An Improved Algorithm for Adaptive Streaming

Siming Zhang[1], Xueshi Ge[2] and Geng Wang[1,*]
[1]School of Software, Shanghai Jiao Tong University, Shanghai, China
[2]Naval Academy of Armament, Shanghai, China
[*]Corresponding author

Abstract—**With the development of network facilities and the wide spread of smart phones, traditional single bitrate video streaming strategy is no longer able to satisfy people's growing needs for high quality video streams. Most of today's adaptive streaming strategies are based on bandwidth estimation such as Smooth Streaming by Microsoft IIS and HDS by Adobe. Bandwidth estimation is less accurate in network jitters which may lead to too many unnecessary stream-switches. Some other strategies are based on client-side metrics monitoring such as dropped frames, average throughput, CPU usage, buffer level. In this paper, we present a strategy that combines bandwidth estimation and client-side metrics monitoring. The client-side controller monitors the variation of the buffer level and triggers the stream-switching event accordingly. When the level of the buffer reaches to a relatively stable state, the bandwidth-estimation event is triggered to estimate the available bandwidth. If higher bitrate stream is available with the currently estimated bandwidth provided, then the stream-switching event is triggered to switch to a higher bitrate stream.**

Keywords-adaptive streaming; bandwidth estimation; buffer level variation

I. INTRODUCTION

With the continuous improvement of Internet facilities and the growing popularity of video applications, the need for high quality video streams are increasing dramatically. But the need for high bitrate video streams usually conflicts with the need for seamless experience due to the network jitter like bandwidth fluctuation and packet loss. Adaptive video streaming is a technique that dynamically switches the stream that best fits the network bandwidth and at the same time providing the most seamless playback experience.

To stream a video adaptively, the server transcodes the video stream into multi-bitrate(resolution) streams, and transports the stream according to clients' available bandwidth. When bandwidth drops or increases, the stream is switched to lower or higher level to make the best use of network resource. The logic of stream-switching is usually implemented on client side to meet their own demands so that the client side is more flexible and the server side is decoupled from the detail of the client side logic. Basically stream-switching strategy is based on bandwidth estimation or client-side buffer level monitoring.

A. Bandwidth-Estimation Based Strategy

A client may estimate bandwidth based on its historical throughput before it requests a new video segment from server. Let $B_{estimated}$ be the estimated bandwidth, B_{cur} be the bitrate of current stream, $B_{cur\pm1}$ be the bitrate of one level up/down of current stream, and B_{next} be the bitrate of next segment to request. The logic may seem like the following:

$$B_{next} = \begin{cases} B_{cur+1}, & B_{est} > \lambda \times B_{cur+1} \\ B_{cur}, & B_{cur} < B_{est} < \lambda \times B_{cur+1} \\ B_{cur-1}, & B_{est} < B_{cur} \end{cases}$$

where λ is a user-defined factor to minimize the effect of inaccurate bandwidth estimation.

B. Buffer-Level Based Strategy

This strategy decides when stream-switch will be triggered by setting buffer thresholds T_{high} and T_{low} and monitoring the buffer level T_{cur} on client side.

$$B_{next} = \begin{cases} B_{cur+1}, & T_{cur} > T_{high} \\ B_{cur}, & T_{low} < T_{cur} < T_{high} \\ B_{cur-1}, & T_{cur} < T_{low} \end{cases}$$

But the effectiveness of buffer-level based strategy is strongly influenced by values of T_{low} and T_{high}. If T_{low} is set too low, it may not be able to avoid buffer outage, but if it is set too high, it may cause unnecessary switches. The same goes for T_{high}.

C. Buffer-Variation Based Strategy

This is a branch of buffer-level based strategy, only it uses the buffer-level variation instead of buffer-level thresholds as the basis for stream-switching. When it detects that the buffer-level drops continuously, it switches to a lower level to avoid buffer outage. The problem with this strategy is its inability of switching to a higher level stream when the bandwidth is improved. The client side player merely requests segment aggressively, which means it would try to maintain a stable buffer level. When the buffer level seems to outbound the target buffer level, it may slow down its rate of requesting new segments.

