FoG and Clouds: Optimizing QoE for YouTube

Tobias Hoßfeld[1,3], Florian Liers[2], Thomas Volkert[2], Raimund Schatz[3]
[1]University of Würzburg, Institute of Computer Science,Würzburg, Germany, hossfeld@informatik.uni-wuerzburg.de
[2]Technical University of Ilmenau, Ilmenau, Germany, [florian.liers;Thomas.volkert]@tu-ilmenau.de
[3]Telecommunications Research Center Vienna – FTW, Vienna, Austra, [hossfeld,schatz]@ftw.at

I. Introduction

Video streaming dominates global Internet traffic and is expected to account for 57% of all consumers Internet traffic in 2014 [1]. It can be distinguished between delivery of live video streaming with on-the-fly encoding, like IPTV or Facetime, and delivery of pre-encoded video, so called Video-on-Demand (VoD). The most prominent VoD portal is the YouTube cloud which accounts for more than two billion video streams daily. In order to attract users, the quality of experience (QoE) of the video playback is a very important criterion for such portals. YouTube QoE is different from traditional UDP-based video streaming, since it transmit the videos with TCP. Therefore, only the video playback itself is delayed while the transmitted audiovisual content remains unaltered. If available network data rate is lower than the video bit rate, video transmission becomes too slow, gradually emptying the playback buffer until underruns occur. Then, the user notices interrupted video playback, commonly referred to as *stalling*.

This work focuses on optimizing QoE for YouTube video streaming. We consider a bottleneck scenario, in which the available network data rate is limited to B. When downloading a video which is encoded with a certain video bit rate $V<B$, stalling may occur. However, stalling may also occur if the network data rate is sufficient on average to download the video during the playout time because of the variable video bit rate. To compensate such effects, a video player typically implements a video buffer. Thus, if the video is buffered long enough, no stalling will occur. From the end user's point of view, it is more convenient to experience no stalling at all during the playout even at the cost of an increased initial delay, than having small initial delays but also stalling [2]. The question arises how to set up the initial delay such that stalling occurs with low probability. In Section II, we model the impact of variable bit rate encoding on stalling. Then, we derive an approximation for the initial delay, such that stalling will likely not occur. Next, we discuss in Section III the interaction between application and network stack. In particular, we discuss how to exchange required information using the GAPI interface and propose the FoG ("Forwarding on Gates") stack as possible implementation solution which is currently work in progress.

II. Stalling vs. Initial Delays Considering QoE

A. Influence of Video Bit Rates on Stalling

Figure 1 shows the CDF of the video bit rate of typical YouTube videos. However, we distinguish here whether stalling occurs or not during the video playout when streaming over a bottleneck. In the experiment with B = 384 kbps, about 300 videos were completely downloaded from the YouTube platform and analyzed. No stalling occurs for 116 videos corresponding to 38.67%. In this case, the video bit rate is mostly smaller than the bottleneck data rate, $V<B$. However, there were two videos without stalling, although the video bit rate was significantly larger than B. These videos were quite short with a duration of 10.8s and 9.8s, respectively. In these cases, the filled video buffer was able to compensate for insufficient network data rates.

In addition, Figure 1 shows that for some videos with video bit rate $V<B$ stalling still occurs. In that case, stalling is caused by the variability of the video bit rate, which is illustrated in the following. The video files consist of two different types of frames, called "key frames" and "interframes". A key frame is spatially compressed and the main reference frame for the following interframes. The interframes are temporally compressed and significantly smaller than the key frame. Let us approximate the size of the key frames and the interframes by a normal distribution $K \sim NORM(\mu_K, \sigma_K)$ and $I \sim NORM(\mu_I, \sigma_I)$ with corresponding mean and standard deviation, respectively.

