

# Parallel Adaptive HTTP Media Streaming

Chenghao Liu<sup>1</sup>, Imed Bouazizi<sup>2</sup>, and Moncef Gabbouj<sup>1</sup>

<sup>1</sup>Tampere University of Technology  
Tampere, Finland  
chenghao{dot}liu{at}tut{dot}fi  
moncef{dot}gabbouj{at}tut{dot}fi

<sup>2</sup>Nokia Research Center  
Tampere, Finland  
imed{dot}bouazizi{at}nokia{dot}com

**Abstract**— Recent advances in adaptive HTTP streaming of 3GPP have paved the way to the development of media streaming over HTTP. The traditional HTTP streaming relies on a series of request segments and receive segments sequentially. In the current Internet media segments can be delivered through distributed networks. In such a scenario, the traditional series request-receive based adaptive HTTP streaming method is, however, unable to provide optimum streaming since the distributed networks resources are not fully utilized. In this paper, we present a novel parallel adaptive HTTP streaming method. Compared to the traditional series of adaptive HTTP streaming technique, our method a) enables the receiver to request multiple segments in parallel b) provides a solution to maintain a limited number of HTTP sessions for receiving segments in parallel and to determine when to start a new HTTP session to request the next segment c) can adapt media bitrates while receiving previously requested segments. Simulation results show that the proposed parallel adaptive HTTP streaming method outperforms the traditional series of adaptive HTTP streaming with respect to providing a higher playback media quality and decreasing the interruption frequency of media playing.

**Keywords**—adaptive HTTP streaming; dynamic adaptive streaming over HTTP; rate adaptation; 3GPP PSS; distributed media delivery; multimedia streaming

## I. INTRODUCTION

Recently, adaptive HTTP streaming is specified in 3GPP Packet-Switched Streaming (PSS) [1] group. HTTP/TCP is easy to configure and is typically granted traversal of firewalls and address translators, which makes it attractive for multimedia streaming applications. In spite of the above advantages, the congestion avoidance algorithm of TCP results in a saw-tooth shaped instantaneous transmission rate. Additionally, the extreme reliability of TCP results in excessive and variable transmission delays due to retransmissions and in-order delivery. As a result, it was widely accepted that TCP is not adequate for multimedia streaming, which is delay sensitive but to some extent loss tolerant. Later, the authors of [2] reported that the instantaneous transmission rate and transmission delay variation of TCP can be smoothed out by receiver-side buffering. Moreover, a dominant share of multimedia traffic is being delivered using HTTP/TCP.

In 3GPP PSS adaptive HTTP streaming, the client continuously requests media segments each of which contains certain duration of media data. Adaptive HTTP streaming reduces the start-up delay dramatically as small media segment sizes are used. Furthermore, the rate adaptation can easily be supported by requesting media segments encoded with

different bitrates to achieve rate adaption. It is one of the key advantages of the adaptive HTTP streaming compared to the current web streaming. In the current web streaming, media bitrates are selected manually by the user from a set of pre-defined quality levels of a video clip. However, the manual selection of quality levels may result in the frequent interruption in playback or inefficient usage network resources because of inappropriate quality level selection.

In the current 3GPP PSS, the informative instructions of the client behavior are provided for the adaptive HTTP streaming. A few rate adaptation methods were reported in the past couple of years. Paper [3] proposed a rate adaptation method for adaptive HTTP streaming, which detects bandwidth changes using a smoothed HTTP throughput measured based on the segment fetch time (SFT). Based on the smoothed throughput measurement, paper [3] adapts the bitrates to the estimated bandwidth for the adaptive HTTP streaming. Paper [4] presents the buffered media time and fragment inter arrival time based bitrates switching to address the issue of bit rate switching in HTTP-based adaptive video streaming for wireless access networks. In paper [5], an experimental evaluation of rate adaptation algorithms in adaptive streaming over HTTP is presented. In addition, the scalable video coding (SVC) based adaptive HTTP streaming method is discussed in paper [6]. However all of the above mentioned papers [3] [4] [5] focus on the rate adaptation algorithm in order to match the media bitrates to the end-to-end bottleneck capacity.

