

Rate Adaptation for Adaptive HTTP Streaming

Chenghao Liu
Department of Signal Processing,
Tampere University of Technology
Tampere, Finland
+358 5 0934 9231
chenghao.liu@tut.fi

Imed Bouazizi
Nokia Research Center
Tampere, Finland
+358 5 0486 0855
imed.bouazizi@nokia.com

Moncef Gabbouj
Department of Signal Processing,
Tampere University of Technology
Tampere, Finland
+358 3 3115 3967
moncef.gabbouj@tut.fi

ABSTRACT

Recently, HTTP has been widely used for the delivery of real-time multimedia content over the Internet, such as in video streaming applications. To combat the varying network resources of the Internet, rate adaptation is used to adapt the transmission rate to the varying network capacity. A key research problem of rate adaptation is to identify network congestion early enough and to probe the spare network capacity. In adaptive HTTP streaming, this problem becomes challenging because of the difficulties in differentiating between the short-term throughput variations, incurred by the TCP congestion control, and the throughput changes due to more persistent bandwidth changes.

In this paper, we propose a novel rate adaptation algorithm for adaptive HTTP streaming that detects bandwidth changes using a smoothed HTTP throughput measured based on the segment fetch time (SFT). The smoothed HTTP throughput instead of the instantaneous TCP transmission rate is used to determine if the bitrate of the current media matches the end-to-end network bandwidth capacity. Based on the smoothed throughput measurement, this paper presents a receiver-driven rate adaptation method for HTTP/TCP streaming that deploys a step-wise increase/ aggressive decrease method to switch up/down between the different representations of the content that are encoded at different bitrates. Our rate adaptation method does not require any transport layer information such as round trip time (RTT) and packet loss rates which are available at the TCP layer. Simulation results show that the proposed rate adaptation algorithm quickly adapts to match the end-to-end network capacity and also effectively controls buffer underflow and overflow.

Categories and Subject Descriptors

D.3.3 [Computer-Communication Networks]: Network Protocols – *Application (multimedia streaming)*.

General Terms

Algorithms, Measurement, Standardization.

Keywords

Adaptive HTTP streaming, multimedia streaming over TCP, rate adaptation, 3GPP PSS, TCP congestion control.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MMSys '11, February 23–25, 2011, San Jose, California, USA.
Copyright 2011 ACM 978-1-4503-0517-4/11/02...\$10.00.

1. INTRODUCTION

The current Internet is a best effort network; therefore, the network resources are characterized with varying available end-to-end bandwidth. In order to improve the user experience of multimedia streaming services, rate adaptation is used to prevent the client buffer from under-flowing and to achieve maximum possible playback quality. Rate adaptation may be performed at the sender, the receiver, or both. If the rate adaptation is performed by the server, it is categorized as sender-driven rate adaptation. The proposed algorithms in [6] and [7] can be categorized as a class of sender-driven rate adaptation. In [6], a rate adaptation algorithm over TCP is proposed which estimates the network bandwidth and client buffer occupancy using implicit feedback information built in the TCP congestion control. In [7], the authors propose an adaptive streaming algorithm for streaming of scalable video over UDP based on client buffer feedback. When streaming adaptation is performed by the client, it is classified as receiver-driven rate adaptation. One typical technique of this class is the receiver-driven layered multicast (RLM) [9]. In RLM, the server uses scalable video coding techniques to produce a set of layered bit streams and transmit each layer of the bit stream to a different multicast group. The receiver periodically joins multicast groups to probe the spare network capacity until it detects congestion. When multiple receivers observe packet loss, they conclude the network undergoes a congestion situation. However, the packet loss based congestion detection may not differentiate between losses due to congestion and link layer induced loss. To solve this problem, paper [8] proposes a multi-buffer based congestion control for multicast streaming of scalable video which uses the media time in the client buffer to detect congestion even before packet loss happens. However, the rate adaptation methods presented in [7], [8] and [9] are designed for multimedia streaming over UDP. The authors in [6] discuss the sender-driven adaptive streaming over TCP. Given that HTTP/TCP [1] is used for multimedia streaming, the sender is expected to be an HTTP server or a web cache, thus typically not keeping information about the receiver's connection state. Furthermore the sender-driven rate adaptation method has limitation in supporting the large-scale multimedia delivery since it will dramatically increase the burden on the web server or cache. Hence, it is expected that the rate adaptation in adaptive HTTP streaming will solely be receiver-driven. In this paper we examine the problem of receiver-driven rate adaptation for the application of the adaptive HTTP streaming.

