Enabling High Frame-rate UHD Real-time Communication with Frame-Skipping

Tingfeng Wang[1], Zili Meng[2], Mingwei Xu[2], Rui Han[3], Honghao Liu[3]
[1]Beijing University of Posts and Telecommunications, [2]Tsinghua University, [3]Tencent
wangtingfeng@bupt.edu.cn, zilim@ieee.org, xumw@tsinghua.edu.cn {raymondhan, coreyliu}@tencent.com

Abstract

With a high frame-rate and high bit-rate, ultra-high definition (UHD) real-time communication (RTC) users could sometimes suffer from severe service degradation. Due to the fluctuations of frames incoming and decoding at the client side, a decoder queue could be formulated before the streaming decoder at the client side. Those fluctuations could easily overload the decoder queue and introduce a noticeable delay for those queued frames. In this paper, we propose a Frame-Skipping mechanism to effectively reduce the queuing delay by actively managing the frames inside the decoder queue. We jointly optimize the frames with skipping to maintain the end-to-end delay while ensuring the decoding quality of video codec. We also mathematically quantify the potential performance with a Markovian chain. We evaluate the Frame-Skipping mechanism with our trace-driven simulation with real word UHD RTC traces. Our experiments demonstrate that Frame-Skipping can reduce the ratio of severe decoder queue delay by up to 23× and the ratio of severe total delay by up to 2.6×.

CCS Concepts

• **Information systems** → **Multimedia streaming**; • **Mathematics of computing** → *Queueing theory*;

Keywords

real-time communication; queue management; Frame-Skipping

ACM Reference format:
Tingfeng Wang, Zili Meng, Mingwei Xu, Rui Han, Honghao Liu. 2022. Enabling High Frame-rate UHD Real-time Communication with Frame-Skipping. In *Proceedings of 3rd ACM Workshop on Hot Topics in Video Analytics and Intelligent Edges, New Orleans, LA, USA, January 31–February 4, 2022 (HotEdgeVideo '21),* 6 pages.
https://doi.org/10.1145/3477083.3481582

1 Introduction

With the emergent demand for high quality and interactive video streaming under COVID-19 [5, 17], the network volume shared by ultra-high-definition (UHD) real-time communication (RTC) streaming is remarkable and still increasing rapidly [30]. These UHD RTC services like UHD Remote Conferencing, virtual reality, and cloud gaming are now attracting attention in both academia and industry side [23, 35]. To satisfy users, UHD RTC services aim to deliver streaming with a high bit-rate (up to 30Mbps) and high frame-rate (up to 60 frames per second (fps)) [4, 16] along with the ultra-low latency for interactivity.

The existing streaming delivery pipeline could satisfy the traditional streaming service like 4k video streaming [20]. However, under the high frame-rate UHD RTC service, This delivery pipeline could be sub-optimal. For example, the streaming decoder at the client side might not be capable of decoding video frames timely because of the network jitter and decoding time variation. In this case, when new frames arrive before the decoding of previous frames, new frames have to be queued before the decoder. The formulation of such a decoder queue could introduce a high queuing delay for queued frames. The high frame-rate, along with the network jitter, will cause the bursty incoming of video frames, which can instantly overload the queue (§2.2.1). And the high bit-rate means more complexity of video frames, which could increase the decoding time [19]. These reasons will result in a queuing delay at the decoder queue of tens of milliseconds (ms) or even longer (§2.1). With the increasing needs of real-time communications from the applications (e.g., less than 20 milliseconds in virtual reality applications [7, 26]), such an enormous decoder queue delay becomes noticeable and needs to be eliminated.

However, managing the decoder queue to achieve both low latency and high image quality is non-trivial due to the following reasons. Straightforward solutions include controlling the arrival rate (i.e., frame-rate) or the service rate (i.e., decoding time) of the decoder queue. However, since the encoder and decoder are located in distance, it is challenging to dynamically change the encoding parameters due to the control loop (including in-between network delay, effective delay at the encoder). Therefore, it is not timely enough to dynamically adjusting the frame-rate (to reduce the arrival rate of frames) or the bit-rate (to accelerate the decoding time of frames).

In response, motivated by research efforts in the active queue management (AQM), instead of controlling the arrival rate or service rate, we directly control the frames inside the decoder queue: when the decoder queue has been built up, directly dropping frames in the queue could effectively reduce the queuing delay for remaining frames. However, simply dropping frames in the queue would make the subsequent frames undecodable and further degrade the image quality due to the inter-dependency [32]. Therefore, we need to jointly optimize the frames to maintain the low decoder queue delay, and ensure the decoding quality by reordering the encoder buffer.