In adaptive HTTP streaming, the scheduling of requesting segments plays a critical role in the achievable media bitrates of adaptive HTTP streaming. However, the specific scheduling method of requesting segments has not been given in the current 3GPP PSS. To the best of our knowledge, few research works have been conducted in this field. In our previous work, segments are requested and received sequentially in [3]. In the distributed media delivery networks [7], the series of requesting and receiving based adaptive HTTP streaming method cannot provide the optimum streaming service to end user. When the distributed networks are utilized, the requested segments are delivered to the end user through different passes. Thus transmitting the different segments in parallel will not affect the segments until transmitting through the last mile of the Internet, starting from the Internet connector and ending at the end user. In such a scenario, the distributed network resources cannot be fully utilized by using the traditional series of requesting and receiving segments method. This is due to the fact that only one HTTP streaming session operates for the delivery of media segments over the distributed networks in the traditional method.

This work was supported by the Academy of Finland, (application number 129657, Finnish Programme for Centres of Excellence in Research 2006-2011).

Content Distribution Network (CDN) is one of the most successful distributed commercial networks. The CDN [8] distributes the web and multimedia content to the multiple surrogate servers distributed in the networks from which the end users receive the requested data. As the surrogate servers close to the end users, CDN has the advantages of saving bandwidth and reducing delay perceived at the receiver side. In addition, the CDN enables to allocate the storage and bandwidth to multiple surrogate servers. However, due to expensive network resources in the CDN, the achievable bandwidth for media segments delivery is still limited. Furthermore, more and more end users are connected to the Internet using high-bandwidth. In the distributed media delivery over the Internet, congestion not only occurs in the last mile but also in the middle of the distributed networks. Therefore, effectively utilizing the distributed network resources is one of the key factors for the adaptive HTTP streaming to provide high quality media streaming. The parallel HTTP streaming is an essential and important feature for enabling the receiver to effectively make use of distributed network resources. To the best of our knowledge no parallel adaptive HTTP streaming method was presented in the literature in order to efficiently utilize the bandwidths in the distributed surrogates.

In a multiple TCP connection based media streaming, requesting a set of media chunks using multiple TCP streams was presented in [9][10]. Multiple TCP connection based media streaming was previously presented in [11]. The work in [9][10] extended multiple TCP connections based media streaming to the request-response based Internet video streaming. Although this paper uses the parallel HTTP requests, the aim and underlying principles are totally different from those in [9][10]. The aim of multiple TCP connections is to provide resilience against short-term insufficient bandwidth by using multiple TCP connections for a streaming application. But the target of our paper is to fully utilize the distributed network resources, for example CDN, to improve the HTTP-Streaming quality at the client. The underlying principle of the works [9][10][11] is that by allocating a certain bottleneck bandwidth to multiple TCP streams, the TCP throughput variation can be smoothed out and the achievable TCP throughput can be increased to some degree as the multiple TCP streams do not observe packet losses at the same time. In contrast, our paper is inspired by the distributed infrastructures of CDN.

In this paper, we propose a method of parallel adaptive HTTP streaming method in order to solve the above mentioned problems. The proposed method a) keeps a limited number of parallel threads for fetching segments over HTTP and b) determines the time for opening a new parallel request in order to improve the average receiving media bitrates and reduce playback interruption frequency. In addition, an advanced rate adaptation method is proposed for the proposed parallel adaptive HTTP streaming.

The rest of the paper is organized as following. Section 2 gives a brief overview of adaptive HTTP streaming in 3GPP PSS. The proposed parallel adaptive HTTP streaming method is presented in Section 3. Simulation results and conclusions are presented in Sections 4 and 5, respectively.

## II. ADAPTIVE HTTP STREAMING IN 3GPP PSS

In this section, we give a brief overview of adaptive HTTP streaming specified in 3GPP PSS. Fig. 1 shows a 3GPP adaptive HTTP streaming system that delivers content from standard HTTP servers to an HTTP-streaming client and enables caching content by standard HTTP caches.

