a new segment, the recent sample of measured SFT from the proxy cache to the DASH client ($\widetilde{\text{SFT}}'_{p2c}$) is specified as follows.
Case 1. The received segment is a cached segment by the proxy cache, and the measured SFT by the DASH client ($\widetilde{\text{SFT}}^{c}$) is $\widetilde{\text{SFT}}'_{p2c}$; hence, $\widetilde{\text{SFT}}'_{p2c}$ can be set as

$$\widetilde{\text{SFT}}'_{p2c}\left(\text{rid}_c\right) = \widetilde{\text{SFT}}^{c}, \tag{C.1}$$

where $\widetilde{\text{SFT}}^{c}$ can be measured by the DASH client.
Case 2. The received segment is a being fetched segment by the proxy cache, and $\widetilde{\text{SFT}}'_{p2c}$ is determined as

$$\widetilde{\text{SFT}}'_{p2c}\left(\text{rid}_c\right) = \widetilde{\text{SFT}}^{c} - \left(t_{\text{cache}} - t_{\text{req}}\right), \tag{C.2}$$

where t_{req} and $\widetilde{\text{SFT}}^{c}$ can be measured by the DASH client, and t_{cache} is measured at the proxy cache and informed to the DASH client.
Case 3. The received segment is uncached, and $\widetilde{\text{SFT}}'_{p2c}$ can be specified as

$$\widetilde{\text{SFT}}'_{p2c}\left(\text{rid}_c\right) = \widetilde{\text{SFT}}^{c} - \widetilde{\text{SFT}}^{p}, \tag{C.3}$$

where $\widetilde{\text{SFT}}^{p}$ is measured at the proxy cache and informed to the DASH client.

Each time after receiving a new segment, $\widetilde{\text{SFT}}_{p2c}\ (\text{rid})$ is updated from the previous $\widetilde{\text{SFT}}_{p2c}\ (\text{rid})$ and the recent sample of $\widetilde{\text{SFT}}_{p2c}\ (\text{rid}_c)$, that is, $\widetilde{\text{SFT}}'_{p2c}\ (\text{rid}_c)$, as

$$\widetilde{\text{SFT}}_{p2c}\left(\text{rid}\right) = \alpha\widetilde{\text{SFT}}_{p2c}\left(\text{rid}\right) + \beta\widetilde{\text{SFT}}'_{p2c}\left(\text{rid}_c\right)\left(\frac{br_{\text{rid}}}{br_{\text{rid}_c}}\right), \tag{C.4}$$

where α and β are weighting factors of the previous estimate and the recent measurement sample, and br_{rid} and br_{rid_c} denote the media bitrates of representation with representation ID equals to rid and rid_c, respectively. Since the SFT is proportional to the encoded media bitrates, the scaling factor, that is, $(br_{\text{rid}}/br_{\text{rid}_c})$, is multiplied to represent $\widetilde{\text{SFT}}'_{p2c}$ in terms of rid. A segment typically contains several seconds of media duration (five seconds are used in this paper). This is already long in terms of providing an effective rate adaptation method to promptly react to bandwidth changes. So, α and β are both set to 0.5.

C.2. SFT Measurement from the Origin Server to the Proxy Cache. The idea of specifying $\widetilde{\text{SFT}}_{s2p}\ (\text{rid})$ is to use $\widetilde{\text{SFT}}^{p}$, which is measured at the proxy cache and informed from the proxy cache to the DASH client.

Each time after receiving a new segment, a recent sample of measured SFT from the origin server to the proxy cache ($\widetilde{\text{SFT}}'_{s2p}$) is specified as

$$\widetilde{\text{SFT}}'_{s2p}\left(\text{rid}\right) = \widetilde{\text{SFT}}^{p} \cdot \left(\frac{br_{\text{rid}}}{br_{\text{rid}_r}}\right), \tag{C.5}$$

where $\widetilde{\text{SFT}}^{p}$ denotes the measured SFT from the origin server to the proxy cache for the recently cached segment, and br_{rid_r} denotes the bitrate of the recently cached segment. The cached segment is considered as recently cached segment if the duration from start time to fetch a segment to the current time is less than or equal to a certain threshold (ε). The candidates consist of a list of segments having segment numbers in the range of $[\text{snum}_c - w_p, \text{snum}_c + w]$ and rid in the range of $[\text{rid}_c - \gamma, \text{rid}_c + \gamma]$. Then, $\widetilde{\text{SFT}}_{s2p}\ (\text{rid}_c)$, is estimated as

$$\widetilde{\text{SFT}}_{s2p}\left(\text{rid}\right) = \alpha\widetilde{\text{SFT}}_{s2p}\left(\text{rid}\right) + \beta\widetilde{\text{SFT}}'_{s2p}\left(\text{rid}\right), \tag{C.6}$$

where both α and β are also set to 0.5.