Researchers recently revisited the fundamental question about the suitability of HTTP/TCP for delay-critical applications such as multimedia streaming. The dominant usage of TCP is mainly attributable to the congestion avoidance algorithm, which has so far ensured the scalable growth of the Internet. However, the

congestion avoidance algorithm of TCP results in a saw-tooth shaped instantaneous transmission rate. Additionally, the extreme reliability of TCP results in excessive transmission delays and delay jitter due to retransmissions and in-order delivery. As a result, it was widely accepted that TCP is not adequate for multimedia streaming applications, which are delay sensitive but to some extent loss tolerant. Despite this common understanding, a dominant share of multimedia traffic is being delivered using TCP nowadays. HTTP/TCP is easy to configure and is typically granted traversal of firewalls and network address translators, which makes it attractive for multimedia streaming applications.

Recent studies reveal that the instantaneous transmission rate variation of TCP called short-term throughput can be smoothed out by receiver-side buffering. In [5], the authors propose that the receiver side buffer can be used to smooth out the variation effect of TCP transmission rate. Furthermore, paper [4] discusses the receiver buffer requirement and presents an analytic expression of the minimum receiver buffer size to achieve the desired video quality. These research results show that interruption-free multimedia streaming over TCP can be achieved under the assumption that the network resources are not dynamically changing. Most of the current media streaming over web shortly called web streaming uses a similar approach, so called progressive download, to provide streaming services. In the current web streaming, such as provided by popular video portals, a set of pre-defined quality levels of a video clip is offered to the users for manual a-priori selection. Each level represents a specific definition and bitrate, and is henceforth called a representation. If the bitrate of the selected representation turns out to be higher than the available end-to-end bandwidth, then the user will most probably experience playback interruptions and re-buffering events due to buffer underflows. Otherwise, if the bitrate of the representation is lower than the available network bandwidth, then the user will consume the content at a sub-optimal quality. Moreover, as the bandwidth capacity is higher than the representation bitrate, the client will be downloading the content at a faster pace, which could result in bandwidth waste if the user decides to stop watching the content (e.g. when zapping).

To solve the problems in the current web streaming, the 3GPP group recently standardized the adaptive HTTP streaming solution as part of Packet-switched Streaming Service (PSS) [12]. Adaptive HTTP streaming in 3GPP PSS follows a strategy of sequential requesting and receiving of small media chunks of the multimedia content, so-called media segments. 3GPP PSS adaptive HTTP streaming further enables the client to request media segments from different representations to react to varying network resources. Each representation consists of multiple media segments containing certain duration of media data and encoded at a specific bitrate [13]. The research problems in the adaptive HTTP streaming include the following aspects in addition to the common rate adaptation in media streaming. First, the rate adaptation method must deploy a metric to identify if the bitrate of a specific representation matches the available end-to-end bandwidth or not. This metric is expected to distinguish between throughput changes due to network bandwidth variations and those attributable to the congestion control and avoidance algorithm in TCP. To achieve efficient rate adaptation, the metric should identify any mismatch between the representation bitrate and the available end-to-end bandwidth quick enough in order to react promptly and reach the optimum representation level quickly. Second, the rate adaptation algorithm has to manage the client buffer in order to prevent buffer underflows and overflows,

since buffer underflows cause playback interruptions and overflows result in bandwidth waste. Third, the rate adaptation algorithm should be equipped with good convergence property and prevent hopping between neighbor media representations, especially when the available end-to-end bandwidth lies within the bitrate range of two adjacent representations. Fourth, the media segment duration needs to be set appropriately in order to minimize the HTTP overhead, thus minimizing the delay introduced by HTTP request processing and transmission and maximizing the adaptation speed.

