HTTP-BASED SCALABLE VIDEO STREAMING OVER MOBILE NETWORKS

Ktawut Tappayuthpijarn[1,2]*, Thomas Stockhammer*[1]*, Eckehard Steinbach*[2]

[1]Nomor Research GmbH, Munich, Germany
[2]Institute for Media Technology, Technische Universität München, Munich, Germany
Email: ktawut@nomor.de, stockhammer@nomor.de, eckehard.steinbach@tum.de

ABSTRACT

This paper proposes an adaptive HTTP-based video streaming framework for mobile networks using the Scalable Video Coding (SVC) extension to the H.264 standard. We present a method to statistically estimate the channel at the mobile client and use it in our work to adapt the bit rate of the video. The adaptation takes into account both the quality contributions and the probability of successful timely decoding for different video segments. The simulation results show significant improvements in terms of a reduction of playback interruptions and improved perceived quality of service.

Index Terms— Mobile multimedia communication, Scalable video, HTTP-based video streaming

1. INTRODUCTION

Rapid deployment of the 3G and 4G mobile networks and a large variety of smart phones in the market have increased the popularity of mobile video streaming significantly. A large portion of that traffic is carried out over TCP due to several benefits, e.g, it is already a widely deployed protocol, compatibility with existing infrastructures and fewer problems with firewalls. However, the TCP throughput in a mobile environment is not only affected by congestion in the network, but also by variation in reception quality due to fading and interference as well. Given that TCP traffic is likely to be treated as best-effort with no QoS guarantee from the network, this fluctuation in throughput can severely degrade the streaming quality.

This paper proposes a TCP-based adaptive streaming framework using H.264/SVC designed for the mobile networks. The proposed architecture has a typical HTTP-style client-server relationship where the mobile client makes the adaptation decisions and requests video segments from the server. Additionally, it is compatible with the ongoing Dynamic Adaptive Streaming over HTTP (DASH) standard [1] which specifies how the media should be presented and retrieved, but not the adaptation algorithm itself.

Other works related to TCP-based video streaming are, e.g, [2], [3], [4], [5] and the references therein. [2] demonstrates the benefit of using SVC with TCP streaming to improve caching and storage efficiency of the servers. [3] and [4] propose an adaptation mechanism to adapt the number of enhancement layers in each frame. However, the algorithm resides on the server which can significantly increase its load. It also does not permit sending separated enhancement layers, when opportunities allow, to already-transmitted frames which are not due for decoding yet. This limits the adaptation options and the ability to exploit the full potential of the scalable video. This is also the case in [5] where the adaptation of the frame rate is done only for the currently considered part of the video.

The contributions of this work are as follows. First, it introduces a DASH-compatible adaptation engine for HTTP-based streaming with SVC which allows sending of additional enhancement layers later to improve any parts of the stream sent previously but are not decoded yet for a better RD performance. Second, a simple and accurate method to estimate a mobile channel throughput for making adaptation decisions is presented. Simulations with a Long Term Evolution (LTE) mobile network simulator show that our approach is able to follow changes in the throughput well. In addition, the algorithm runs at the client which reliefs the server's load in practical scenarios. It requires neither additional network node nor cross-layer activity, but only properly-prepared videos at the server and simple Round Trip Time (RTT) and throughput measurements at the client.

The rest of this paper is organized as follows. Section 2 describes the method to estimate the throughput of the mobile channel. Section 3 covers the preparation of the SVC videos while the adaptation algorithm is discussed in Section 4. Lastly, simulation results and conclusions are provided in Section 5 and 6, respectively.

2. ESTIMATING WIRELESS TCP THROUGHPUT

This section covers a method the client uses to estimate the probability of getting a number of bytes from a TCP connection over a wireless channel in a limited time. For a wireless channel based on the Carrier Sense Multiple Access (CSMA) technique, TCP is known to misinterpret channel losses as congestion-related losses and inappropriately reduces its transmission rate [6]. However, for the LTE-like technologies where there are several retransmission mechanisms at the MAC layer, channel losses are converted into larger delay and congestion-related losses when the base station's buffer is full instead. This justifies the TCP's behavior of reducing its transmission rate on losses. The streaming application can hence observe and estimate the channel from the long term throughput TCP provides.