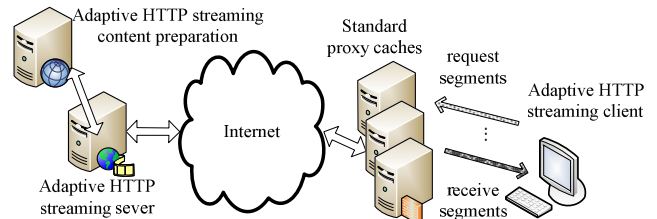


Figure 1. System of the adaptive HTTP streaming of 3GPP PSS

The HTTP-streaming, a client sequentially requests and receives the segments from the adaptive HTTP streaming server. To sequentially request media segments, the client accesses a Media Presentation Description (MPD) from the HTTP-streaming server. MPD contains the meta data to make request to fetch segment. The segment contains certain duration of media data and metadata to decode and present the included media content. 3GPP PSS adaptive HTTP streaming further enables the client to request media segments from different representations for reacting to varying network resources. Each representation consists of multiple media segments and represents the coding choice such as encoded bitrate, spatial and temporal resolution etc. And the representation is identified through a unique level identifier (ID) [1].

Fig. 2 shows an example of HTTP-streaming client behavior. The client adapts the bitrates ( $r_a$ ), requests a media segments (req #x) and receives segment (rec #x) one after another. The available time to fetch the segment #1 ( $\hat{f}t$ ), the minimum buffered media time ( $m_{min}$ ) and the current playback timestamp ( $t_0$ ) are labeled in Fig. 2. The current playback timestamp denotes the timestamp of the frame that is being played-out at the current time.

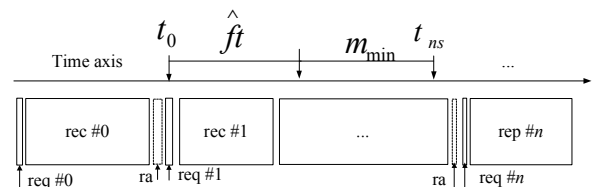


Figure 2. Traditional series of request and receive segments of the adaptive HTTP streaming

As shown in Fig. 2, the available time to fetch the next segment ( $\hat{f}t$ ) is determined as

$$\hat{f}t = t_{ns} - m_{min} - t_0 \quad (1)$$

where  $t_{ns}$  denotes the start timestamp of the next segment,  $m_{min}$  denotes the minimum buffered media time and  $t_0$  denotes the current playback timestamp at the time of requesting the next segment. The minimum media time can be buffered by fetching the next segment using the interval of  $\hat{f}t$  given in (1).

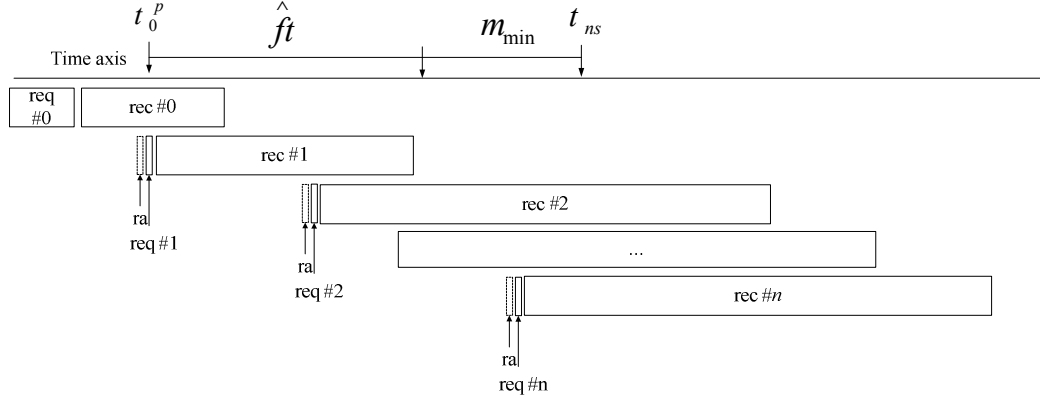


Figure 3. Proposed parallel requesting segments

In adaptive HTTP streaming, the client processes rate adaptation by identifying the media bitrate to match with the end-to-end network capacity. The adaptation takes place each time before requesting a new media segment.

The HTTP-streaming server can be a standard web server to provide adaptive HTTP streaming to the HTTP-streaming client. Content can be prepared in offline (static mode) or upon request (dynamic mode). In the static mode, the media presentation description (MPD) and representations are created before starting adaptive HTTP streaming sessions at the streaming server or content provider. In the dynamic mode, the server constructs the media segments based on the received HTTP requests [1].