In this paper, we propose a receiver-driven rate adaptation algorithm for adaptive HTTP streaming. For deciding switch-up or switch-down operations between different representations, a smoothed HTTP/TCP throughput measurement method is presented that compares the segment fetch time with the media playback time contained in that segment shortly media segment duration. A typical media segment contains 5-10 seconds which is sufficiently long to smooth short-term variations in the TCP throughput. For probing the spare network capacity a step-wise switch-up method is used to switch to a higher representation. Upon detecting network congestion, an aggressive switch down method is deployed to prevent playback interruptions. Possible switch up and switch down operations are assessed each time after receiving a media segment. In order to save network bandwidth and memory resources for the users, a method for determining the idle time between two consecutive GET requests for media segments is deployed, thus limiting the maximum amount of media pre-fetching.

The rest of this paper is organized as follows. Section 2 describes the adaptive HTTP streaming system. The proposed rate adaptation method for HTTP streaming is presented in section 3. Section 4 and 5 show the simulation results and conclusion.

2. ADAPTIVE HTTP STREAMING SYSTEM

In this section, we give an overview of the system specified in the 3GPP PSS Adaptive HTTP streaming solution [12] shortly denoted as 3GPP adaptive HTTP streaming.

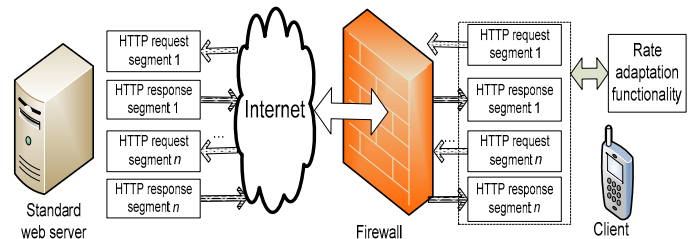


Figure 1. Adaptive HTTP streaming system

Figure 1 shows a 3GPP adaptive HTTP streaming system. As mentioned in the introduction, adaptive HTTP streaming operates as a set of sequential HTTP requests and responses.

The server can be a standard web server with the functionality to create media presentations as specified in 3GPP adaptive HTTP streaming. Creating media presentations may be done offline (static mode) or upon request (dynamic mode). In the static mode, the media presentation description (MPD) and representations are already created before starting any adaptive HTTP streaming sessions. In the dynamic mode, the server creates the media segments based on the received HTTP GET requests [12].

The client may send a series of GET requests, each of which requests a media segment of a representation that is identified through a unique level identifier (ID). In adaptive HTTP streaming, the client performs rate adaptation by identifying the representation that matches as closely as possible the end-to-end network capacity. In 3GPP adaptive HTTP streaming, the adaptation may take place each time before requesting a new media segment.

3. PROPOSED RATE ADAPTATION ALGORITHM

This section presents an advanced rate adaptation algorithm for HTTP streaming. Our algorithm compares the segment fetch time with the media duration contained in the segment to detect congestion and probe the spare network capacity. An effective rate adaptation algorithm is presented which adapts the bitrate by switching up/down between different representations each time after receiving a media segment and before sending the next request.

3.1 Smoothed HTTP/TCP Throughput Measurement

It is well known that the instantaneous TCP transmission rate is dynamically changing hence it is not feasible to measure the network capacity using the instantaneous TCP transmission rate. So instead, the client measures the segment fetch time, which covers a relatively long period of time, to determine if the bitrate of the current representation matches the available end-to-end bandwidth capacity. The segment fetch time (SFT) denotes a period of time from the time instant of sending a GET request for a media segment to the instant of receiving the last bit of the requested media segment.

In order to play media smoothly, the playing rate should be equal to the receiving rate in terms of media time. Thus if the encoded media bitrate of the current representation matches the end-to-end average TCP throughput, then the segment fetch time should be equal to the media segment duration. Otherwise, if the segment fetch time is larger than the media segment duration then it means that the average TCP throughput is lower than the bitrate of the current representation. Otherwise (if the segment fetch time is lower than the media segment duration), it indicates that the average TCP throughput is higher than the bitrate of the current representation. The last situation can occur in HTTP streaming because the TCP sender transmits the available data at the highest possible rate provided by the TCP congestion control and avoidance algorithm. Hence the ratio of media segment duration to segment fetch time denoted as μ is used as metric to detect congestion and probe the spare network capacity.

$$\mu = \frac{MSD}{SFT} \quad (1)$$

where MSD and SFT denote the media segment duration and the segment fetch time. The SFT measures how quickly the current segment is fetched on average.