In LTE, the allocation of small temporal-frequency radio resource units or "radio chunks" to mobile clients is done by a base station's resource scheduler at every 2ms Transmission Time Interval (TTI). Clients with good instantaneous channels are likely to be given more radio chunks in successive TTI's to take advantage of their brief favorable channels than others with worse channels, e.g, suffering deep fading. By measuring a lot of instantanoues throughput which reflects the channel and the scheduler's decision, the client should be able to infer its future throughput as well. Let the estimated amount of bytes the client gets in a period of T seconds be a random variable, denoted as B_T, and be thought of as a sum of smaller random variables B_τ where T is in a range of 5-20 seconds and τ is much smaller to have sufficient number of summands.

$$B_T = B_{\tau,1} + B_{\tau,2} + \ldots + B_{\tau,N} \; ; N = \lceil T/\tau \rceil \qquad (1)$$

However, setting τ to be too small, comparable to the TTI will result in high fluctuation of B_τ where most of them are close to zero and only a small fraction has large values representing a burst of data.

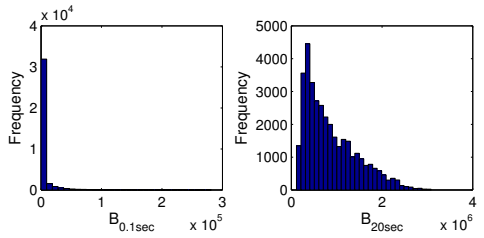

Fig. 1. Distribution of B_T for different time periods

Such B_τ shows signs of a heavy-tailed distribution with a large or infinite variance, thus renders the Central Limit Theorem (CLT) unusable to estimate B_T. In addition, successive B_τ's at such a short τ will show high correlation between them, e.g, they tend to have similar values, either very small due to fading or very large during a burst of data since the fast fading effect could last for several TTI's.

From experimental results, τ of 100 ms provides a good balance between having enough B_τ samples and reducing fluctuation and correlation among them since a B_τ covers as many as 50 TTI's. Thus a short-term effect at TTI level is partially averaged, although signs that B_τ is heavy-tailed are still present as shown in the left histogram of Figure 1. If B_τ is approximated by a heavy-tailed Log Normal distribution, B_T which is a sum of B_τ can also be approximated as Log Normal as well [7]. This is confirmed by the right histogram of B_T at 20 seconds in Figure 1 which shows its tendency toward a Log Normal distribution. Thus, $B_T \approx LogN\left(\mu, \sigma^2\right)$ where μ and σ^2 are defining parameters of the distribution to be estimated. [7] approximates them by pairing the mean and variance of B_T with those of the sum and solve for the μ and σ^2 as follows.

$$E\left(B_T\right) = E\left(B_{\tau,1} + B_{\tau,2} + \ldots + B_{\tau,N}\right) \tag{2}$$

$$e^{\mu+\sigma^2/2} = N \cdot E\left(B_\tau\right) \tag{3}$$

Similarly, equation (4) and (5) are from pairing the variances.

$$var\left(B_T\right) = var\left(B_{\tau,1} + B_{\tau,2} + \ldots + B_{\tau,N}\right) \tag{4}$$

$$e^{2(\mu+\sigma^2)} - e^{2\mu+\sigma^2} = Nvar\left(B_\tau\right) + 2\sum_{i<j} Cov\left(B_{\tau,i}, B_{\tau,j}\right) \tag{5}$$

Solving (3) and (5), the μ and σ^2 can be written as follows.

$$\mu = \ln\left[\frac{N^2E^2\left(B_\tau\right)}{\sqrt{Nvar\left(B_\tau\right) + 2\sum_{i<j} Cov\left(B_{\tau,i}, B_{\tau,j}\right) + N^2E^2\left(B_\tau\right)}}\right] \tag{6}$$

$$\sigma^2 = 2\left(\ln\left(N \cdot E\left(B_\tau\right)\right) - \mu\right) \tag{7}$$