In this paper, we propose the Frame-Skipping mechanism, which reorders the encoder buffer and carefully skip frames in the decoder queue without damaging decoding quality. Moreover, Frame-Skipping controls the delayed improvement and frame loss performance by controlling the *skip rate*, which indicates how many frames will be skipped. To helping to deploy this brand-new decoder queue management to UHD RTC services, a mathematical

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. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

HotEdgeVideo '21, January 31–February 4, 2022, New Orleans, LA, USA

© 2022 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.

ACM ISBN 978-1-4503-8700-2/22/01…$15.00
https://doi.org/10.1145/3477083.3481582

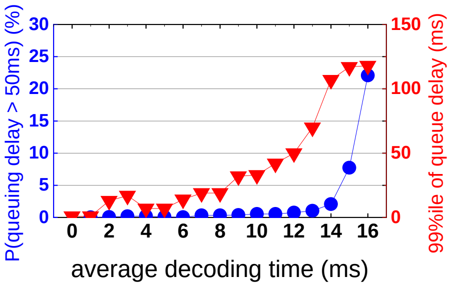
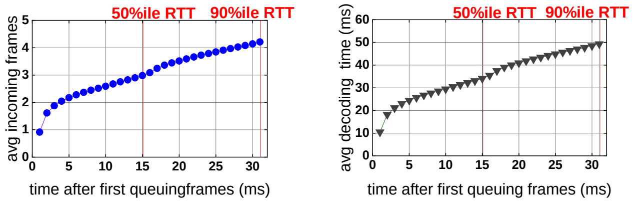


Figure 1: The ratio of decoder queue delay > 50ms (blue line) and the 99th percentile decoder queue delay (red line) of frames grouped by average decoding time.



(a) bursty incoming frames.

(b) decoding time of those bursty incoming frames.

Figure 2: Incoming frames and corresponding decoding time after the first queuing frame with severe queuing delay (decoder queue delay > 50ms) later.

model will be introduced to quantify the performance under different skip rates. And a numerical illustration based on the model will be presented to recommend skip rates.

We evaluate the Frame-Skipping mechanism on our trace-driven simulator with real word UHD RTC traces of 6 Billion video frames from a popular cloud gaming service. Our simulation demonstrates that Frame-Skipping can reduce the ratio of frames with severe decoder queue delay by up to 23× and the ratio of frames with severe total delay by up to 2.6× (§4.1). Moreover, Frame-Skipping only introduces a slight loss of image quality and will not gives extra stress to the decoder and the network transportation (§4.2).

2 Background and Motivation

In this section, we illustrate the decoder queue overload scenario, and introduce the existing solutions for eliminating queue overload to motivate our design of Frame-Skipping.

2.1 Decoder queue overload

Due to the high frame-rate and high bit-rate in UHD RTC service, there are more incoming video frames and more video data that need to be transferring and decoding, Result in higher utilization of the decoder queue. With more fluctuation due to the streaming over the internet, an enormous decoder queue delay could occur and degrade the user's experience.

We investigate the severity of decoder queue delay based on massive online traces of 6 Billion frames. We first notice that the decoder queue delay problem will be more severe with a long average decoding time. According to Figure 1, we notice that for those frames streaming to the client with average decoding time ≥ 12ms, the ratio of frames with decoder queue delay > 50ms will exceed 1%. I.e., it could happen every two seconds on average under 60 fps streaming. This severe decoder queue delay (>50ms) is conspicuous compared to a dozen milliseconds delay of RTT and decoding time [15], and could easily cause a sensible total delay to degrade the user's experience. Therefore, eliminating the severe decoder queue delay becomes necessary.

Referring to queue theory, a queue overload only happened with a higher arrival rate or lower service rate or both [6]. The arrival rate, i.e., the speed of frames incoming from network, is affected by the network environment. The service rate will be represented as the average decoding time for a period under UHD RTC service. Having a long average decoding time will more likely to have a long decoding time for a period (low service rate), so it will be more likely to encounter queue overload. Therefore it will be more valuable for those clients with a long average decoding time.

2.2 Possible solutions for queue overload

To maintain a low queuing delay, we will discuss some possible solutions.

2.2.1 *Arrival rate management.*

One solution to maintain a low queuing delay is to avoid queue accumulation by altering the arrival rate on the arrival side of the queue.

Because the variation of the arrival and service rate will also cause the queue overhead [6], pre-negotiate a lower arrival rate can not guarantee a persistent low queuing delay under the network jitter and decoding time variation [28].

When we are trying to alter the arrival rate in real-time, under UHD RTC service, the encoder, which decides the arrival rate, and the client are located in distance. Therefore the arrival rate adjustment will have a long control loop due to the network transportation. And this control loop can be considered as the round-trip time (RTT) of the service. Depending on the types of UHD RTC service, the control loop considered as RTT could range from 10ms (edge-accelerated streaming service like cloud gaming) [4] to 50ms+ (UHD Remote Conferencing).