The strategy proposed in this paper combines the advantage of buffer-variation and bandwidth-estimation based strategies. When the network is unstable, it uses the buffer variation as its basis to switch the stream to avoid buffer outage. When the network is rather stable, it estimates the bandwidth to decide whether it's available for a higher level stream.

The paper is organized as follows. Section II summarizes the related work concerning video streaming techniques. In Section III, an improved adaptive algorithm is proposed. And Section IV describes the tests and results. Section V draws the conclusion.

II. RELATED WORK

A great work of research has been made to improve the multimedia Quality of Experience during the last few decades, which we summarize as follows:

A. *Video Encoding*

H.264/SVC is an extension to the H.264/MPEG-4 Advanced Video Coding standard for video compression. It standardizes the encoding of a high-quality video bitrate stream that contains a base layer and one or more enhance layer(s). An enhance layer is derived from original stream by dropping packets or frames to reduce the bandwidth required for the subset stream. The enhance layer can represent a lower spatial resolution (smaller screen), lower temporal resolution (lower frame rate), or lower video quality. This allows representing different bitrate streams in different frame rate, resolution and quality with only one single video stream. A study by Travis et at [1] shows how we can improve video Quality of Service (QoS) with SVC streams.

B. *Stream Delivering*

To deliver video streams, techniques like error concealment and error recovery can be applied. And protocols like Real-time Transport Protocol (RTP) [3] and Real-time Message Protocol (RTMP) are proposed. RTP is used together with RTP Control Protocol (RTCP). While RTP is used to transport the video data, RTCP is used to transport statistics like packet loss rate or time delay that can be used to support stream-switching decisions. But RTP/RTCP based applications are more difficult to develop and deploy and usually need firewall traversing.

Dynamic Adaptive Streaming over HTTP (DASH) [4] is provided as another way. In DASH, a video stream is transcoded into multiple different bitrate streams. Video players on the client side can switch streams based on network environment. Compared with traditional RTP/RTCP based applications, DASH provides inherent firewall traversing, and is easier to integrate into existing web services. These years, different implementations based on DASH have been developed such as HLS by Apple and Smooth Streaming by Microsoft.

C. *Client-Side Techniques*

The most commonly used techniques on client side are data buffering and bandwidth estimation.

With data buffering, the video player can combat network jitter and avoid playback interruptions to some extent. By monitoring the client side buffer level, the video player can adjust the playback interval to meet the data receiving rate in order to improve clients' quality of experience [5]. But it could not switch to higher bitrate streams when there is enough bandwidth provided, especially in live streaming.

To switch to a higher stream, the client side video player needs to estimate the available bandwidth. When the estimated bandwidth is not enough for current video stream, it switches to a stream with lower bitrate, and when the estimated bandwidth is enough for next level stream with higher bitrate, it switches to next level stream. To estimate the available bandwidth, the client may trigger a traffic probing event every few seconds and takes the throughput as an approximation of available bandwidth [2]. The problem is that it wastes too much network resources and may cause unnecessary network congestion. In Dash, streams are spliced into segments and then sent to client. Client can take the throughput of every segment as an approximation of the available bandwidth, which is more resource-friendly. But bandwidth estimation based techniques may cause too many unnecessary switches due to the inaccuracy of estimation when the network is unstable.

In this paper we dedicated to a new strategy that combines the advantages of data buffering and bandwidth estimation together.

III. AN IMPROVED ALGORITHM

Assume the video server has the original stream transcoded into *n* streams with different bitrates:

$$L = \left\{ l_i \mid i = 1, 2, \ldots, n \right\} \qquad (1)$$

The information of available video stream l_i and its bitrate is listed in a manifest file. When the client video player starts, it first requests the manifest file from the server and parse it to get the video stream list and start to request video stream from the lowest level. Client can only choose from L in later requests for video streams.

In our strategy, there are three states: Adjust State, Network Stable State, and Stream Switch State.