The covariance terms in (6) which involve all combinations of $B_{\tau,i}$ and $B_{\tau,j}$ where $1 \le i < j \le N$ are the result of breaking up the variance of the sum of correlated random variables B_τ in (4). To avoid calculating all these terms, assume the correlation among $B_{\tau,i}$ and $B_{\tau,j}$ to be negligible once $|i-j| > 2$, e.g, they are more than 200 ms apart. Let $\Phi_1 = Cov\left(B_\tau, B'_\tau\right)$ and $\Phi_2 = Cov\left(B_\tau, B''_\tau\right)$ be the covariance between B_τ and the adjacent B'_τ and between B_τ and B''_τ that are two τ periods away. We have an approximation

$$\sum_{i<j} Cov\left(B_{\tau,i}, B_{\tau,j}\right) \approx \left(N-1\right)\Phi_1 + \left(N-2\right)\Phi_2 \tag{8}$$

From its recorded samples of B_τ, the client estimates the μ and σ^2 from (6), (7) and (8). The success probability Q of getting b_Q bytes within T seconds, denoted as $P\left(B_T \ge b_Q, T\right)$ or $P\left(b_Q, T\right)$ can then be derived from the Log Normal's Complementary CDF as

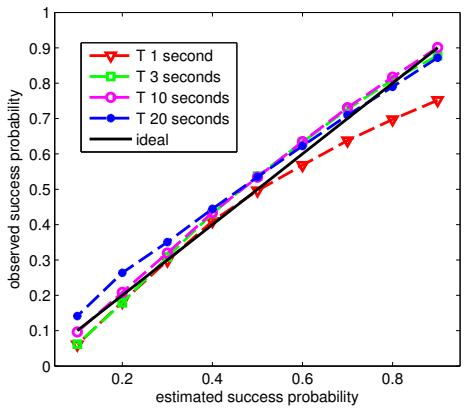

Fig. 2. Estimated and actual success probabilities

T	$Q=0.1$	$Q=0.2$	...	$Q=0.9$
T_1	b_{11}	b_{21}	...	b_{91}
T_2	b_{12}	b_{22}	...	b_{92}

Table 1. An example look-up table of b_Q with respect to Q and T

$$P\left(b_Q, T\right) = Q = \frac{1}{2} - \frac{1}{2}erf\left(\frac{\ln\left(b_Q\right) - \mu}{\sqrt{2\sigma^2}}\right) \tag{9}$$

$$b_Q = \exp\left(\sqrt{2\sigma^2} \cdot \chi_Q + \mu\right) \tag{10}$$

Here, $\chi_Q = erf^{-1}\left(1-2Q\right)$ depends only on the success probability Q and can be pre-computed beforehand. A look-up table of b_Q with respect to various Q and T similar to Table 1 can be quickly constructed every time the client needs to estimate the success probability. Note the relationship from (1) that $N = \lceil T/\tau \rceil$.

Figure 2 shows the accuracy of the estimation from simulations where the estimated success probability Q is plotted against the observed probability that the amount of bytes received within T seconds is no less than the corresponding b_Q. The result shows satisfactory accuracy where estimations for various values of T are most of the time very close to the "ideal" line. The settings for these simulations will be covered in more details later in Section 5.

3. CONTENT PREPARATION

The SVC extension of H.264 [8] allows video adaptation on-the-fly by adding or removing enhancement layers from it. Since the client must be able to request different layers separately, the video is first cut into smaller self-decodable units called "chunks" which can be, e.g, one or more Group of Pictures (GoP). Then smaller Network Abstraction Layer (NAL) units in each chunk are grouped further into "blocks" such that each block represents only a single layer of that chunk. This is shown in Figure 3 where a chunk of 1 GoP with 4 temporal and 1 Medium Grain Scalability (MGS) layers is cut into 8 blocks. To decode this chunk, at least block 0 for the lowest temporal scalability layer must be present. More enhancement blocks can be requested given that their "ancestry" blocks have been requested. In this example, temporal layers are requested before MGS layers due to quality improvement in having a higher frame rate is generally better than having more MGS layers. In addition, the RD information in terms of quality improvement, e.g, $\Delta PSNR$ that each layer contributes from its lower layer of each chunk must be generated during encoding and made available to the client as well.