Then the smoothed TCP throughput measurement can be estimated by multiplying μ with the media bitrate of the currently received segment. The receiver can obtain the encoded media bitrate of each representation from the Media Presentation Description (MPD). To produce the smoothed TCP throughput, the media segment duration shall be selected appropriately. Typically longer period is capable of producing smoother throughput measurement. However the longer period will cause

slower rate adaptation behavior. Based on our observations, media segments of around 10 seconds are basically sufficient to smooth out the varying instantaneous TCP transmission rate, and hence to produce the smoothed HTTP/TCP throughput measurement.

The advantage of using smoothed HTTP/TCP throughput measurement compared to the TCP throughput calculation equation in paper [10] is that our method does not require information from the transport layer (TCP layer). In order to use the TCP throughput calculation equation, the packet loss rates and round trip time (RTT) are required, however, such information is not available at the application layer. By contrast our method only needs to measure the segment fetch time. Therefore our method is feasible for application layer end-to-end rate adaptation.

3.2 Rate Adaptation Algorithm

In this section we present a rate adaptation algorithm based on the proposed smoothed HTTP/TCP throughput measurement presented in section 3.1. The smoothed HTTP/TCP throughput reveals the available network capacity and is suitable to be used as a metric of detecting network congestion and probing spare network resources. Fig. 2 shows the flowchart of the proposed rate adaptation algorithm for the adaptive HTTP streaming. The rate adaptation algorithm determines the representation for fetching the next media segment each time after receiving a media segment. The rate adaptation deploys a step-wise switch up and aggressive switch-down method to change the consumed representation from different bitrates encoded representations.

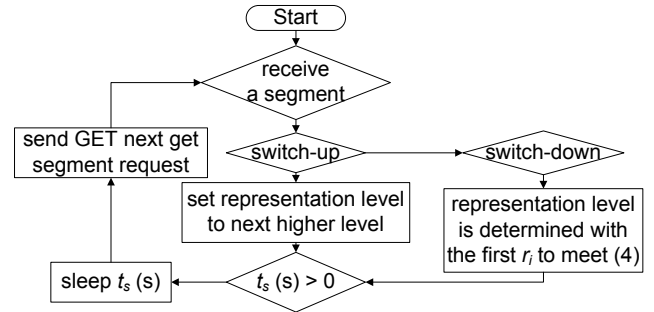


Figure 2. Flowchart of the proposed rate adaptation algorithm of the adaptive HTTP streaming

The switch up/switch down operations are determined as follows: Switch up: takes place if inequality (2) is met and the buffered media time is larger than the predefined minimum.

$$\mu > 1 + \varepsilon \quad (2)$$

where μ denotes the ratio of the media segment duration to the segment fetch time and ε denotes a switch up factor. In (2), the left term represents the metric to detect congestion and the right term denotes the condition to switch up to the next higher representation level. For determining the switch-up factor, it can be set as

$$\varepsilon = \max \left\{ \frac{b_{r_{i+1}} - b_{r_i}}{b_{r_i}}, \forall i = [0, 1, \dots, N - 1] \right\} \quad (3)$$

where b_{r_i} denotes the encoded media bitrate of representation i and N denotes the highest representation level.

As mentioned in section 3.1, the equation 2 is satisfied in the HTTP/TCP streaming scenario when the average TCP throughput for fetching a segment is higher than the encoded media bitrate of the fetched media segment. In case of a decision to switch up, the

rate adaptation algorithm selects the next higher representation level. The reason for using a conservative step-wise switch-up strategy is to prevent playback interruptions that might occur in case of aggressive switch-up operations. During the step-wise switch-up, the buffered media time is allowed to accumulate to a safety level in order to prevent buffer underflows that might be incurred by sudden bandwidth drops. So the initial buffering time, spent to reach the protection level of buffered media, is reduced significantly, which improves the user experience.

Switch down: It will be performed if inequality (4) is met.

$$\mu < \gamma_d \quad (4)$$

where μ represents the ratio of media segment duration to segment fetch time which is used as the rate adaptation metric and γ_d denotes switch down threshold. In case of congestion, the segment fetch time is typically much higher than the media segment duration. Hence, inequality (4) enables to detect network congestion before the media buffer is drained and switches down to a suitable representation as discussed in the following. In the case that (4) fails to detect slight mismatches between the media bitrates and the network capacity, the buffered media time may gradually decrease. Hence, buffered media time can be compared with a pre-calculated minimum, which is used as a complementary switch-down condition to prevent client buffer underflows.

