DGBA: Delay-Guaranteed Bitrate Adaptation for Mobile Broadcaster in Interactive Live Video Streaming

Minglong Dai1,*, Chuming Li2, Xirong Que1 and Yong Cui2

1State key Lab of Networking and Switching, Beijing University of Posts and Telecommunications, Beijing, China
2Department of Computer Science and Technology, Tsinghua University, Beijing, China

*Corresponding author e-mail: daiminglong@mail.dlut.edu.cn

Abstract. Improving the broadcaster-side performance is crucial for ensuring the video quality experience (QoE) for mobile interactive live streaming. But it is challenging in the presence of mobile network’s bandwidth fluctuation, as we need not only to ensure a high video bitrate but also to meet the broadcaster-side constraint. This paper presents Delay-Guaranteed Bitrate Adaptation (DGBA), an algorithm based on the idea of controllable bitrate adaptation used to optimize the broadcaster-side performance. We compare the performance of DGBA with a commercial bitrate adaptation algorithm by implementing these two algorithms in a prototype system. The experiment results show that DGBA can ensure 95.2% broadcaster-side delay constraint satisfaction compared to commercial algorithm’s 84.7%, while being more effective in bitrate adaptation.

1. Introduction

With the rapid development of Mobile Internet and Smartphone, we have seen drastic improvement in terms of high bandwidth and low delay [1]. Mobile interactive live streaming apps have become more and more popular in our daily lives (e.g., Twitch, Douyu, Facebook Live). How to ensure high video quality of experience is a critical problem for interactive live streaming apps. Recently, a lot of works [2, 3, 4] have been done on interactive live streaming to obtain a high QoE, but all these works focus on dealing with dynamic traffic pattern and optimization on video distribution architecture. There is not enough effort to improve the performance of broadcaster-side which need to get a high video bitrate while ensuring broadcaster-side delay satisfies constraint according to the actual industrial demand.

Unlike the traditional live streaming, interactive live streaming requires low delay and high video bitrate since the broadcasters are required to interact with viewers. The measurement results of [5] show that the QoE of interactive live streaming can suffer from the terrible performance of broadcaster-side. The key to high broadcaster-side performance are high video bitrate and low delay. However, it is not easy to get high video bitrate and achieve low delay simultaneously. The delay of the broadcaster-side consists of network transmission delay (from the broadcaster to the resource server) and the queuing delay at the broadcaster buffer. Although the network transmission delay is uncontrollable, a straightforward solution is that we can try to minimize the queuing delay by adjusting the video bitrate equal to the network bandwidth. However, it is hard to do that due to network bandwidth fluctuation. Low video bitrate can easily meet the the broadcaster-side constraint, but it will result in terrible video quality. On the other side, a high bitrate which exceeds the network...
bandwidth can lead to high queueing delay at the buffer of broadcaster. As a summary, a better QoE can be obtained if we can find a way to keep the video bitrate sticking to the network bandwidth tightly. Meanwhile, it is also critical to make the bitrate switch more smooth, as frequent and drastic bitrate switch will result in a bad user viewing experience.

In this paper, we present DGBA, a bitrate adaptation algorithm that aims to improve the broadcaster-side video quality as far as possible while ensuring broadcaster-side delay constraint based on the idea of controllable bitrate adaptation.

The remainder of this paper is organized as follows. Section 2 provides an elaborate analysis of the problem and describes the design of the proposed bitrate adaptation algorithm. Section 3 presents a prototype system and the experimental results. Finally, we conclude the paper in the section 4.

2. Problem Formulation and Algorithm Design

To achieve the optimal balance between video quality and bitrate smoothness under the condition of satisfying constraint, the appropriate action space is essential, and should be carefully selected. An action space regulates the action set where changes to the bitrate can be chosen from based on current network conditions, e.g. an additive increase to the bitrate by a pre-defined stride. We use an action space with two actions, i.e. multiplicative increase and multiplicative decrease (MIMD), which are also taken in [6]. Multiplicative operations enable the bitrate to move on a logarithmic scale, which adapts to devices with a wide range of maximum bitrate.