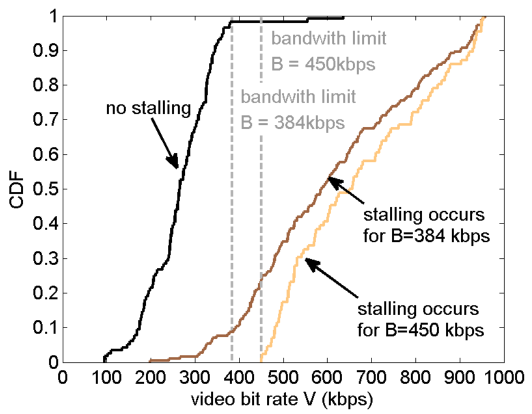

Figure 1. Stalling occurrence depending on video bit rate for two different bottleneck capacities

We assume that the video data is constantly delivered with bandwidth B = V and the video frames are played out with a constant frame rate, which is typically F= 30 frames/s. We further assume that all key frames are followed by N_i interframes. A key frame and its interframes form a so-called group of pictures (GOP). The entire video consists of N_G GOPs. Thus, the average video bit rate V follows as $V = 1/(N_I+1)(\mu_K + N_I \mu_I) F$ under the assumption that the random variables of the key frame and interframe sizes are independent. The video buffer status S indicates the amount of video data in the video buffer yet not played out. During the playout time of a frame $1/F$, V/F bytes of video data are downloaded at constant speed. Since the linear combination of independent normal random variables follows again a normal distribution, the video buffer status follows as $S \sim NORM(0, N_G(\sigma_K^2 + N_I\ \sigma_I^2))$. Although the network capacity is sufficient on average, i.e. B=V, stalling occurs with probability $P(S < 0)$ due to the variable bit rate of the video.

B. Approximation of Initial Delay

Another influence factor on stalling is the implementation of the buffer of the video player. The video player buffers W seconds of the video before playing out the video, i.e., video data of size VW is buffered with video bit rate V. The corresponding initial delay T_0 for buffering the video data follows as $T_0 = VW / B$. Consequently, stalling occurs, if the remaining video data, i.e. $DV - WV$, cannot be downloaded faster than the video lasts itself, $(DV\text{-}WV)/B>D$. However, due to the variable bit rate encoding, we further have to increase the initial delay to avoid stalling. We approximate the buffer size *C* with the normal distribution as described above and set the initial delay T_0 to be set to *S/C* for a network data rate *B*.

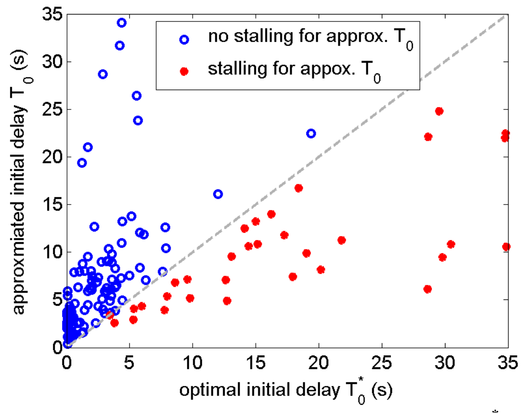

Figure 2. Scatter plot of the optimal initial delay T_0^* and the approximated initial delay T_0

Figure 2 shows the numerical results for the approximated initial delay T_0 compared with the optimal value T_0^* derived by analyzing the video file frame-by-frame. Although in most cases the approximation is close to the optimum, there are several cases (about 25%) which still lead to stalling. The reason behind this phenomenon lies in scene changes with the video clips. Thus, the required parameters for the approximation, i.e. mean and variance of key frame and interframe sizes, have to be specified for each scene to improve the approximation. This is only necessary for a fraction of the YouTube videos, which often consist only of a single scene (in terms of video encoding). Another option is to send the size for each frame or aggregated for *N* consecutive frames before the video is transmitted. This information enables the computation of the optimal initial delay. Since there is an upper limit on the duration of YouTube videos to be uploaded [3], which is now 15 minutes, the number of maximum frames per video is below 27,000 at a frame rate $F = 30$ *frames/s*. In addition, the parameter *N* adjusts the trade-off between signaling overhead and accuracy of the approximation.