In the switch down, an aggressive switch down will be performed. The selected representation level is determined to be the first representation (in descending order) with level r_i to meet

$$b_{r_i} < \mu b_c \quad (5)$$

where b_{r_i} denotes the encoded media bitrate of the representation r_i , μ denote the ratio of media segment duration to segment fetch time and b_c denotes the bitrate of current representation.

The idle time calculation algorithm is deployed before sending the next GET request, in order to prevent client buffer overflow. The rate adaptation algorithm will wait a certain period of time after determining the representation level of the next segment and before sending the next request if the buffered media time in the client buffer is large enough to cover the maximum draining of buffered media time during fetching the segment. When the average TCP throughput drops from the bitrate of the current presentation to the bitrate of the lowest representation, the maximum amount of buffered media time will be drained. So the idle time between determining representation level and sending the next request is set as t_s if the inequality (6) is met

$$t_s = t_m - t_{min} - \frac{b_c}{b_{min}} MSD > 0 \quad (6)$$

where t_s , t_m and t_{min} denote the idle time in seconds, the buffered media time, the predefined minimum buffered media time respectively and b_c and b_{min} denote the current representation bitrate and the minimum representation bitrate respectively, and MSD denotes the media segment duration. The key advantage of the idle time method is to limit the maximum amount of buffered media data; hence, saving network bandwidth consumption and memory resources of the receiver.

4. SIMULATION RESULTS

We implemented the proposed rate adaptation algorithm for adaptive HTTP streaming in ns2 [2]. Fig. 3 shows the network topology used in the simulations. The server and client denote the

HTTP streaming server and client. To simulate the varying delays and bandwidths an exponential traffic generator (Exp_G) and receiver (Exp_R) are used as background traffic with the average “on” time of 500ms, average “off” time of 500ms and the average sending rate during “on” times is set as 1000 Kbits/s during the “on” period hence the overall bitrate of the exponential traffic during whole period is equal to 500 Kbits/s. Both web server and exponential traffic generator start operation at time 0s and end at 1200s. In addition to background traffic a constant bitrate traffic generator (CBR_G) and receiver (CBR_R) are added at time 400s to 800s which is used as competitive traffic at the bottleneck bandwidth between node 0 and the proxy. In Fig.3 the bandwidth (Mbits/s) and delay (ms) are given for each link. For media data, 10 sets of representations are provided to perform the rate adaptation wherein the bitrates vary from 100Kbits/s to 1000Kbits/s with a step of 100Kbits/s and representation level 0 to 9 respectively. In the simulation, we set the MSD , t_{min} , and γ_d to 10s, 9s and 0.67.

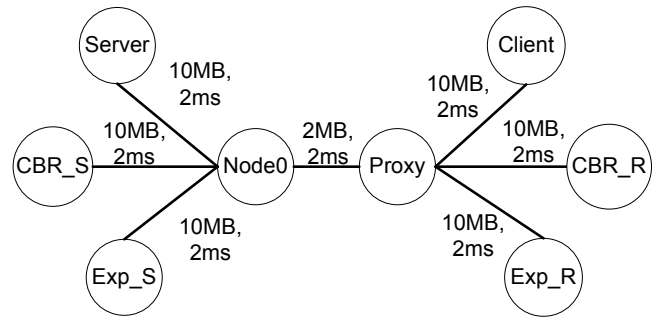


Figure 3. Network topology

To the best of our knowledge, we haven’t found a receiver-driven rate adaptation algorithm for HTTP streaming and only few research works have been conducted in the field of client buffer requirement for media streaming over TCP [4] [5] and TCP-based multimedia streaming performance analysis [11]. Hence, we evaluated the efficiency of the proposed method based on the aspect of rate adaptation accuracy and rate adaptation speed as follows. To identify the impact of the different variations of the bottleneck bandwidths on the proposed rate adaptation algorithm, we vary the bitrates of competitive traffic source, i.e. the CBR traffic generator, from 400 Kbits/s to 1400 Kbits/s at steps of 200 Kbits/s, hence 6 sets of simulation were ran. For evaluating the accuracy, we analyzed the representation level statistics together with the buffered media times after the representation first reaches a stable stage to identify how accurately the rate adaptation algorithm approaches to the network capacity and if it is capable to converge to a stable representation level. If the optimum representation level is n , then reaching representation level $n-1$ or $n+1$ is considered as reaching the stable stage. The rate adaptation speed is represented as the time spent to reach the stable representation level starting from the instant of changing the bottleneck bandwidth.