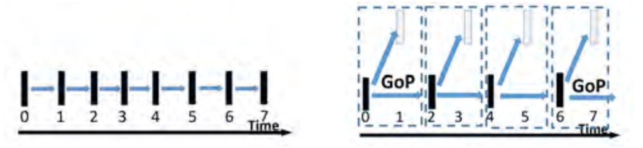
Our real word traces demonstrate that the decoder queue could still be overloaded even with an instant reaction of arrival rate adjustment. In figure 2(a), we assume that the arrival rate management can react instantly, i.e., we can predict there will be a severe queuing delay (decoder queue delay > 50ms) later for the first frame start queuing. And we will send an arrival rate altering command which will activate after a long control loop of RTT. With the 50th percentile of traces RTT of 15ms (middle line in Figure 2(a)), there are already *three* frames that have enqueued on average. It illustrates that the decoder queue will overload with a bursty video frame incoming. And the arrival rate management is limited to solve it due to its inherent long control loop.

Moreover if we are trying to use some predictive methods to foresee the bursty incoming to react before it happens, There still will be some limitations. Firstly, the output action for arrival rate adjustment is based on real-time variables measurement [9]. The delay of measurement to generating output action, along with a dozen milliseconds of long control loop could still cause some frames enqueued before arrival rate adjustment activate. Secondly, trying to predict the bursty incoming before it happens will face the error predict problem [1], and minimize predict error is still challenging [10, 13].

Besides, the time of waiting for those bursty incoming frames dequeuing is noticeable. According to Figure 2(b), these *three* bursty incoming frames under 15ms RTT need to spend *34*ms to dequeue (decode) all of them on average. So subsequent incoming frames could suffer from dozens of milliseconds of queuing delay. And this problem will even be worse under the 90th percentile RTT of 32ms (right line in Figure 2(b)).

2.2.2 *Speeding up decoding time.*

Another solution to maintain a low queuing delay is to speed up the service rate (decoding time).



(a) original reference frames sequence.

(b) reference frames sequence after reordering.

Figure 3: Reference frames reordering. Blue arrows indicate the reference frames used to predict the frames being coded.

But the decoding time represents the frame complexity [19], simply boosting decoding time by reducing the frame complexity will degrade the image quality. And trying to boost the executing time inside the codec model is challenging for an application that unauthorized to modify system procedures like CPU scheduling.

In order to eliminate the decode queue delay instantly and efficiently, we propose the Frame-Skipping mechanism to achieve all these objects.

3 Design

In this section, we first introduce the Frame-Skipping mechanism in §3.1, and then theoretically analyze the mechanism with a Markovian model in §3.2.

3.1 Frame-Skipping Mechanism

To introduce the Frame-Skipping mechanism, we first briefly introduce the enabler of dropping frames at the decoder queue, and answer two key questions in our design, i.e., which frames to skip and how many frames to skip.

Background: Reordering decoder buffer. The inter-frame prediction of video codec standard makes video frames encoding and decoding based on the prediction of the previous frame. Now with the recent development of scalable Video Coding (SVC), some frames within the streaming are now discardable [31]. Therefore, we can leverage the SVC extension or NVIDIA video codec *InvalidateRefFrames* API [8] to reordering the encoder reference frames sequence from Figure 3(a) to Figure 3(b).

After reordering the encoder reference frames sequence, some frames won't be used to predict the subsequent frames to ensure the decoding quality after discarding them. So when we extract the Group of Picture (GoP) capacity amount of neighbouring frames in the decoder queue, there must be a base layer frame (black frames 0, 2, Etc. in Figure 3(b)). Because the base layer frame is the only frame used to predict the following GoP frames, only decoding the base layer frame and dropping others within GoP will not damage the decoding quality.

Which frames to skip? After having the ability to dropping frames in decoder queue , we propose our active queue management (AQM) called Frame-Skipping to efficiently eliminate the decoder queue delay.

In Frame-Skipping, we will choose to drop earlier frames at dequeuing rather than dropping the newest arrival frames at enqueuing like those widely used tail-dropping AQM (RED [25], PIE [27], Etc.). That is because our goal is maintaining low queuing latency instead of mere low queue length, dropping earlier and displaying recent frames can deliver a lower latency streaming.

More specifically, when the decoder queue is overloaded, we will extract the GoP amount of neighbouring frames in the head of the decoder queue, then decode only one and skip (drop) others for those neighbouring frames. Intuitively, extracting more frames at the same time means eliminating decoder queue delay more efficiently, but it also means more frames will be dropped (more waste of network bandwidth). There will be a trade-off between them.