### III. PROPOSED PARALLEL ADAPTIVE HTTP STREAMING METHOD

The proposed parallel adaptive HTTP streaming method and parallel rate adaptation method operated at the receiver are presented in this section.

#### A. Parallel adaptive HTTP streaming

Fig. 3 shows the parallel requesting and receiving segments over HTTP at the client. The HTTP-streaming client requests the first segment (req #0). While the client is still receiving the first segment (rec #0), it requests the next segment (req #1) in parallel. The second segment (#1) is expected to be consecutive to the first segment (#0) in terms of playout time. In the same way, the client requests the  $n$ th segment (req # $n$ ) while receiving one or more previous segments (rec # $i$   $i < n$ ) in parallel.  $\hat{f}t$ ,  $m_{min}$  and  $t_0^p$  are labeled in Fig. 3.  $t_0^p$  denotes the current playback timestamp at the time of requesting the next segment in parallel and the others are as in Fig. 2

As demonstrated in Fig. 3, the second segment (#1) is requested while receiving the first segment (#0). In contrast, the traditional method requests the second segment (#1) only after receiving the first segment (#0) as shown in Fig. 2. As the proposed method requests the second segment (req #1) earlier than the traditional method, the remaining time for fetching the second segment ( $\hat{f}t$ ) in the proposed method is longer than the available fetch time ( $\hat{f}t$ ) in the traditional method. Under the condition that the same available bandwidth in the distributed networks, the client with the parallel HTTP

streaming can achieve higher bitrates encoded segments than the bitrates of the traditional method. The advantage of the parallel adaptive HTTP streaming method is distinct in the case where there are limited bandwidths in distributed networks.

Fig. 4 shows the flowchart of the proposed parallel adaptive HTTP streaming method operated at the receiver. In the parallel HTTP streaming, a multiple HTTP session is operated each of which receives a segment. Here, the number of parallel receiving segments is denoted as  $n_p$ .  $n_p$  is compared with the predefined upper limits ( $u_p$ ) to limit the maximum number of parallel segments that a client can receive. In the parallel receiving segments, each segment is identified with a parallel index  $k$ , wherein the latest segment has the index 0 and the earliest segment has the index  $n_p - 1$ . As requesting a new segment in parallel, all segment indexes should be updated accordingly as mentioned above. The segment index is different with the segment IDs (# $x$ ) labeled in Fig. 2 and Fig. 3. In each receiving segment  $s_k$ , the portion of the segment received by the client is denoted as sub-segment  $ss_k$ .

In order to determine if request a new segment in parallel, we use the following:

$$\varphi_k = \frac{d_{ss_k}}{d_{s_k}}, \quad 0 \leq k \leq n_p - 1 \quad (2)$$

where  $d_{ss_k}$  denotes the duration of sub-segment  $ss_k$  received by the client and  $d_{s_k}$  denotes the duration of segment  $s_k$ . Here the duration denotes the media time included in a sub-segment or a segment.

The client compares  $\varphi_k$  with the corresponding threshold ( $\mu_k$ ) for each parallel receiving segment from index 0 to index  $n_p - 1$ .

In order to assist the receiver to receive the earlier segment in time,  $\mu_k$  is determined as

$$\mu_k = Ak + C \quad (3)$$

where  $A$  and  $C$  denote the slope and first term of the linear function, respectively, and  $k$  denotes the parallel segment index. Eq. (3) produces a higher threshold  $\mu_k$  for the parallel receiving segment having a larger index, i.e., earlier segment,