Fig. 4 shows the mean index of the consumed representations at the different CBR traffic bitrates, wherein the x axis denotes the CBR bitrates and y axis denotes the representation level. Here the representation level changes from 0 and 9 corresponding to the media bitrates of 100 Kbits/s to 1000 Kbits/s respectively. We partitioned the whole simulation period into three different periods including 0-400s, 400s-800s and 800s-1200s representing the period before CBR traffic appearing, the period during CBR traffic and after CBR traffic. The mean of representation level in

0-400 is constantly equal to 8.51 for all CBR bitrates since CBR traffic is not added until 400s. In the period of 400-800s, the mean of the representation level drops along with the increase of CBR bitrates. It shows that the rate adaptation effectively switches down to the lower representation level to match the media bitrates to the sharable end-to-end bandwidth. In the period of 800-1200s, when the CBR traffic disappears, the mean of representation level remains relatively constant with the different CBR bitrates and always higher than 8.5. This observation reveals that the performance of the rate adaptation is independent of the change in the bandwidth.

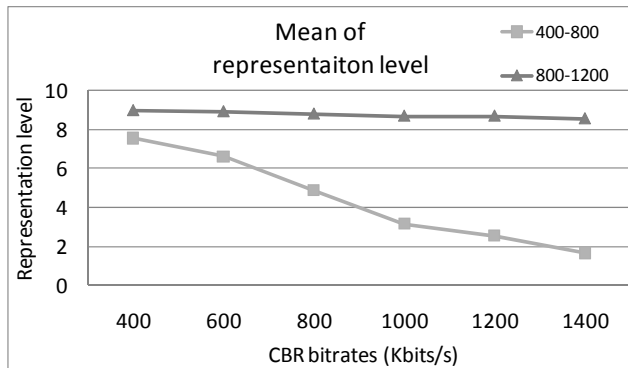


Figure 4. Mean of representation levels with CBR bitrates in different time periods

In order to show the convergence property of the proposed rate adaptation method after reaching the stable state, the standard deviation (STD) of the representations level at the stable state is depicted in Fig 5. The x and y axes are set similarly to Fig. 4. In the period 0-400s, the STD of the representation level is constantly equal to 0.26. In the period 800-1200s the STD is below 0.26 and during the period 400-800s the STD is below 0.7 except 0.99 for the CBR bitrate of 1000 Kbits/s.

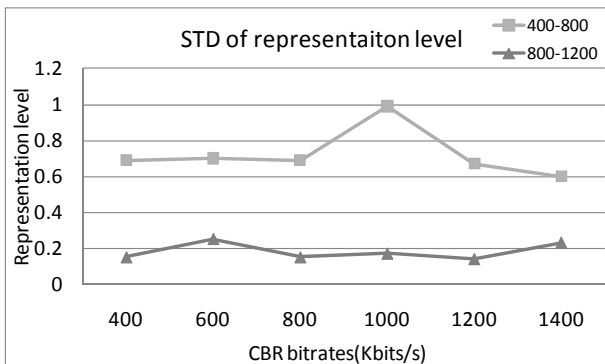


Figure 5. STD of representation level with different CBR bitrates in different time periods