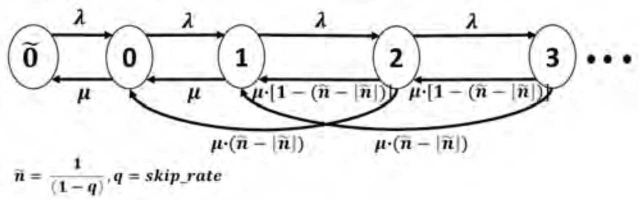

λ
λ
λ
λ
$\tilde{0}$
0
1
2
3
• • •
μ
μ
$\mu\cdot[1-(\tilde{n}-\lfloor\tilde{n}\rfloor)]$
$\mu\cdot[1-(\tilde{n}-\lfloor\tilde{n}\rfloor)]$
$\mu\cdot(\tilde{n}-\lfloor\tilde{n}\rfloor)$
$\mu\cdot(\tilde{n}-\lfloor\tilde{n}\rfloor)$
$\tilde{n}=\frac{1}{(1-q)}, q = skip\_rate$

Figure 4: The transition diagram under skip rate $q \in [0, 0.5]$ with the M/M/1 queuing system

How many frames to skip? To balance this trade-off, in Frame-Skipping, we offer a parameter called skip rate q, which represents the expectation of dropping frames proportion. So the amount of extracting frames will be $\tilde{n} = 1/(1-q)$.

We set the skip rate q in Frame-Skipping to $q \in (0, 0.5]$. That's because according to our evaluation, under skip rate $q = 0.5$ (extracting $\tilde{n} = 2$ frames at the same time), the frames loss frame rate can exceeds 2% or even 10% in some scenario (§4.1.2). Wasting this large amount of network bandwidth in UHD streaming for exchanging delayed improvement will be unacceptable.

Moreover, if the administrator not willing to suffer high frame loss rate, it's still feasible to set a skip rate $q < 0.5$. When skip rate $q \in (0, 0.5)$ with extracting frames amount $\tilde{n} = 1/(1-q) \in (1, 2)$, We will set GoP capacity = 2 and take turns to extracting $\lceil\tilde{n}\rceil$ frames with probability of $(\tilde{n} - \lfloor\tilde{n}\rfloor)$ and extracting $\lfloor\tilde{n}\rfloor$ frames with probability of $[1 - (\tilde{n} - \lfloor\tilde{n}\rfloor)]$ to achieve the expectation extracting frames amount $\tilde{n}$. Because

$$\lfloor\tilde{n}\rfloor\cdot[1-(\tilde{n}-\lfloor\tilde{n}\rfloor)]+\lceil\tilde{n}\rceil\cdot(\tilde{n}-\lfloor\tilde{n}\rfloor) = \lfloor\tilde{n}\rfloor+(\lceil\tilde{n}\rceil-\lfloor\tilde{n}\rfloor)\cdot(\tilde{n}-\lfloor\tilde{n}\rfloor) = \tilde{n} \quad (1)$$

In conclusion, we propose an AQM called Frame-Skipping to eliminate the decoder queue delay efficiently. We chose to drop frames at dequeuing to deliver lower latency streaming, and offer a parameter skip rate q to balance the trade-off between delayed improvement and frame loss.

To quantify the delayed improvement and frame loss performance with different skip rates q in Frame-Skipping, we introduce a mathematical model with numerical illustration in the following.

3.2 Theoretical Analysis

To abstract the decoder queue model, we will assume that the decoder queue will be an M/M/1 Queuing System as an approximation which is generally used [22, 33]. I.e., the arrival and departure process will be a Poisson process with the rate λ and μ, and there is only a single server [6]. So we have the state space of $\tilde{0}$ (no frame in decoding) and n ($n \geq 0$, n frames waiting in decoder queue) with probability of P_n.

With skip rate $q \in (0, 0.5]$ along with GoP capacity = 2, $\tilde{n} = 1/(1-q) \in (1, 2]$. the transition diagram will be shown in Figure 4. According to the transition diagram, we can calculate the balance equations [29] will be:

$$\begin{array}{cc}
\text{State} & \text{Rate of process leaves = Rate of it enters} \\
\tilde{0} & \lambda P_{\tilde{0}} = \mu P_0 \\
0 & (\lambda+\mu)P_0 = \lambda P_{\tilde{0}} + \mu P_1 + \mu(\tilde{n}-\lfloor\tilde{n}\rfloor)P_2 \\
n, n \geq 1 & (\lambda+\mu)P_n = \lambda P_{n-1} + \mu[1-(\tilde{n}-\lfloor\tilde{n}\rfloor)]P_{n+1} \\
 & +\mu(\tilde{n}-\lfloor\tilde{n}\rfloor)P_{n+2}
\end{array}$$

Now the set of recurrence equations

$$(\lambda+\mu)P_n = \lambda P_{n-1} + \mu[1-(\tilde{n}-\lfloor\tilde{n}\rfloor)]P_{n+1} + \mu(\tilde{n}-\lfloor\tilde{n}\rfloor)P_{n+2} \quad (2)$$