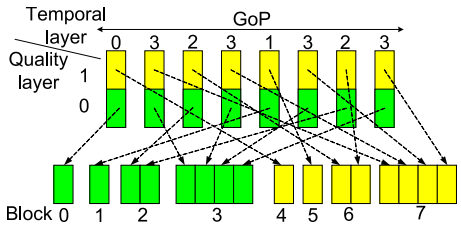

Fig. 3. An example on preparing video content into blocks

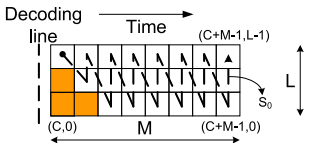

Fig. 4. An adaptation window of size $M \times L$

4. CLIENT'S ALGORITHM

4.1. Definitions

Let each block from a video with L layers be identified by a coordinate pair (c, l) where $c \in \{0, 1, \ldots\}$ and $l \in \{0, 1, \ldots, L-1\}$ denote the chunk and layer ID respectively. $B(c, l)$ is the size of the block and $Q(c, l)$ represents the quality improvement from its lower-layer block $(c, l-1)$ if $l > 0$ or the total quality if $l = 0$, e.g, it is the base-layer block. The client manages its receiving buffer by constructing an imaginary "adaptation window" of size $M \times L$ blocks as shown in Figure 4. The "decoding line" marks the chunk currently being retrieved and decoded from the buffer. The window is shifted forward continuously such that it always covers the next M chunks from the decoding line. In addition, let the simulation time t start from 0, the initial buffering time be T_{init} and the chunk period be T_p. Thus the time remaining for a block (c, l) in the window from the current simulation time until it is due to be decoded is

$$T_R((c, l), t) = T_{init} + cT_p - t \tag{11}$$

Let K be the number of unrequested blocks remaining in the adaptation window. A request sequence $\widehat{S}$ is defined as

$$\widehat{S} = \left\{(c, l)_i \,|\, (c, l)_i \text{ is missing}, i = 1, 2, \ldots, K\right\} \tag{12}$$

which is a set of all K missing blocks and represents the schedule the client will request for these blocks from the server sequentially based on the order of their appearances, denoted by a subscript i. The ordering of blocks in $\widehat{S}$ must also comply with their dependencies, e.g, a block $(c, l)_i$ can be requested if its missing ancestry blocks such as its lower-layer blocks or the base-layer blocks of the preceding chunks are requested in any of the $i-1$ positions before it in $\widehat{S}$.

At each decision instance, there are many possible request sequences the client can choose from. Thus, a quality metric for a sequence $\widehat{S}$ at time t is defined as follows to compare between them.

$$Q_{tot}\left(\widehat{S}, t\right) = \sum_{i=1}^{K} \left[Q\left((c, l)_i\right) P\left(B_i, T_R\left((c, l)_i, t\right)\right)\right] \tag{13}$$

Each individual summand is the quality contribution $Q\left((c, l)_i\right)$ of each block $(c, l)_i$ in $\widehat{S}$ weighted by its success probability in getting all the preceding $i-1$ blocks and itself within its decoding time as defined in (9) and (11). Here, B_i represents the total size of the first i blocks in $\widehat{S}$, e.g, $B_i = \sum_{j=1}^{i} B\left((c, l)_j\right)$. The success probability is obtained from a look-up table similar to Table 1 with M rows representing decoding times of all M chunks in the window.

for $n = 1 : K$ **do**
  foreach $(c, l)_p \in \widehat{I}_{\widehat{R}}$ *where* $p = 1, 2, \ldots, \left|\widehat{I}_{\widehat{R}}\right|$ **do**
    $\widehat{S}_n^p = \widehat{S}_{n-1}$;
    Move $(c, l)_p$ from its original location to the n^{th} location in $\widehat{S}_n^p$;
    Calculate $Q_n^p\left(\widehat{S}_n^p, t\right)$;
  end
  $\widehat{S}_n = \arg\max_{\forall \widehat{S}_n^p}\left(Q_n^p\left(\widehat{S}_n^p, t\right)\right)$;
  Update $\widehat{R} = \widehat{R} \cup \left\{(c, l)_n\right\}$ where $(c, l)_n \in \widehat{S}_n$;
  Update $\widehat{I}_{\widehat{R}}$;
end