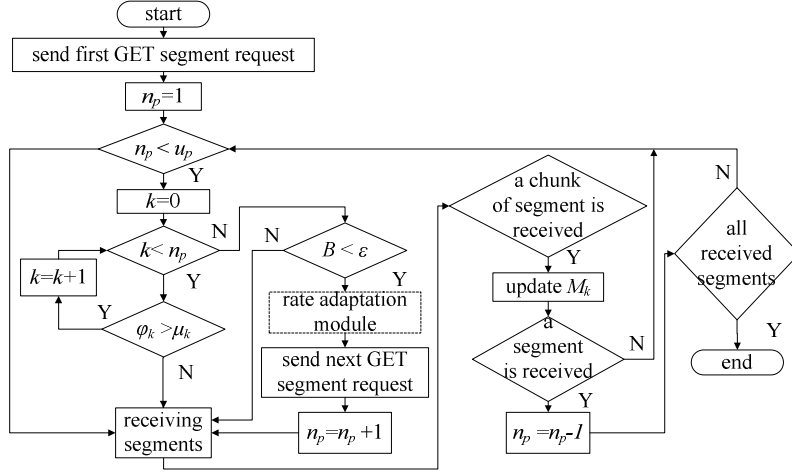


Figure 4. Flowchart of the proposed parallel adaptive HTTP streaming method

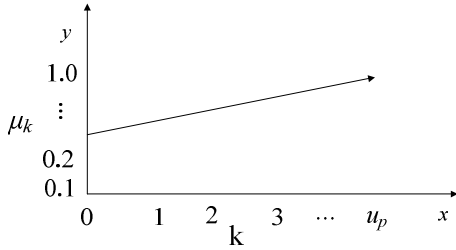


Figure 5. Thresholds of different segments with index  $k$ .

as shown in Fig. 5. Here the x-axis denotes the segment index  $k$  starting from 0 to  $u_p - 1$  and the y-axis denotes the threshold  $\mu_k$  varying between 0 and 1.

In Fig. 4, if  $\varphi_k$  is larger than the threshold  $\mu_k$  for all  $k$  from 0 to  $n_p - 1$ , then the buffered media time ( $B$ ) is compared with the upper limit  $\varepsilon$ . If  $B$  is smaller than the upper limit  $\varepsilon$ , then the client requests a new segment in parallel after performing the rate adaptation module. In the flowchart,  $n_p$  is increased by one, when a new segment is requested for receiving in parallel. In the parallel adaptive HTTP streaming, it should be prevented to request a new parallel segment when the buffered media time exceeds  $\varepsilon$  because of following two reasons. First, requesting too many parallel segments may cause pre-receiving too much media data which may result in bandwidth waste since the user may leave HTTP-streaming before playing-out all received media data. Second, receiving too many segments in parallel probably results in the sum of the bitrates of all parallel receiving segments exceeding the bandwidth of the last mile of the Internet even it is a high bandwidth. This will affect receiving the previously requested segments which are expected to be played-out earlier.

The upper limit  $\varepsilon$  can be set to cover the maximum draining of the buffered media time for fetching a segment due to the most severe bandwidth decrease. Here the most severe bandwidth decrease happens when the bandwidth decreases from the maximum bitrate to the lowest bitrate of all provided media representations. In the parallel adaptive HTTP streaming, if the bandwidth of the last mile of the Internet

suddenly decreases, then the maximum draining of the buffered media time should be further multiplied by the number of the parallel receiving segments number. Hence  $\varepsilon$  is set as

$$\varepsilon = d_{ns} \frac{br_{max}}{br_{min}} u_p \quad (4)$$

where  $d_{ns}$  denotes the next segment duration,  $br_{max}$  and  $br_{min}$  denote the maximum and minimum bitrates of all provided representations, respectively, and  $u_p$  denotes the upper limit of the parallel receiving segment number. Such a  $\varepsilon$  is beneficial for reducing the playback interruption even in the case of most severe bandwidth decreases.

As shown in Fig. 4, the receiver does not request a new segment and continues to receive the previously requested segments in the following three cases. First, the parallel receiving segment number is larger than its upper limit. Second,  $\varphi_k$  is not larger than  $\mu_k$  in one of parallel receiving segment. Third, the buffered media time is larger than or equal to its upper limit.

With chunked transferring mode of HTTP, the server can split the segment into small chunks and transmits the chunks. The client can receive a segment chunk by chunk until receiving the whole segment. When a new chunk of a segment is received by the client, the client updates sub-segment duration of the segment. If a whole segment is received by the client, then  $n_p$  is decreased by one. Finally, if all segments are received then the HTTP streaming is ended.