2.1. Problem Formulation

The problem can be formulated as an optimization problem over an infinitely long-time span. To elaborate this problem, the evolution of the state of broadcaster-side system needs to be described. In this subsection, we consider a toy model of broadcaster-side, which is composed of a sender who generates data at the rate of the current bitrate, a bottleneck link draining the produced data at a speed equaling to the bottleneck link capacity (BLC) and a buffer storing data when bitrate overwhelms BLC.

With the model set above, a set of differential evolution equations can be deduced to render the state of broadcaster-side, before which some notations should be introduced. \( d(t) \) and \( r(t) \) are used to denote the delay and bitrate at time \( t \). \( b(t) \) and \( c(t) \) represent the amount of data in buffer and BLC at time \( t \) respectively.

The definition of \( d(t) \) is the difference of timestamp of data produced at the sender and the data drained by bottleneck link.

With all the notations defined above, the evolution equations of the broadcaster system can be deduced, with \( r(t) \) and \( c(t) \) as the inputs and \( b(t) \) and \( d(t) \) as the outputs.

First, the buffer \( b(t) \) has the derivative as follows:

\[
\dot{b}(t) = \begin{cases} 
 r(t - t_f) - c(t) & \text{if } r(t - t_f) > c(t) \\
 0 & \text{if } r(t - t_f) < c(t) 
\end{cases}
\]  

(1)

Where \( t_f \) is the delay between the sender and the bottleneck link.

Based on the solved buffer \( b(t) \), the evolution of \( d(t) \) is divided into two conditions for discussion. For the first condition, when buffer is vacant:

\[
\dot{d}(t) = \begin{cases} 
 1 - \frac{c(t)}{r(t-t_f)} & \text{if } r(t - t_f) > c(t) \\
 0 & \text{if } r(t - t_f) < c(t) 
\end{cases}
\]  

(2)

It describes a time at which data begins to accumulate in buffer. The form of difference denotes the mismatch of the increase of the timestamp of the data at the sender and at the buffer of bottleneck link.

For another condition, when buffer is with load:

\[
d(0) = t_f \\
\dot{d}(t) = 1 - \frac{c(t)}{r(t-d(t))}
\]  

(3)

It is because that data being drained at the buffer of bottleneck link is \( d(t) \) time before.

In summary, the evolution of the system is described in differential equations as follow:
Specifically, in the model above, if MIMD is used, latency \( d(t) \) will not exceed \((\alpha - 1)T\), where \( \alpha \) is the multiplicative increasement ratio and \( T \) is the time needed to inspect that bitrate exceeds current BLC and to take action to reduce bitrate.

These equations describe the latency \( d(t) \) of the broadcaster-side system, taking \( r(t) \) and \( b(t) \) as inputs. This description helps achieve an optimal balance between bitrate and smoothness under latency restriction. The optimization problem is formed as a sequential decision problem. The time is divided in discrete time step, and an action is taken to change current bitrate at each step on the basis of current network states (i.e. \( r(t), c(t), b(t), d(t) \)). The two optimizations objectives, bitrate and smoothness, are described respectively as:

\[
\lim_{t \to \infty} \frac{\sum_{i=0}^{t} r(i)}{t} \tag{5}
\]

and

\[
\lim_{t \to \infty} \frac{\sum_{i=0}^{t} |r(i+1) - r(i)|}{t} \tag{6}
\]

### 2.2. Algorithm

(1) **Algorithm Design.** In this subsection, an algorithm is designed to achieve our goal. As mentioned above, the problem is a sequential decision problem and the action space is defined as multiplicative increase and multiplicative decrease. The algorithm chooses actions from action space on the basis of current network conditions.