A. *Adjust State*

In *Adjust* state, network environment is considered unstable, buffer-level on client's side drops or increases dramatically. If current bitrate of requested stream is beyond the available bandwidth, the buffer-level drops continually and finally leads to buffer outage. If the buffer level drops continuously, the client switches to a lower level video stream that is available in stream list *L* to avoid buffer outage. When the bandwidth fits the video stream bitrate, the buffer-level would eventually get stable, and client switches to *Stable* state.

However, when the buffer level increases continuously, it indicates that the client has some trouble decoding current video stream which would result in slower playback rate. In this case, the client also needs to switch to a lower level to avoid buffer overflow and so the bitrate of the stream fits the client's ability of decoding.

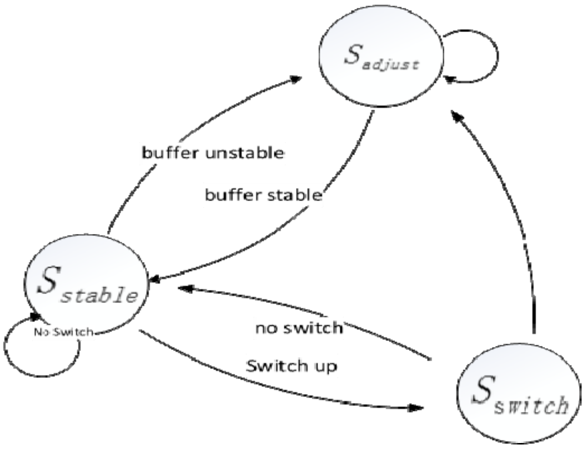

FIGURE I. STATE TRANSFER

B. *Stable State*

In *Stable* state, the bandwidth is considered sufficient and the buffer level is considered stable. However the problem is, the buffer level variation in stable state would not reflect bandwidth improvement. It is due to the shortcomings of buffer-level variation strategy. As previous bandwidth is sufficient, packet loss and retransmission has little influence on buffer-level, the buffer-level would remain stable even bandwidth is improved, which means stream-switching event would not be triggered. So bandwidth and the opportunity for better video quality is wasted when there is a higher video stream available.

To overcome this, the client estimates the available bandwidth. If the available bandwidth is larger than the bitrate of current stream, the client switches to *Switch* state to switch to a higher level, otherwise, it stays in Stable state and estimate the bandwidth later. Because bandwidth estimation is triggered in *Stable* state, the network condition is considered stable. So without the effect of network fluctuation, the estimated bandwidth is more accurate and thus would not cause many unnecessary stream switches. When buffer level fluctuates, the client switches back to *Adjust* state.

C. *Switch State*

In *Switch* state, the client checks if the available bandwidth estimated in *Stable* state is sufficient for a higher level stream. By comparing the estimated bandwidth with the bitrate of streams in *L*, the client decides whether to switch to a higher level. If a higher stream is available, the client switches to it, otherwise, it switches back to *Stable* state.

When client starts, it start to request stream from the lowest stream level, and switches to *Adjust* state. If the bandwidth is sufficient for the lowest level stream, client would reach *Stable* state a few cycles later. In *Stable* state, client estimates bandwidth in T seconds. If estimated bandwidth is sufficient for higher level stream, client switches to *Switch* state and starts to request higher stream from next request on.

The flow of the algorithm is described in Figure 2.