#### B. Rate adaptation for parallel adaptive HTTP streaming

The rate adaptation module depicted in Fig. 4 is described in this section. To perform the rate adaptation, the available time to fetch the next segment  $\varepsilon$  is firstly estimated as

$$\hat{f}t = t_{ns} - m_{min} - t_0^p \quad (5)$$

where the  $t_{ns}$  denotes the start timestamp of the next segment,  $m_{min}$  denotes the minimum buffered media time and  $t_0^p$  denotes the current playback timestamp at the time of requesting the next segment in parallel.

Next, the receivable total bits during the available time to receive the next segment are estimated. This is estimated as the

available time to receive the next segment multiplied by the minimum bitrates for receiving the previous parallel segments. Finally the bitrate is estimated as the above mentioned receivable total bits divided by the next segment duration as follows

$$\tilde{B} = \frac{(t_{ns} - m_{min} - t_0^p) \min\left\{\frac{b_{s_k}}{f t_{s_k}}, k=0,1 \dots n_p-1\right\}}{d_{ns}} \quad (6)$$

wherein the  $\tilde{B}$  denotes the estimated bitrates of the next segments,  $t_{ns}$  denotes the start timestamp of the next segment,  $m_{min}$  denotes the minimum buffered media time,  $t_0^p$  denotes the current playback timestamp. In the parallel receiving segments,  $s_k$  denotes the segment with index  $k$ , wherein the segment index  $k$  starts from 0 and ends at  $n_p - 1$ . In (6),  $b_{s_k}$  denotes the received bits of  $s_k$ ,  $f t_{s_k}$  denotes the fetch time from requesting  $s_k$  to the current time, min operation operates on the multiple ratios with different indexes  $k$  starting from 0 and ending at  $n_p - 1$ , and  $d_{ns}$  denotes the next segment duration.

In order to prevent the buffered media time from gradually decreasing or increasing due to the slight mismatch between the estimated bitrates with the end-to-end network bandwidth bottleneck, the buffered media time difference in the current time and the last time to adapt the bitrates is used to adjust the available time to fetch the next segment as (7).

$$\tilde{B} = \frac{(t_{ns} - m_{min} - t_0^p + (B_{cur} - B_{last})) \min\left\{\frac{b_{s_k}}{f t_{s_k}}, k=0, \dots, n_p-1\right\}}{d_{ns}} \quad (7)$$

where  $B_{cur}$  and  $B_{prev}$  denote the buffered media time at the current time and last time to adapt the bitrate, respectively.

In the adaptive HTTP streaming, the available bitrates for the user to select from is informed to the receiver through MPD. The next segment bitrate is finally selected as the highest one from the bitrates which are lower than the estimated bitrates in (7).

#### IV. SIMULATION RESULTS

In order to demonstrate the efficiency of the proposed parallel adaptive HTTP streaming method, we compare the proposed method with the series adaptive HTTP streaming method presented in [3] and both methods are implemented in the ns2 [12].

For performing the rate adaptation, the server provides 10 sets of representations for users to process the rate adaptation wherein the bitrates vary from 100Kbits/s to 1000Kbits/s with a step of 100Kbits/s and the corresponding representation level starts from 0 and ends at 9.

Fig. 6 shows the network topology used in the simulation, wherein the bandwidths and the link delays are depicted. The simulation starts from 0s and ends at 1200s. To simulate the dynamics of the Internet, the Exponential (Exp) traffics are delivered from Exp sender 0, 1 to Exp receiver 0, 1, respectively. In addition, CBR traffic is added from time 400s to 800s to compete the network bandwidth from the edge server 0 and the Internet connector, hence the rate adaptation algorithms can be evaluated. The competition CBR traffic

bitrates are set from 0.3 Mbits/s to 0.6 Mbits/s with a step of 0.1 Mbits/s to evaluate the rate adaptation efficiency in the different volumes of competition traffics.

In our simulation, two layers of media distribution networks are deployed including the surrogate server 0, 1 of the first and edge server 0, 1, 2 and 3 of the second layer. As shown in Fig. 6, the media segments are delivered from the HTTP streaming server to surrogate servers of layer 1 and edge servers of layer 2, cache and finally to the HTTP streaming user.

The segments are evenly delivered in the distributed networks as follows. The even ID tagged and odd ID tagged segments are distributed to upper and lower parts of the first layer of surrogate servers, respectively. From the first layer of surrogate servers, the segments are evenly distributed to the edge servers of the second layer. Finally all segments arrive at the proxy and end users. The different segment distribution method may be deployed, however, this is out of the focus of this paper.