The multiplicative increase and decrease actions are defined as follows:

\[
r(t + 1) = \min \{ \alpha \cdot r(t), r_{\text{max}} \} \quad \alpha > 1 \tag{7}
\]

and

\[
r(t + 1) = \beta \cdot \hat{c}(t) \quad \beta < 1 \tag{8}
\]

where \( \alpha \) and \( \beta \) are parameters to be tuned, and \( \hat{c}(t) \) is the evaluation of current BLC. \( r_{\text{max}} \) is the maximum of bitrate which is decided by the device used by broadcaster. An accurate evaluation of BLC is undeniably crucial for the performance of our algorithm. Taking a conservative view, we use harmonic mean of observed throughputs over several historical time slots, which is:

\[
\hat{c}(t) = \frac{1}{\frac{1}{\sum_{i=1}^{N} thr(t-i)}} \tag{9}
\]

\( thr(t) \) is the measured throughput during the \( t \) th time slot, this quantity is calculated via the received ACKs:

\[
thr(t) = \frac{ack(t) - ack(t-1)}{time(t) - time(t-1)} \tag{10}
\]

where \( ack(t) \) and \( time(t) \) respectively represent data acknowledged by the beginning of the \( t \) th time slot and the timestamp of the \( t \) th time slot.

The key of the proposed algorithm is the decision of whether to increase or reduce bitrate at each time slot. The principle we took is to decrease when:

\[
b(t) > (l_t - t_f) \hat{c}(t) \tag{11}
\]

Otherwise, we should increase current bitrate. In the equation (11), \( l_t \) is the given latency restriction, thereby eq (10) is equivalent to that the data at the tail of current buffer can not be sent in time, under the condition that \( \hat{c}(t) \) is a precise prediction of future BLC. In implementation, we tend to ignore \( t_f \), which is usually tens of milliseconds and very subtle compared to the magnitude order of \( l_t \), which is generally hundreds of milliseconds or even seconds [6].

(2) **Parameter Tuning.** Now that an algorithm is designed, \( \alpha \) and \( \beta \) are parameters needing fine tuning. \( \alpha \) is closely related to the frequency at which \( r(t) \) swings between \( \beta \cdot \text{BLC} \) and \( \text{BLC} \). \( \beta \) regulates the range of \( r(t) / \text{BLC} \). In this paper, we use empirical method to get all parameters that need be tuned. Through a series of offline profiling, different pairs of \( \alpha \) and \( \beta \) are tested. Empirically, \((\alpha, \beta)\) are set \((1.03, 0.85)\), which shows outstanding performance compared to other combinations.
Additionally, the choice of $\beta$ is also under the consideration that sometimes real data rate are fluctuating around the set bitrate. $\beta$’s selection guarantees that $1.1\beta < 1$, where 1.1 is the max ratio between real data rate and the set bitrate.

3. Implementation and Evaluation

3.1. Implementation

Based on the theoretical analysis and algorithm design, we implemented a prototype system to demonstrate the practicability of our idea. The prototype system consists of three parts, i.e. broadcast APP, Openwrt wireless router and resource server, which are shown in figure 1.

(1) Broadcast APP. A broadcast app is implemented in an IOS mobile device based on a our sponsor’s commercial broadcaster architecture. The broadcaster architecture contains a video encoder and a broadcaster streaming protocol stack based on UDP. The process of broadcast can be described as follows. First, the IOS app obtains raw image information captured from the camera. Then the encoder will encode all raw images to a lot of video frames with encoding parameters controlled by bitrate adaptation algorithm. Finally all the video frames will be transmitted to the resource server for future processing (e.g., transcoding, transport to viewers through CDN) through the broadcaster streaming protocol stack. Our DGBA algorithm is implemented in this app as a callback function. This function will be invoked with a certain period interval to calculate a target adapted bitrate for the encoder. All the input signals that DGBA needed are provided by the commercial broadcaster architecture.

(2) Openwrt Wireless Router. In order to create a controllable bandwidth fluctuation, we connected the broadcaster side to the Internet through the Openwrt wireless router. In this way, we can control the access bandwidth of the broadcaster-side using bandwidth control script based on Linux tc (traffic control) tool.