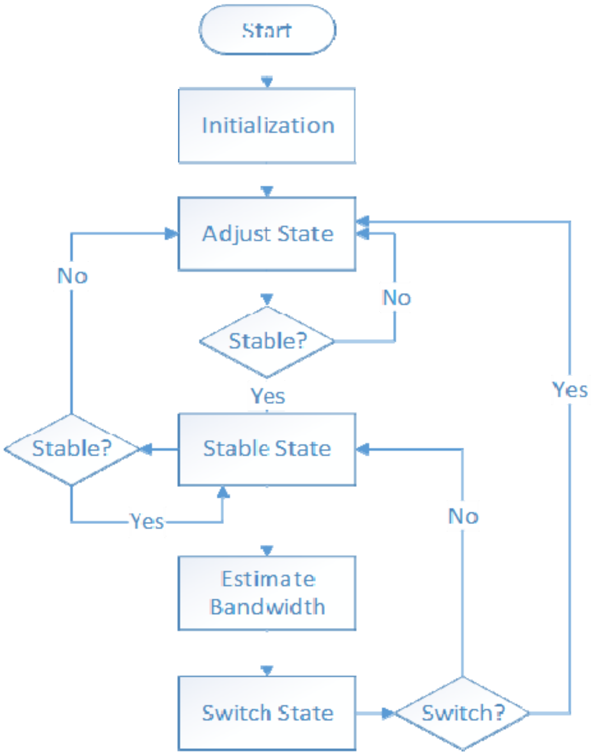

FIGURE II. IMPROVED ALGORITHM

IV. EXPERIMENTS

We use iproute2 to control the network. Experiments are conducted under the following indices:

1. estimated bandwidth and real bandwidth.
2. buffer level variation.
3. stream-switch.

In this paper we use dash.js as our client video player. Original stream is transcoded into multiple streams with different bitrate on the server side. We use the average throughput of recently received segments as the estimated bandwidth.

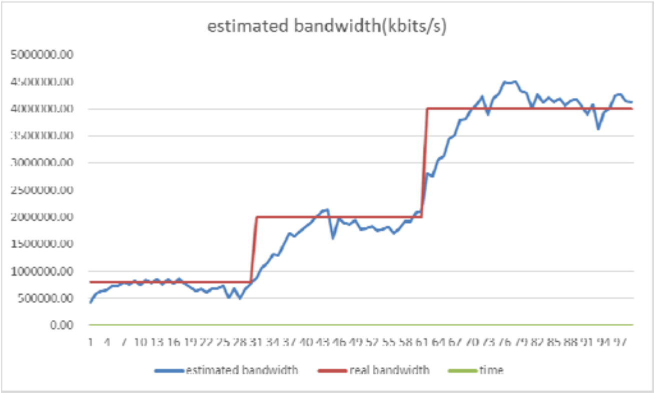

FIGURE III. ESTIMATED BANDWIDTH

As Figure 3 shows, the estimated bandwidth would slowly converge to the real bandwidth. But bandwidth is not the only factor that can influence the stream-switching decision, other metrics like CPU load and memory usage should also be considered.

Buffer level could effectively reflect bandwidth drop and other metrics such as CPU load and memory usage, as shown in Figure 4 and Figure 5. If bandwidth is sufficient, buffer level would remain stable. If buffer level drops continuously, it can be inferred that the network bandwidth is not sufficient. However, if the buffer level grows continuously beyond the target buffer level, the client should also switch down to a lower level stream because of the client's inability of decoding the stream in normal rate.

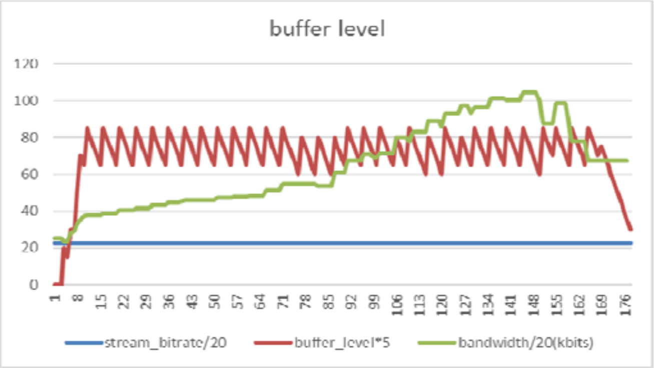

FIGURE IV. BUFFER LEVEL IN STABLE STATE

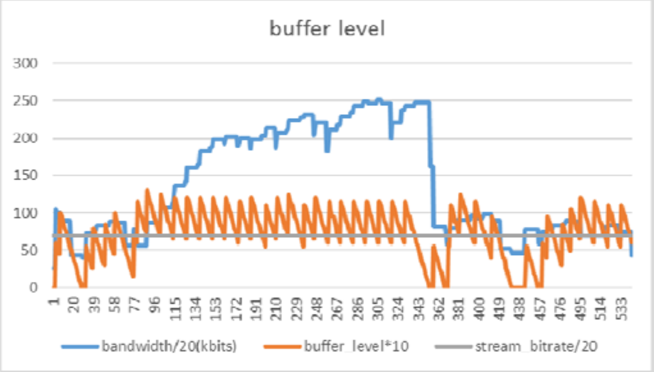

FIGURE V. BUFFER LEVEL VARIATION

By monitoring buffer level variation in *Adjust* state, the client can decide when to switch down to a lower level. When the client transfers to *Stable* state, it uses the estimated bandwidth to decide if it can switch to a higher level, as shown in Figure 6.