For locating the edge server from which the end user receives a segment, the end user fetches segment by sending request to the Domain Name System (DNS) which resolves the edge servers name and responds to the end user. Once DNS resolves the edge server's name, the user's request is issued to the edge server, which then requests segments from the upper surrogate server if necessary. The end user receives the requested segments from the DNS edge server redirected edge server, for detailed information please refer to [8]. For simplicity and to focus our work, it is assumed that the end user knows from which edge server to fetch each segment in advance according to the media segments distribution described in the above paragraph.

In our simulation, the upper limit of the parallel segment is selected as 2, segment duration is equal to 8s and the minimum buffered media time is 10s.

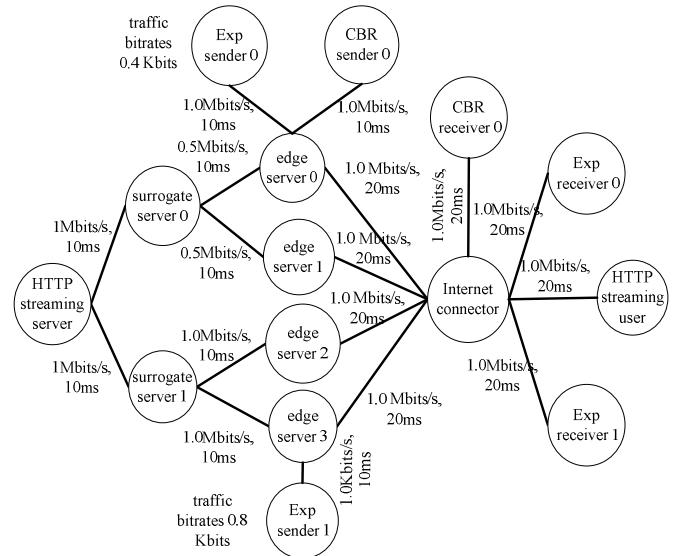


Figure 6. Network topology used in the simulations

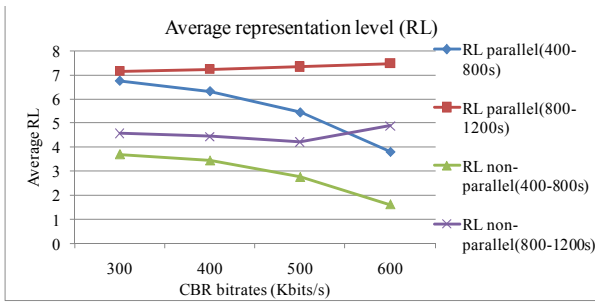


Figure 7. Average representation level with different CBR bitrates

Fig. 7 shows the average representations level, denoted as RL, of the proposed method and the series request-receive method, noted as non-parallel method, [3] with different CBR bitrates. In our simulation results, the whole simulation period is partitioned into before CBR traffic, during CBR traffic and after CBR traffic, corresponding to 0-400s, 400-800s and 800-1200s. The average RL of the proposed parallel adaptive HTTP streaming method exceeds the traditional non-parallel adaptive HTTP streaming on average 48.27% and 37.99% in the period of 400-800s and 800-1200s, respectively. In the period of 0-400s, the average RL of the proposed parallel method and non parallel method are 6.81 and 4.45 for the different CBR bitrates, respectively, achieving 34.65% improvement on the average RL.