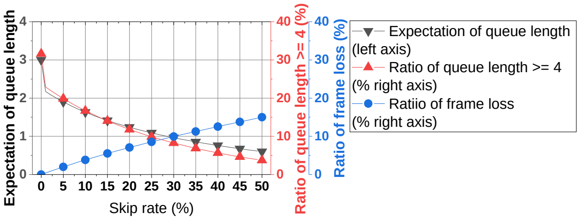


Figure 5: The expectation of queue length, ratio of queue length ≥ 4 and ratio of frame loss under different skip rate

has a solution of the form $P_n = \alpha^n C = \alpha^n P_0$ [29], combining the form to equation 2, and considering the probability must be positive, the solution would be

$$\alpha = \frac{\sqrt{1 + 4(\tilde{n} - \lfloor \tilde{n} \rfloor)\lambda/\mu} - 1}{2(\tilde{n} - \lfloor \tilde{n} \rfloor)}, \quad \tilde{n} = \frac{1}{1-q} \tag{3}$$

Along with the $P_{\tilde{0}} = (\mu/\lambda)P_0$ from balance equations and the theory $P_0 + P_{\tilde{0}} + \sum_{n=1}^{\infty} P_n = 1$ [29] we can calculate the result:

$$P_0 = \frac{\lambda(1-\alpha)}{\lambda + \mu(1-\alpha)} \tag{4}$$

and, thus

$$P_n = \frac{\alpha^n \lambda(1-\alpha)}{\lambda + \mu(1-\alpha)}, \quad n \geq 1 \tag{5}$$

So, the expectation of decoder queue length will be:

$$\begin{aligned} L_Q &= \sum_{n=1}^{\infty} n P_n \\ &= \frac{\lambda(1-\alpha)}{\lambda + \mu(1-\alpha)} \sum_{n=1}^{\infty} n\alpha^n = \frac{\lambda\alpha}{(1-\alpha)[\lambda + \mu(1-\alpha)]} \end{aligned} \tag{6}$$

where α is the equation in (3).

The expectation of the frame loss rate and tail delay are also the targets we interest in. According to Figure 4, extracting two frames and skipping one means one frame loss. So the probability of frame loss can be considered as probability of making skip action, i.e. $R_{Loss} = \sum_{n=2}^{\infty} (\tilde{n} - \lfloor \tilde{n} \rfloor) P_n$.

For the tail delay of the decoder queue, according to Figure 1, the decoder queue delay becomes noticeable when the average decoding time ≥ 12ms. So if the decoder queue length ≥ 4, the expected time of waiting for them being decoded will be ≥ 48ms. So we will set the tail delay ratio $R_{Tail} = \sum_{n=4}^{\infty} P_n$.

3.3 Numerical example

To quantify the performance and offer some recommended skip rate q, we will present a numerical illustration. We set the arrival rate in 60 frames per second, i.e. $\lambda = 60$, and set the service rate in decoder time = 12ms (having noticeable decoder queue delay ratio in Figure 1), i.e., service rate $\mu = (1/0.012) \approx 80$.

We evaluate the delayed improvement and frame loss performance under skip rate $q, q \in [0, 0.5]$. According to Figure 5, with the skip rate q increasing, the expectation of queue length and the ratio of tail delay (queue length ≥ 4) is decreasing. However, the frame loss status becomes worse when the skip rate increase. Therefore, there will be a trade-off between the delayed improvement and the frame loss.

For the skip rate chose under this trade-off, we recommend $q = 0.25$ and $q = 0.5$. With $q = 0.25$, the expectation of queue length reduces to 1, the ratio of tail delay reduced to 10%, and the frame loss rate will not exceed 9%. With $q = 0.5$, we can get a significant delayed improvement, which is attractive for those delay-sensitive RTC services like cloud gaming. We leave the further investigation of skip rate with users' experiences as our future work.

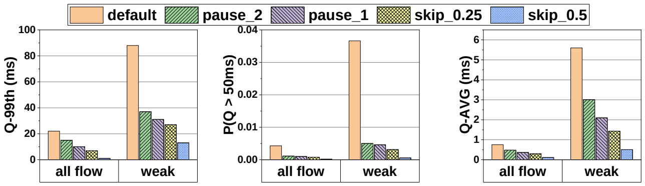


Figure 6: Simulation results of decoder queue delay (the 99%ile, the ratio of frames with decoder queue delay > 50ms, and the average

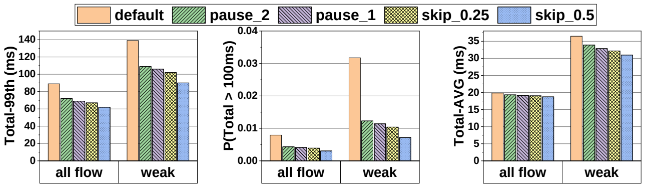


Figure 7: Simulation results of total delay (the 99%ile, the ratio of frames with total delay > 100ms, and the average

4 Evaluation

In this section, we evaluate the Frame-Skipping mechanism's performance by implementing it with a frame-level trace-driven RTC simulator, and investigate some potential obstacles in deploying Frame-Skipping.