V. CONCLUSION

Traditional data buffering technique for adaptive video streaming is easy to understand and implement, but it's not able to respond to better bandwidth. While bandwidth estimation technique is able to respond to better bandwidth, its accuracy is often affected by network fluctuation.

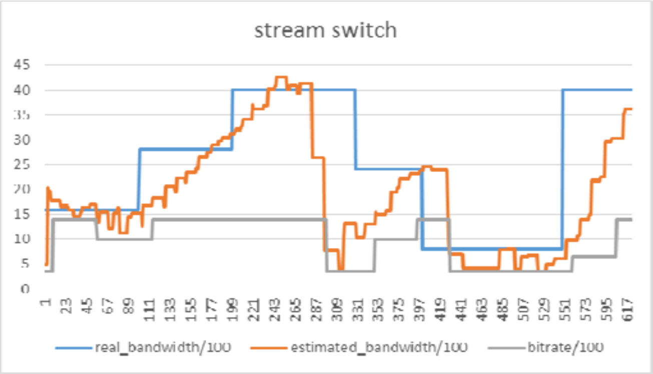

FIGURE VI. STREAM SWITCHING

Figure 7 shows the result of our new strategy combining buffer level variation and bandwidth estimation together. From these Figures we can see that our new strategy estimates the bandwidth less frequent and is better in avoiding buffer outages.

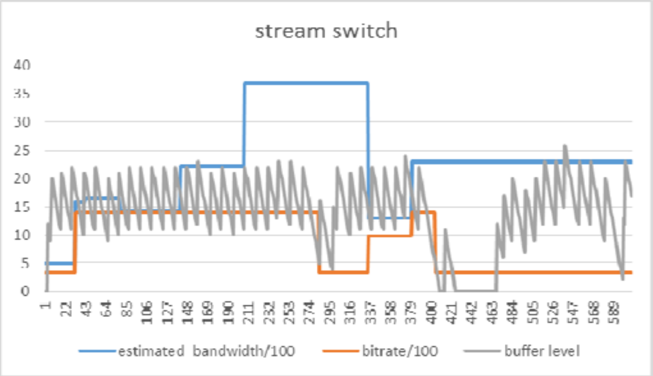

FIGURE VII. STREAM SWITCHING

In this paper we propose a new strategy for adaptive video streaming to combine data buffering technique and bandwidth estimation technique together. The new strategy in this work aims at providing clients with the best quality of video stream and seamless quality of experience. Experiments in IV show our new strategy effectively.

REFERENCES

[1] Travis Andelin, Vasu Chetty, Devon Harbaugh, Sean Warnick, and Daniel Zappala. Quality selection for dynamic adaptive streaming over http with scalable video coding. In Proceedings of the 3rd Multimedia Systems Conference, pages 149–154. ACM, 2012.

[2] Luca De Cicco and Saverio Mascolo. An adaptive video streaming control system: Modeling, validation, and performance evaluation. Networking, IEEE/ACM Transactions on, 22(2):526–539, 2014.

[3] Henning Schulzrinne, Stephen Casner, Ron Frederick, and Van Jacobson. Rtp: A transport protocol for real-time applications. Technical report, 2003.

[4] Thomas Stockhammer. Dynamic adaptive streaming over http–: standards and design principles. In Proceedings of the second annual ACM conference on Multimedia systems, pages 133–144. ACM, 2011.

[5] Ya-Fan Su, Yi-Hsuan Yang, Meng-Ting Lu, and Homer H Chen. Smooth control of adaptive media playout for video streaming. Multimedia, IEEE Transactions on, 11(7):1331–1339, 2009.