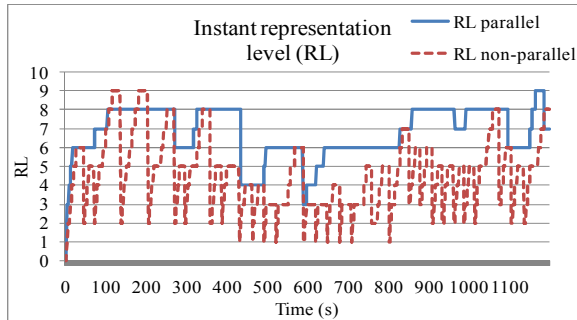


Figure 8. Instant representation level with time series

Fig. 8 shows the instant RL of the proposed parallel and traditional non-parallel method with the CBR bitrates of 0.5 Mbits/s. As shown in Fig. 8, the proposed parallel adaptive HTTP streaming method outperforms the traditional series adaptive HTTP streaming technique in the achievable instantaneous representation because of the following reason. The series adaptive HTTP streaming method cannot efficiently utilize the distributed networks' resources, such as the bandwidths in the surrogate servers and the bandwidths in the edge servers, since the series adaptive HTTP streaming method only uses one HTTP thread to deliver the media segments over the distributed networks. In contrast, the proposed parallel adaptive HTTP streaming method can more efficiently deploy the distributed bandwidths as the proposed method can use more than one parallel HTTP threads, to deliver the media segments through the different surrogate servers and edge servers. In order to prevent networks congestion in the last mile of the networks, e.g., from the Internet connector to the HTTP-

streaming client, the number of parallel HTTP threads is set not to exceed a predefined maximum level in the parallel adaptive HTTP streaming.

## V. CONCLUSION

In this paper, we presented a novel parallel adaptive HTTP streaming technique, which copes with the problem of inefficiency use of the distributed network resources by fetching media segment in parallel. The parallel HTTP streaming method enables a receiver to use relatively a longer period of time to fetch the segment compared to the traditional series technique. So the achievable media bitrates at the HTTP-streaming client can be enhanced with the proposed parallel adaptive HTTP streaming method. Simulation results show that the proposed method outperforms the traditional non-parallel adaptive HTTP streaming on average 40.3% of receiving bitrates.

## REFERENCES

- [1] 3GPP TS 26.234: "Transparent end-to-end packet-switched streaming service (PSS); protocols and codecs," (Release 9.5.0), Dec. 2010.
- [2] C. Krasic, K. Li, and J. Walpole, "The case for streaming multimedia with TCP," In Proceedings of IDMS, Lancaster, UK, Sep. 2001.
- [3] C. Liu, I. Bouazizi, M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," ACM MMSys 2011, Special Session: Modern Media Transport, Dynamic Adaptive Streaming over HTTP (DASH), San Jose, California, Feb. 23-25, 2011.
- [4] X. Qiu, H. Liu, D. Li, S. Zhang, D. Ghosal, B. Mukherjee, "Optimizing HTTP-based Adaptive Video streaming for wireless access networks," 2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), Beijing, China, Oct. 26-28, 2010.
- [5] S. Akhshabi, A. Begen, C. Dovrolis, "An experimental evaluation of rate adaptation algorithms in adaptive streaming over HTTP," ACM MMSys 2011, Special Session: Modern Media Transport, Dynamic Adaptive Streaming over HTTP (DASH), San Jose, California, Feb. 23-25, 2011.
- [6] Y. Sánchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, Y. Lelouedec, "iDASH: improved dynamic adaptive streaming over HTTP using scalable video coding," ACM MMSys 2011, Special Session: Modern Media Transport, Dynamic Adaptive Streaming over HTTP (DASH), San Jose, California, Feb. 23-25, 2011.
- [7] P. Frossard, J. C. D. Martin, R. Civanlar, "Media streaming with network diversity," Proceedings of the IEEE, vol. 96, no 1, pp. 39-53, Jan. 2008.
- [8] J. Dille, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, B. Weihl, "Globally distributed content delivery," IEEE Internet Computing, v.6 n.5, p.50-58, Sep. 2002.
- [9] R. Kuschnig, I. Kofler, H. Hellwagner, "Evaluation of HTTP-based request-response streams for internet video streaming," ACM Multimedia Systems Conference (ACM MMSys 2011), San Jose, California, Feb. 23-25, 2011.
- [10] R. Kuschnig, I. Kofler, and H. Hellwagner, "Improving internet video streaming performance by parallel TCP-based request-response streams," Proceedings of the 7th Annual IEEE Consumer Communications and Networking Conference (IEEE CCNC 2010), January 2010.
- [11] S. Tullimas, T. Nguyen, R. Edgecomb, S. Cheung, "Multimedia streaming using multiple TCP connections," ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), Vol. 4(2), 2008.
- [12] The network simulator Ns-2. Available [online]. <http://www.isi.edu/nsnam/ns/>.