(3) Resource Server. The resource server is a cloud server used to receive video frames sent from the broadcaster-side. The resource server will serve for the future processing procedures, which is out of the scope of this paper.

3.2. Evaluation

In this subsection, we conducted experiments to evaluate the performance of our bitrate adaptation algorithm compared with a commercial method. First, based on the industrial requirement, we set the broadcaster-side constraint to 500ms in the broadcast app. Then we ran the bandwidth control script in the Openwrt wireless router to produce access bandwidth fluctuations with regular intervals according to table 1. Finally, we ran the same broadcast session with DGBA algorithm and the commercial one to evaluate the performance.

![Figure 1. Prototype system architecture.](image)

| Time interval | 0s–45s | 45s–90s | 90s–135s | 135s–180s | 180s–215s |
|---------------|--------|---------|----------|-----------|-----------|
| Bandwidth(Mbit/s) | 1      | 2       | 3        | 2         | 1         |
Figure 2. Performance of delay-guarantee.

Figure 3. DGBA’s bitrate adaptation performance.

Figure 4. Commercial algorithm’s bitrate adaptation performance.

Performance of delay-guarantee. In the following, we examined how DGBA helps to meet the broadcaster-side constraint. We recorded all broadcaster-side delays in every callback cycle, and the CDF of recorded delays of two algorithms are shown in figure 2. As can be seen, DGBA can achieve 95.2% broadcaster-side delay constraint satisfaction, while the commercial algorithm is only 84.7%. In summary, DGBA’s delay-guarantee mechanism works better and can outperform the commercial one by 11%.

Performance of bitrate adaptation. Then, we evaluated how effective the DGBA adapt the bitrate to the fluctuating bandwidth. As shown in Figure 3 and Figure 4, our algorithm DGBA outperforms the commercial algorithm in terms of the bitrate adaptation. The bitrates allocated by DGBA switches smoother and more closed to the actual fluctuating network bandwidth. By contrast, the commercial algorithm’s bitrate switches are always drastic. And we also recorded the average bitrate switch amplitude, (ABSA) which is the value of eq (6). The smaller value of ABSA means the bitrate switch
more smooth. The results show that our algorithm DGBA’s ABSA is only 5.47, but the commercial algorithm’s ABSA is 37.91.

4. Conclusion
In this paper, we have elaborately analyzed how to improve the QoE of interactive live streaming applications through the way of bitrate adaptation and proposed DGBA algorithm.

Then, we implemented a prototype system and conducted experiments to evaluate the performance of DGBA and a commercial bitrate adaptation algorithm. The experiment results show that DGBA outperforms the commercial one in terms of both the meet ratio of constraint and the average bitrate. Meanwhile, DGBA can do a better job in bitrate adaptation to improve the video quality.

Acknowledgments
The author would like to thank Mowei Wang’s advices to this paper and Powerinfo company’s technical support to this paper.

References
[1] Huang J, Qian F, Guo Y, et al. An in-depth study of LTE: effect of network protocol and application behavior on performance[J]. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 363-374.
[2] Zhang C, Liu J. On crowdsourced interactive live streaming: a Twitch.tv-based measurement study[C]//Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video. ACM, 2015: 55-60.
[3] Siekkinen M, Masala E, Kämäräinen T. A first look at quality of mobile live streaming experience: the case of periscope[C]//Proceedings of the 2016 Internet Measurement Conference. ACM, 2016: 477-483.
[4] Wang B, Zhang X, Wang G, et al. Anatomy of a personalized livestreaming system[C]//Proceedings of the 2016 Internet Measurement Conference. ACM, 2016: 485-498.
[5] Ren Q, Cui Y, Wu W, et al. Improving Quality of Experience for Mobile Broadcasters in Personalized Live Video Streaming[J].
[6] Carlucci G, De Cicco L, Holmer S, et al. Analysis and design of the google congestion control for web real-time communication (WebRTC)[C]//Proceedings of the 7th International Conference on Multimedia Systems. ACM, 2016: 13.