Algorithm 1: The client's algorithm to determine $\widehat{S}_{opt}$

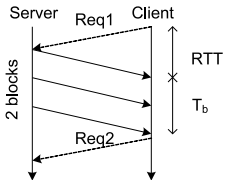

Fig. 5. RTT measurement by the client

4.2. Determining the best request sequence

First, define and initialize $\widehat{R}$ as a set of all received blocks in the window. Also, $\widehat{I}_{\widehat{R}}$ is a set of all blocks that can be immediately requested given that their ancestry blocks have been included in $\widehat{R}$. For example, shaded blocks in Figure 4 which represent the already-received blocks in the window are included in $\widehat{R}$ at initialization. The $\widehat{I}_{\widehat{R}}$ in this case is initialized to include blocks $(C, 2)$, $(C+1, 1)$ and $(C+2, 0)$ since their ancestry blocks are already in $\widehat{R}$. Finally, initialize a starting sequence $\widehat{S}_0$ to be a sequence that progressively requests all missing blocks of a chunk first before moving on to the next chunk. Figure 4 shows an example of $\widehat{S}_0$ with a dashed line.

The proposed Algorithm 1 works iteratively to find the best request sequence $\widehat{S}_{opt}$. At each step $n = 1, \ldots, K$, the best block to request at the n^{th} position of $\widehat{S}_{n-1}$ from the previous step will be selected out of all the blocks that can be immediately requested at this position, which by definition are the members of $\widehat{I}_{\widehat{R}}$. Denote these blocks as $(c, l)_p \in \widehat{I}_{\widehat{R}}$ where $p = 1, \ldots, \left|\widehat{I}_{\widehat{R}}\right|$ and $\left|\widehat{I}_{\widehat{R}}\right|$ is the total number of blocks in $\widehat{I}_{\widehat{R}}$. For each p, the algorithm tries moving a block $(c, l)_p$ from its original position to the n^{th} position of $\widehat{S}_{n-1}$ instead, denoted the modified sequence as $\widehat{S}_n^p$. The best block to be placed at this location, or equivalently the best modified sequence $\widehat{S}_n^p$ is the one that yields the highest metric in (13) and is selected to be $\widehat{S}_n$. This selected block is also added to $\widehat{R}$ and $\widehat{I}_{\widehat{R}}$ is updated for the next round. These steps continue until $n = K$ and the blocks to request at all K locations are selected, thus $\widehat{S}_{opt} = \widehat{S}_K$ is found.

However, the client should not request all K blocks in $\widehat{S}_{opt}$ at

once since the transmission for these blocks could last several seconds, during which time, the channel might have changed and a better adaptation decision could be made instead. On the contrary, requesting too few blocks from $\widehat{S}_{opt}$ would result in low channel utilization, especially if the RTT is relatively large compared to the transmission time of a block. Thus, the amount of blocks, or bytes to request is determined such that the approximated channel utilization exceeds a certain threshold, e.g, 80%. This requires the client to take regular measurements of the RTT between sending the request until the first byte of the reply arrives as in Figure 5. The utilization rate U where T_b is the transmission time of requested blocks is

$$U = T_b / (T_b + RTT) \tag{14}$$

The minimum bytes to request such that a target U_{min} is achieved is

$$B_{min} = (\text{minimum Tx period to achieve } U_{min})(\text{avg. throughput})$$
$$B_{min} = ((U_{min} \cdot RTT) / (1 - U_{min})) \cdot (E(B_\tau)/\tau) \tag{15}$$

5. SIMULATION RESULTS

Simulations of the proposed algorithm with a SVC-encoded video and of a non-adaptive streaming with an AVC-encoded video were conducted using the LTE mobile network simulator [9] which simulates the wireless channel of the LTE between an external server and a client connected to it. There were 8 best-effort clients in the cell including the streaming client walking randomly at 3 km/h which received no QoS guarantee from the network. The video used was a concatenation of the Crew, Mobile and Soccer clips. It was encoded into both a non-scalable AVC version at a bit rate of 1.7 Mbps and a scalable SVC at a bit rate from 1.2 to 1.9 Mbps at the same quality. The SVC version has 5 operating points with MGS scalability.

The client recorded the B_τ at τ of 100 ms in the last 20 seconds for throughput estimation. The T_{init} was set to 7 seconds and the adaptation window was of size 20×5 blocks. In addition, the client would stop the playback and wait for more data if its buffer was empty. Both the AVC and SVC simulations lasted for 30 minutes.