It may occur that $\widetilde{\text{SFT}}_{s2p}$ or $\widetilde{\text{SFT}}'_{s2p}$ in (C.6) does not exist when updating $\widetilde{\text{SFT}}_{s2p}$. For example, α (β) can be set as 1 if $\widetilde{\text{SFT}}'_{s2p}$(or $\widetilde{\text{SFT}}_{s2p}$) does not exist. In case both do not exist, $\widetilde{\text{SFT}}_{s2p}$ is estimated as a predefined value such as half of MSD. Similar process is applied to update $\widetilde{\text{SFT}}_{s2p}$ in (C.4).

If a segment which has not been fully received by the proxy cache but the scaled spent time for fetching the part of the segment as in (C.5) was already larger than $\widetilde{\text{SFT}}_{s2p}$, then

$\widetilde{\mathrm{SFT}}_{s2p}$ (rid) is updated according to (C.6) by considering the scaled spent time as $\widetilde{\mathrm{SFT}}'_{s2p}$.

In case there is no $\widetilde{\mathrm{SFT}}'_{s2p}$ for a certain period of time, for example, 60s as used in our simulation, the client estimates that the segments are continuously cached in a relatively long period of time at the current representation. The client indicates to the proxy cache the adjacent higher representation as the representation for the anticipated segments so that the proxy cache can prefetch the segment from the higher representation. So, $\widetilde{\mathrm{SFT}}_{s2p}$ (rid) can be updated even in case the current representation is fully cached.

References

[1] R. Fielding, J. Gettys, J. C. Mogul et al., "Hypertext Transfer Protocol-HTTP/1.1," RFC 2616, June 1999.

[2] T. Kim and M. H. Ammar, "Receiver buffer requirement for video streaming over TCP," in *Visual Communications and Image Processing (VCIP '06)*, Proceedings of SPIE, San Jose, Calif, USA, January 2006.

[3] "3GPP TS 26.234 Release 9: Transparent end-to-end packet-switched streaming service (PSS); protocols and codecs".

[4] "ISO/IEC 23009-1: Dynamic adaptive streaming over HTTP (DASH)-Part 1: Media presentation description and segment formats," Draft International Standard, August 30, 2011.

[5] "3GPP TS 26.247 Release 10: Transparent end-to-end packet-switched streaming Service (PSS); Progressive download and dynamic adaptive Streaming over HTTP (3GP-DASH)," http://www.3gpp.org/ftp/Specs/html-info/26247.htm.

[6] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proceedings of the 2nd Annual ACM Multimedia Systems Conference (MMSys'11)*, pp. 169–174, San Jose, Calif, USA, February 2011.

[7] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj, "Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network," *Signal Processing*, vol. 27, no. 4, pp. 288–311, 2012.

[8] N. Färber, S. Döhla, and J. Issing, "Adaptive progressive download based on the MPEG-4 file format," *Journal of Zhejiang University Science A*, vol. 7, supplement 1, pp. 106–111, 2006.

[9] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Bitrate and video quality planning for mobile streaming scenarios using a GPS-based bandwidth lookup service," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '11)*, Barcelona, Spain, July 2011.

[10] S. Chen, B. Shen, S. Wee, and X. Zhang, "Designs of high quality streaming proxy systems," in *Proceedings of the 23 Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04)*, pp. 1512–1521, Hong Kong, China, March 2004.

[11] C. Liu, I. Bouazizi, and M. Gabbouj, "Parallel adaptive HTTP media streaming," in *Proceedings of the International Conference on Computer Communications and Networks (ICCCN '11)*, Maui, Hawaii, August 2011.

[12] R. Fielding, J. Gettys, J. Mogul et al., Eds., "HTTP/1.1, part 6: Caching," draft-ietf-httpbis-p6-cache-17 (work in progress), October 2011.

[13] "The network simulator Ns-2," http://www.isi.edu/nsnam/ns/.

[14] V. Jacobson, R. Braden, and D. Borman, "TCP extensions for high performance," RFC1323, May 1992.

[15] H. Zhang, Y. Zhang, and Y. Liu, "Modeling internet link delay based on measurement," in *Proceedings of the International Conference on Electronic Computer Technology (ICECT '09)*, Macau, China, February 2009.