4.1 Delayed improvement

To compare the delayed improvement to the different decoder queue management, we design a simulator that can faithfully replay the online traces on the client-side and react to the decoder queue adjustment.

4.1.1 *Simulation Setup.*

Our simulator will be running with following set up.

Baselines: To evaluate the delayed improvement of the Frame-Skipping, we also evaluate the following baselines:

- *Default.* The default queue management in our online traces service is similar to WebRTC [12]. When the decoder queue size exceeds the set threshold, the client will request a new key frame and try to flush the decoder queue.
- *Pause-encoder.* The encoder will be paused until the decoder queue length below our set threshold. Under this management, we will set the decoder queue length threshold with one (pause_1) and two (pause_2). So if the queue length exceeds the set threshold, we will send a pause command until the length below the threshold.

Traces: To investigate the Frame-Skipping performance under clients with long decoding time, the simulation will run separately with *all flow* traces and *weak* decoder traces (traces with average decoding time ≥ 12ms).

Indicators: We evaluate Frame-Skipping performance with delayed improvement and frame loss. The lower delay means better interactivity of RTC, Furthermore, the delayed improvement performance will be demonstrated by the decoder queue delay and the total delay (from user input to graphic display).

The negative effect of Frame-skipping will be demonstrated by frame loss, which would hurt smoothness and network resource utilization. However, we are not going to dig into how frame-rate degradation (60fps to 30fps under skip rate $q = 0.5$) will affect the user's experience, because, even under gaming scenario, no significant difference for quality rating, as well as performance ratings, was found between 60 fps and 25 fps [34].

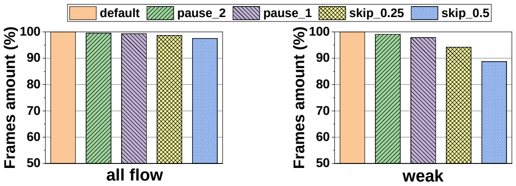


Figure 8: Simulation results of frame loss (haven't been displayed) ratio

4.1.2 *Evaluation result.*

The Frame-skipping is aiming to reducing the delay to help the interactivity service. For decoder queue delay, we present the 99th percentile, the ratio of frames with decoder queue delay > 50ms (severely queued) and the average. In Figure 6, compared with the default queue management, the Frame-Skipping with skip rate $q = 0.5$ can squeeze the 99%ile decoder queue delay from 22ms to 1ms with all flow traces and from 88ms to 13ms with weak decoder traces. For the ratio of frames with severe decoder queue delay (decoder queue delay > 50ms), Frame-Skipping can reduce the ratio by 23× with all flow traces and 53× with weak decode traces. These results indicate that the Frame-Skipping could efficiently reduce the decoder queue delay.

Moreover, the Frame-Skipping can also reduce total delay by eliminating the decoder queue delay. According to Figure 7, the Frame-Skipping with skip rate $q = 0.5$ can squeeze the 99%ile total delay from 89ms to 62ms with all flow traces and from 139ms to 90ms with weak decoder traces. For the ratio of severe total delay (total delay > 100ms), Frame-Skipping can reduce the ratio by 2.6× with all flow traces and 4.4× with weak decode traces. Therefore, the Frame-Skipping could help service providers to offer a better UHD RTC service.

For the frame loss situation, According to Figure 8, we can notice that better delayed improvement performance also means more frame loss. Because the better delayed improvement is resulted by pausing/skipping more frames.

Compare to the pause-encoder baseline, the delayed improvement performance of the Frame-Skipping is still better than the pause-encoder management (Figure 6). It illustrates that due to the limitation of processing bursty queue fluctuations (§2.2), the efficiency of eliminating decoder queue delay of arrival rate management is still not good enough compared to Frame-Skipping.

For simulation results, we also observe that all queue management can have a better delayed improvement performance under the weak decoder traces. With the significant delayed improvement of Frame-Skipping under weak decoder traces, it would be more valuable to enable Frame-Skipping for those clients with long decoding time.

4.2 Micro benchmark

We evaluate some parameters of frames encoded by reordering encoder buffer to investigate some potential obstacles in deploying Frame-Skipping.

Frame size. Under our evaluation, the frame size difference is small. Only 2% of the frames will have a more than a half difference of frame size compared to the default encoding frames. So it will not put more pressure on the network transporting. Besides, there could be a large frame size difference under the same encoding configuration [11].

Decoding time. For our test, the average decoding time between the frames encoded by reordering encoder buffer and the original frames is almost the same (decoding time difference less than 3%). So there will not be extra stress on the decoder.

Image quality. We measure the image quality with PSNR [14], which is widely used for image quality metrics. According to our evaluation, frames encoded by reordering encoder buffer will have an image quality degrade with 0.8dB PSNR score loss on average compared to the original encoding frames, i.e., 2% of image PSNR score loss.