Figure 6 captures an instance from the simulation when the channel's Signal to Interference and Noise Ratio dropped due to the client being at the cell's edge. The number of layers for the SVC case was reduced as an adaption result to the decreasing throughput. On the contrary, the number of layer for the AVC case was always 1 since no adaptation was possible, causing longer playback interruption compared to the SVC case. Table 2 shows the average number of layers and the PSNR of the video, the percentage of interruption time and the Mean Opinion Score (MOS) for both cases. The MOS was computed from the Video Quality Metric (VQM) score which is an approximation of the user's perceived streaming quality [10] and scaled to be in 0-5 range where 5 is the best quality. Then the effects of interruption were modelled by applying a degradation factor as a function of the interruption time to the original MOS [11]. It is obvious that even though the resulting PSNR of the SVC case was slightly lower than the AVC due to layer removal, the SVC still achieved higher MOS as a result from having fewer interruptions.

6. CONCLUSIONS

This paper presents a method to estimate a mobile channel throughput and a HTTP-based adaptive streaming algorithm for SVC-encoded materials with any kind of scalability. The algorithm only requires properly prepared video contents and their RD information at the server. At the client, only basic throughput and RTT measurements are required. Simulations show satisfactory results both for the accuracy of the throughput estimation and the algorithm's adaptability to varying mobile channel throughput. This results in fewer playback interruptions and improved user's viewing experience.

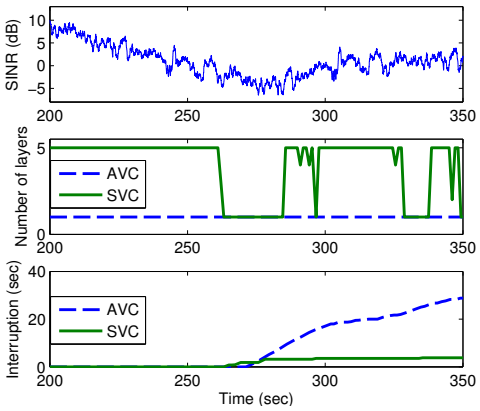

Fig. 6. Effect of degrading channel SINR on AVC and SVC

	No. of layers	PSNR(dB)	Interruption(%)	MOS
AVC	1	35.28	21.33	3.67
SVC	4.15	34.29	6.98	4.11

Table 2. Key performance indicators from simulations

7. REFERENCES

[1] ISO/IEC JTC 1/SC 29/WG 11 (MPEG), "Dynamic Adaptive Streaming over HTTP," Guangzhou, China, October 2010.

[2] F. Hartanto, J. Kangasharju, et al., "Caching Video Objects: Layers vs Versions?," in *Proc. IEEE ICME*, August 2002.

[3] P. Cuetos et al., "Adaptive Rate Control for Streaming Stored Fine-Grained Scalable Video," in *Proc. NOSSDAV*, May 2002.

[4] P. Cuetos, P. Guillotel, and K. W. Ross, "Implementation of Adaptive Streaming of Stored MPEG-4 FGS Video over TCP," in *Proc. IEEE ICME*, August 2002.

[5] N. Seelam, P. Sethi, and W. C. Feng, "A Hysteresis Based Approach for Quality, Frame Rate, and Buffer Management for Video Streaming using TCP," in *Proc. of MMNS*, 2001.

[6] Y. Tian, K. Xu, and N. Ansari, "TCP in Wireless Environments: Problems and Solutions," *IEEE Commun. Mag.*, vol. 43, pp. 32–47, March 2005.

[7] L. F. Fenton, "The Sum of Lognormal Probability Distributions in Scatter Transmission Systems," *IRE Trans. Commun. Syst.*, vol. 8, pp. 57–67, 1960.

[8] H. Schwarz, D. Marpe, et al., "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Trans. on CSVT*, vol. 17, pp. 1103–1120, September 2007.

[9] Nomor Research, "Network Emulator for Application Testing," www.nomor.de/home/solutions-and-products/.

[10] M. H. Pinson and S. Wolf, "A New Standardized Method for Objectively Measuring Video Quality," in *IEEE Trans. on BROADCAST*, September 2004, vol. 50, pp. 312–322.

[11] X. Tan, J. Gustafsson, and G. Heikkilae, "Perceived Video Streaming Quality under Initial Buffering and Rebuffering Degradations," in *Proc. MESAQIN*, June 2006.