In adaptive HTTP streaming, the optimum media bitrates can't simply be estimated as the fair share of the end-to-tend bottleneck bandwidth, since the supported media bitrates for interruption free streaming is also affected by the round trip time (RTT) and packet loss rates. To analyze how accurately the rate adaptation algorithm matches the selected representation bandwidth to the optimal level, the STD and the mean of the buffered media time at different CBR bitrates in different time periods are reported in Fig. 6 and Fig. 7 respectively. Here, the x axis denotes the CBR bitrates and the y axis denotes the buffered media time respectively. In HTTP streaming, the server sends the requested

media data at the highest transmission rate allowed by the TCP congestion and flow control algorithms. If the rate adaptation operates in a lower than optimal representation level, then the buffered media time will increase and vice versa. So the lower STD in the buffered media time demonstrates more accurate rate adaptation. In the period 0s-400s, the STD and mean of buffered media time are constant and equal to 9.9s and 65.5s. Fig. 6 shows that the maximum STD for the buffered media time is lower than 9.18s and 12.92s in the periods 400-800s and 800-1200s respectively. In Fig. 7, the minimum mean of the buffered media time is higher than 54s and 90s respectively in the periods 400-800s and 800-1200s respectively. As demonstrated in the simulation results, the STDs for the buffered media time are relatively small compared to the mean of the buffered media time. When the CBR traffic is added to compete on the bottleneck bandwidth, it is important to identify that the rate adaptation algorithm acts appropriately to prevent buffer underflows. In all of the simulation results the minimum buffered media time is higher than 36s, hence ensuring that playback interruptions do not happen.

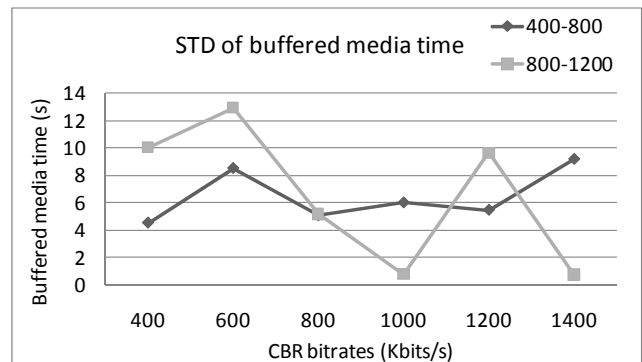


Figure 6. STD of buffered media time with different CBR bitrates in different time periods

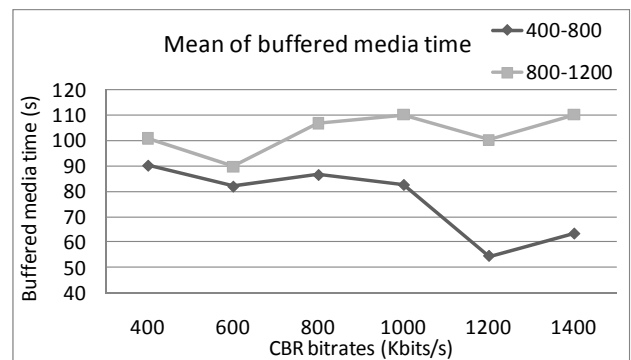


Figure 7. Mean of buffered media time with different CBR bitrates in different time periods

The rate adaptation speed is another important factor to evaluate the behavior of the rate adaptation algorithm. Fig. 8 shows the rate adaptation speed with different competition CBR bitrates in the different time periods. In the period of 0-400 (s) it takes 44s to switch-up to representation 8 from representation 0. The other switch-up period 800-1200 (s) shows that the rate adaptation speed varies with the amount of competitive traffic between a minimum of lower than 1s and a maximum of 41s. As shown in the period of 400-800 (s), the mean of switch-down speeds are around 30.3s and without showing any correlation with the amount of competitive traffic at the bottleneck. The convergence

time in down switching remains within the amount of buffered media time, thus avoiding any buffer underflows. It is worthwhile to note that the major part of the delay is attributable to waiting for the current media segment fetching to finish, which takes significantly longer when the available bandwidth drops.

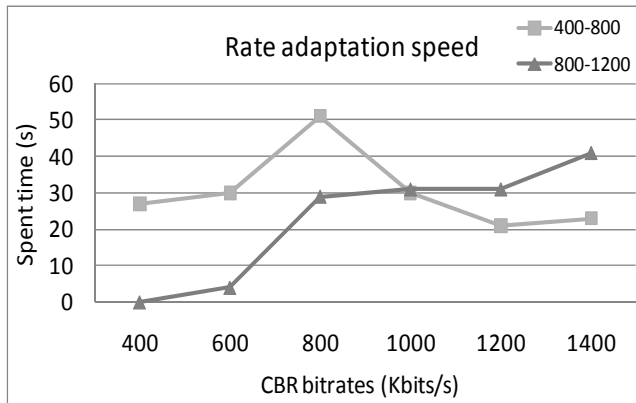


Figure 8. Rate adaptation speed with different CBR bitrates in different time periods