5 Discussion

In this section, we will discuss the feasibility in the future and the character of Frame-Skipping.

Feasibility in the future. Frame-Skipping leverages SVC extension or NVIDIA codec API to reorder the encoder buffer to eliminate decoder queue delay efficiently. So we want to know if it is still feasible to reorder the encoder buffer with codec standard upgrading in the future. The scalable Video Coding (SVC) extension is now supported in H.264 and H.265 standards [2], and also be supported for the brand-new H.266 standard [36]. On the other hand, no matter what kind of codec standard will be used in the future, we can still directly invalidate the recent image in the encoder buffer to reorder encoder buffer by using encoder's API.

Advanced queue management. Many researches were also focusing on improving the streaming service experience by deploying queue management. For example, the controller will alter the sending rate of network packets to maintain a low internet queuing delay [3]. But the decoder queue is working on the video frames level, and a queued frame could introduce a 16ms delay under 60fps streaming. So it will be far more remarkable compared to the queued network packets.

Moreover, those queue managements focusing on frame level [11, 24] are located on the server's encoder side. So the long control loop will limit the eliminating efficiency of decoder queue delay as we discuss before (§4.1). However, our Frame-Skipping mechanism on the client side can instantly and efficiently eliminate the bursty queue fluctuation.

Frame-skipping scenarios. In this paper, we leverage the SVC and encoder's API to skip video frames before decoding without damaging image quality. Besides, some use cases also benefit from video discarding, such as video analytics. However, the objects of discarding frames in Frame-Skipping and video analytics are totally different. Frame-Skipping is aiming to offer ultra-low latency Real-time Communication by eliminating the queuing delay. On the other hand, video analytics is filtering out/discarding frames to solve the challenges of high compute and network resource demands of video streaming and analysis models [18, 21]. The disparity of optimizing objects makes the frames discarding scheduling under different scenarios different.

6 Conclusion

In this paper, we propose a Frame-Skipping mechanism to efficiently eliminate the decoder queue delay by extracting neighbouring frames in the head of the queue with skipping frames. We also introduce a mathematical model to quantify the Frame-Skipping performance and offer some recommended skip rates. Then, we demonstrate the Frame-Skipping can efficiently reduce the decoder queue delay and total delay by evaluating it on our trace-driven simulator.

Acknowledgement

This paper is supported by the National Key R&D Program of China under Grant 2019YFB1802504 and the National Natural Science Foundation of China under Grant 61625203. Mingwei Xu is the corresponding author.

References

[1] Giuseppe Bonaccorso. 2017. *Machine learning algorithms.* Packt Publishing Ltd.

[2] Jill M Boyce, Yan Ye, Jianle Chen, and Adarsh K Ramasubramonian. 2015. Overview of SHVC: Scalable extensions of the high efficiency video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology* 26, 1 (2015), 20–34.

[3] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2017. Congestion control for web real-time communication. *IEEE/ACM Transactions on Networking* 25, 5 (2017), 2629–2642.

[4] Marc Carrascosa and Boris Bellalta. 2020. Cloud-gaming: Analysis of Google Stadia traffic. *arXiv preprint arXiv:2009.09786* (2020).

[5] Zhilong Chen, Hancheng Cao, Yuting Deng, Xuan Gao, Jinghua Piao, Fengli Xu, Yu Zhang, and Yong Li. 2020. Learning from Home: A Mixed-Methods Analysis of Live Streaming Based Remote Education Experience in Chinese Colleges During the COVID-19 Pandemic. *arXiv preprint arXiv:2010.01662* (2020).

[6] Jacob Willem Cohen. 2012. *The single server queue.* Elsevier.

[7] Lorenzo Corneo, Maximilian Eder, Nitinder Mohan, Aleksandr Zavodovski, and Suzan BayhanZ. 2021. Surrounded by the Clouds. In *The Web Conference.*

[8] Nvidia Corporation. 2020. NVIDIA VIDEO CODEC SDK - ENCODER Programming Guide. https://docs.nvidia.com/video-technologies/video-codec-sdk/pdf/NVENC_VideoEncoder_API_ProgGuide.pdf.

[9] John C Doyle, Bruce A Francis, and Allen R Tannenbaum. 2013. *Feedback control theory.* Courier Corporation.

[10] Adam N Elmachtoub and Paul Grigas. 2021. Smart "predict, then optimize". *Management Science* (2021).

[11] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S Wahby, and Keith Winstein. 2018. Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18).* 267–282.

[12] Google LLC. 2018. *webrtc.* https://chromium.googlesource.com/external/webrtc

[13] Jakob Hohwy. 2017. Priors in perception: Top-down modulation, Bayesian perceptual learning rate, and prediction error minimization. *Consciousness and Cognition* 47 (2017), 75–85.

[14] Alain Hore and Djemel Ziou. 2010. Image quality metrics: PSNR vs. SSIM. In *2010 20th international conference on pattern recognition.* IEEE, 2366–2369.