III. Interaction between Network and Application

A. *Requirements for the transmission*

Based on the QoE consideration, the video player is able to derive its requirements for the video transmission. Foremost, it requires an average network data rate with a limited variance. Since the frames have to be shown in order and the player does not want to sort them by itself, they must be delivered by the network stack in order, too. If the transmission is done in order, is transparent for the application and can be decided by the network. Despite today's usage of error and loss free transmission via TCP, the video codec might be able to deal with some loss or bit errors. The maximum amount of bit errors or lost packets depends on the video codec and on the required QoE level.

Such detailed requirements cannot be passed to the network stack with today's APIs. Therefore, new APIs like the GAPI [4] lend itself. The GAPI was developed in the SIG Functional Composition of the German Lab Project [5] especially to provide applications a way to specify their requirements for communication associations. With the help of the GAPI function *Subscribe*, the player is able to specify the name of the server and its list of requirements.

Finally, the network stack must be able to react to these requirements dynamically. On the one hand, the stack must be able to buffer data locally, in order to sort them and to reduce the variance of the data rate. One the other hand, the network must be able to reserve data rates and to fast retransmit lost or corrupted packets. Both must be done in a scalable way in order to support the large amount of YouTube users.

B. *Forwarding on Gates*

One possible dynamic stack is provided by the "Forwarding on Gates" (FoG) framework [6]. It is a scaling inter-network system, based on dynamic composition of functional blocks. An application is enabled to define special requirements for a data transmission as described before. The network stack of FoG uses this information to select appropriate existing functions for the upcoming transmission. If needed functionality isn't available yet, FoG's routing directs each packet to the next intermediate node where new function instances can be placed in order to fulfill at least one of the desired transmission requirements. Packets are used as data input for the placed functions. In general, this system for placing functional blocks in the network can be used to direct packets through a chain of function instances, needed for video transcoding or buffering.

In addition to the creation of new instances, the system is also able to re-use existing function instances and their states for multiple connections in order to improve scalability. Finding existing and creating new function instances is done during the signaling process for setting up a communication association. The requirements are described in the header of the first signaling packet. A demo has shown that the re-use is possible without per-connection state information on the hosts which provide the desired functions [7]. This proof-of-concept for automatic function placement places functions on the first node along the communication route, whose policy allows this. A more sophisticated placement algorithm that places functionality with focus on potential reuse is developed in current work.

IV. Future Steps

Currently, we are able to derive the parameters for the video player and the network communication association theoretically. As concerns future work, we plan to setup a demo showing QoE-enabled video playback based on the GAPI and FoG. The different QoE levels of a YouTube video are in the main focus of this demo and will be compared in a live demonstration.

V. References

[1] Cisco Systems Inc.,:Cisco Visual Networking Index: Forecast and Methodology, 2010-2015, June 2011.

[2] T. Hoßfeld, R. Schatz, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia: Quantification of YouTube QoE via Crowdsourcing. IEEE International Workshop on Multimedia Quality of Experience - Modeling, Evaluation, and Directions (MQoE 2011), Dana Point, CA, USA, December 2011.

[3] T. Hoßfeld and K. Leibnitz: A Qualitative Measurement Survey of Popular Internet-based IPTV Systems. Second International Conference on Communications and Electronics (HUT-ICCE 2008), Hoi An, Vietnam, June 2008.

[4] F. Liers, T. Volkert, D. Martin, H. Backhaus, H. Wippel, E. Veith, A. A. Siddiqui, R. Khondoker: GAPI: A G-Lab Application-to-Network Interface, 11th Würzburg Workshop on IP (EuroView), Germany, Würzburg, August 2011.

[5] G-Lab Project, http://www.german-lab.de.

[6] F. Liers, T. Volkert, A. Mitschele-Thiel: A Flexible Abstraction for the Future Internet, 8th Würzburg Workshop on IP (EuroView), Würzburg, Germany, August 2008.

[7] F. Liers, T. Volkert, A. Mitschele-Thiel: Scalable Network Support for Application Requirements with Forwarding on Gates, EuroView2011, Würzburg, Germany, August 2011.