5. CONCLUSION

In this paper, we propose a novel method for detecting congestion, probing spare network capacity, and measuring the smoothed HTTP/TCP throughput for rate adaptation in adaptive HTTP streaming. The advantage of the proposed smoothed HTTP/TCP throughput measurement compared to the TCP throughput calculation equation used in TCP friendly rate control (TFRC) is that our method does not require the transport layer information such as packets loss rates and round trip time (RTT) to be available at the application layer. Hence, the proposed metric and smoothed TCP throughput measurement method can be used at the application layer. Upon detecting streaming that the media bitrate does not match the current end-to-end network capacity, an algorithm for conservative step-wise up switching and aggressive down switching of representations is presented using the smoothed TCP throughput measurement. In addition an idle time calculation method is used to prevent client buffer overflow and by consequence saving network bandwidth and memory resource at the client. Simulation results show that the proposed metric efficiently detects the congestion and probes the spare network capacity. And the smooth TCP throughput measurement method based rate adaptation method can quickly and accurately reach to the optimum bitrate level.

6. Acknowledgment

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

7. REFERENCES

- [1] Fielding, R., Getty, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Lee, T. Berners. 1999. Hypertext transfer protocol -- HTTP/1.1, RFC 2616. June 1999.
- [2] Information Sciences Institute, The University of Southern California. 2006. The Network Simulator - ns-2. (13 July 2006).
- [3] Kim, T. and Ammar, M. H.. 2006. Receiver buffer requirement for video streaming over TCP. SPIE VCIP 2006 (San Jose, CA, Jan. 2006).
- [4] Kim, T., Avadhanam, N., Subramanian, S. 2006. Dimensioning receiver buffer requirement for unidirectional VBR video streaming over TCP. ICIIP 2006 (Atlanta, USA, Oct. 2006).
- [5] Krasic, C., Li, K. and Walpole, J. 2001. The Case for Streaming Multimedia with TCP. In Proceedings of IDMS (Lancaster. UK, September 2001).
- [6] Lam, L.S, Lee, Jack YB, Liew, S.C, Wang W. 2004. A transparent rate adaptation algorithm for streaming video over the internet. In 18th International conference on advanced information networking and applications (Fukuoka, Japan, March 2004).
- [7] Liu, C., Bouazizi, I., Gabbouj, M. 2010. Advanced rate adaptation for unicast streaming of scalable video. IEEE International Conference on Communications 2010 (ICC 2010) (Cape Town, South Africa. May 2010).
- [8] Liu, C., Bouazizi, I., Gabbouj, M. 2010. Multi-buffer based congestion control for multicast streaming of scalable video. 2010 IEEE International Conference on Multimedia & Expo (ICME 2010) (Singapore, July 19-23, 2010).
- [9] McCanne, S., Jacobson, V., and Vetterli, M. 1996. Receiver-driven layered multicast. In the Proceedings of SIGCOMM'96. ACM Stanford, (CA, Aug. 1996), 117-130.
- [10] Padhye, J., Firoiu, V., Towsley, D. and Kurose, J. 2000. Modeling TCP Reno performance: a simple model and its empirical validation. IEEE/ACM Transactions on Networking, vol. 8, no. 2, pp. 133-145, April 2000.
- [11] Wang, B., Kurose, J., Shenoy, P., and Towsley, D. 2004. Multimedia streaming via TCP: An analytic performance study. In Proceedings of ACM Multimedia (October 2004), 908 - 915. <http://doi.acm.org/10.1145/1352012.1352020>.
- [12] 3GPP TS 26.234. 2009. Transparent End-To-End Packet-Switched Streaming Service (PSS): protocols and codecs. (Release9). http://www.3gpp.org/ftp/Specs/archive/26_series/26.234/.
- [13] 3GPP SP-090710. 2010. Adaptive HTTP Streaming in PSS. (Sophia-Antipolis, France, Jan. 2010.) <http://www.3gpp.org/ftp/Specs/html-info/26234-CRs.htm>.