[15] Chun-Ying Huang, Cheng-Hsin Hsu, Yu-Chun Chang, and Kuan-Ta Chen. 2013. GamingAnywhere: An open cloud gaming system. In *Proceedings of the 4th ACM multimedia systems conference.* 36–47.

[16] Gazi Karam Illahi, Thomas Van Gemert, Matti Siekkinen, Enrico Masala, Antti Oulasvirta, and Antti Ylä-Jääski. 2020. Cloud gaming with foveated video encoding. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 16, 1 (2020), 1–24.

[17] Megan M Jack, Domenico A Gattozzi, Paul J Camarata, and Kushal J Shah. 2021. Live-streaming surgery for medical student education-educational solutions in neurosurgery during the COVID-19 pandemic. *Journal of surgical education* 78, 1 (2021), 99–103.

[18] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication.* 253–266.

[19] Nikolaos Kontorinis, Yiannis Andreopoulos, and Mihaela Van Der Schaar. 2009. Statistical framework for video decoding complexity modeling and prediction. *IEEE transactions on circuits and systems for video technology* 19, 7 (2009), 1000–1013.

[20] Advait Lad, Shivani Butala, and Pramod Bide. 2019. A comparative analysis of over-the-top platforms: Amazon Prime Video and Netflix. In *International Conference on Communication and Intelligent Systems.* Springer, Singapore, 283–299.

[21] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. 2020. Reducto: On-camera filtering for resource-efficient real-time video analytics. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication.* 359–376.

[22] Xiaofei Liao, Hai Jin, Yunhao Liu, Lionel M Ni, and Dafu Deng. 2006. Anysee: Peer-to-peer live streaming. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications.* Citeseer, 1–10.

[23] Zhiwen Liao and Ling Zhang. 2020. Scheduling Dynamic Multicast Requests in Advance Reservation Environment for Enterprise Video Conferencing Systems. *IEEE Access* 8 (2020), 76913–76928.

[24] Yixiang Mao, Liyang Sun, Yong Liu, and Yao Wang. 2020. Low-latency FoV-adaptive Coding and Streaming for Interactive 360° Video Streaming. In *Proceedings of the 28th ACM International Conference on Multimedia.* 3696–3704.

[25] Vishal Misra, Wei-Bo Gong, and Don Towsley. 2000. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication.* 151–160.

[26] Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. 2020. Pruning Edge Research with Latency Shears. In *Proc. ACM HotNets.*

[27] Rong Pan, Preethi Natarajan, Chiara Piglione, Mythili Suryanarayana Prabhu, Vijay Subramanian, Fred Baker, and Bill VerSteeg. 2013. PIE: A lightweight control scheme to address the bufferbloat problem. In *2013 IEEE 14th international conference on high performance switching and routing (HPSR).* IEEE, 148–155.

[28] Johan Pouwelse, Koen Langendoen, R Lagendijk, and Henk Sips. 2001. Power-aware video decoding. In *22nd Picture Coding Symposium, Seoul, Korea.* Citeseer, 303–306.

[29] Sheldon M Ross. 2014. *Introduction to probability models.* Academic press.

[30] Sandvine. 2020. The Global Internet Phenomena Report COVID-19 Spotlight. https://www.sandvine.com/covid-internet-spotlight-report.

[31] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. 2007. Overview of the scalable video coding extension of the H. 264/AVC standard. *IEEE Transactions on circuits and systems for video technology* 17, 9 (2007), 1103–1120.

[32] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. 2003. Overview of the H. 264/AVC video coding standard. *IEEE Transactions on circuits and systems for video technology* 13, 7 (2003), 560–576.

[33] Adam Wierman and Takayuki Osogami. 2003. A unified framework for modeling TCP-Vegas, TCP-SACK, and TCP-Reno. In *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003.* IEEE, 269–278.

[34] Saman Zadtootaghaj, Steven Schmidt, and Sebastian Möller. 2018. Modeling gaming QoE: Towards the impact of frame rate and bit rate on cloud gaming. In *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX).* IEEE, 1–6.

[35] Anfu Zhou, Huanhuan Zhang, Guangyuan Su, Leilei Wu, Ruoxuan Ma, Zhen Meng, Xinyu Zhang, Xiufeng Xie, Huadong Ma, and Xiaojiang Chen. 2019. Learning to coordinate video codec with transport protocol for mobile video telephony. In *The 25th Annual International Conference on Mobile Computing and Networking.* 1–16.

[36] Bin Zhu, Shan Liu, Yuan Liu, Yi Luo, Jing Ye, Haiyan Xu, Ying Huang, Hualong Jiao, Xiaozhong Xu, Xianguo Zhang, et al. 2020. A software decoder implementation for H. 266/VVC video coding standard. *arXiv preprint arXiv:2012.02832* (2020).