Cooperative Video Quality Adaptation for Delay-Sensitive Dynamic Streaming using Adaptive Super-Resolution

Minseok Choi[†], Won Joon Yun[*], and Joongheon Kim[*]
[†]Department of Electronic Engineering, Kyung Hee University, Yongin, South Korea
[*]School of Electrical Engineering, Korea University, Seoul, South Korea
E-mails: choims@khu.ac.kr, {ywjoon95, joongheon}@korea.ac.kr

***Abstract*—This paper proposes a cooperative and dynamic quality adaptation scheme for delay-sensitive video streaming between the transmitter and the receiver. We present a novel adaptive super-resolution (SR) technique that adaptively controls the quality enhancement rate and computation time. Due to the capability of enhancing the quality of video chunks at the user device side, the transmitter can aggressively transcode video chunks to reduce the delivery latency and power consumption. Also, adaptive SR can control the tradeoff between playback stall rate and CPU consumption of the user device. Simulation results verify the performance of adaptive SR and show that the proposed video delivery scheme is very good to balance tradeoff among the following performance metrics for online video services: 1) playback stall rate, 2) average quality measure, 3) transmission power, and 4) CPU consumption of the user device.**

I. Introduction

Motivated by improved capability of computations at mobile devices, dynamic quality adaptation for video streaming becomes available depending on the network state and computational resources of user devices. As online video services dominate the global data traffic and recent multimedia services (e.g., UHD streaming, AR, VR) require much higher data traffic [1], there have been extensive researches on reducing data traffic, resource consumption, network costs, and energy consumption incurred by video streaming. At the same time, a variety of the quality-of-experience (QoE) components of video services, such as average video quality, playback stall rate, and latency, should be largely studied together [2].

As the video streaming system has launched in wireless networks, limited wireless resources and time-varying channel conditions become significant bottlenecks in achieving high QoE. Therefore, dynamic adaptive streaming over HTTP (DASH) was designed to dynamically select the bitrate of video streaming depending on the wireless network state [3]. Because a video stream is partitioned into sequential segments or chunks to be delivered, adaptive bitrate (ABR) streaming that allows video chunks to have different bitrates has been extensively researched [4]. Quality adaptation and scheduling were jointly optimized to maximize the expected bitrate of the entire streaming while limiting playback stall events in [5]. Similarly, adaptive bitrate selection for each delivering video chunk and power allocation were jointly optimized to maximize the average quality measure of users in [6]. Further, the authors of [7] managed a variety of QoE components, such as playback stall rate, average bitrate, quality fluctuations, and packet drop rate, and balanced the tradeoff among them by dynamically controlling streaming quality and transmission rate. There have been also many studies on machine learning-based ABR streaming schemes. In [8], *QFlow*, which is a reinforcement learning approach of selecting bitrates for wireless streaming by adaptively controlling flow assignments to queues, was introduced. A deep reinforcement learning (DRL) approach was adopted to jointly control transmission power and ABR streaming [9].

As mobile edge computing (MEC) technologies have been extensively developed, even small base stations (BSs) and mobile devices can transcode video chunks before delivering them to reduce the latency and communication overheads. The authors of [10] provided the low-latency ABR streaming by jointly optimizing computational transcoding tasks and video delivery when scalable video coding (SVC) is utilized for encoding the video file. Also, a joint transcoding and caching strategy of partial video files with different quality levels were designed to provide smooth and high-quality video services [11]. In [12], the DRL-based quality adaptation scheme was proposed to balance the tradeoff between the QoE of video services and transcoding costs.

In parallel, super-resolution (SR) has been extensively popular for enhancing video quality, and modern SR techniques are majorly developed using deep neural networks (DNNs) [13]. There have been extensive researches on SR technique itself; however, application of SR to ABR streaming has not been broadly studied yet. Recently, *Dejavu* that utilizes historical sessions to improve the quality of real-time videoconferencing was proposed [14], and the DNN-based SR technique was applied to the ABR streaming system [15]. However, the above studies have not considered the adaptive SR which dynamically controls the quality enhancement level depending on receiver and networks states, and not jointly optimized the video delivery together.

When the DNN for SR is deployed at the mobile device having sufficient computational resources, the transmitter can aggressively transcode video chunks to reduce the delivery latency and power consumption. However, SR tasks could

ISBN 978-3-903176-49-2 © 2022 IFIP

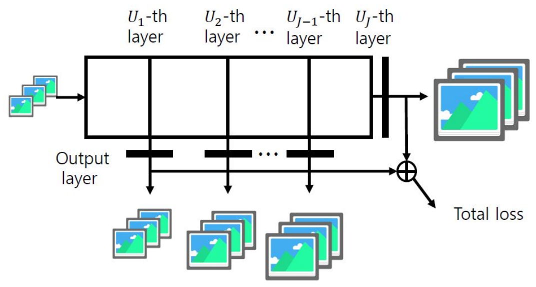

Fig. 1: Adaptive SR network

require huge computational overheads and generate excessive inference delays; therefore, the authors of [16] offload the computational SR task to the cloud to avoid excessive latency, but it requires upload and download of video files causing additional delays. Accordingly, this paper considers an adaptive control of SR at the receiver by designing the DNN for SR based on the anytime neural network (ANN) [17]. The ANN allows coarse and quick inference results from the middle of the DNN blocks; therefore, we can dynamically choose the quality of output images by selecting the block to extract the prediction result. Thus, the main goal of this paper is to adapt the transcoding rate at the transmitter and the quality enhancement rate using ANN-based SR at the receiver for maximizing the average video quality while limiting the playback latency. The main contributions of this paper are as follows:

- This paper presents the adaptive SR network by applying the ANN to the SR network for dynamically controlling the quality of prediction results and the required computations. Our SR network is based on the generative adversarial network (GAN) for image SR (SRGAN).
- This paper proposes the cooperative video quality adaptation by jointly selecting the transcoding rate at the transmitter and the quality enhancement rate of SR at the receiver depending on the network state and queue states of the transmitter and the receiver. The proposed scheme also efficiently consumes transmission power and CPU resources of the receiver for operating SR.
- Simulation results show that the proposed cooperative video quality adaptation scheme provides better performances than ABR streaming without using adaptive SR and balances the tradeoff among a variety of performance metrics, such as playback stall rate, average quality measure, transmission power and CPU consumption of the receiver very well.

The rest of the paper is organized as follows. The adaptive SR network is explained in Section II, and the video streaming system using adaptive SR is described in Section III. The cooperative quality adaptation scheme for delay-sensitive video streaming is proposed in Section IV. Simulation results are presented in Section V, and Section VI concludes this paper.

II. Adaptive Super-Resolution Network

The SR technique has been firstly developed by minimizing the mean squared error (MSE) using the convolutional neural network (CNN); therefore, the SR convolutional neural network (SRCNN) was proposed [13]. Even though the MSE-based SR achieves high peak-signal-to-noise-ratio (PSNR) and structural similarity index measure (SSIM) for output images, it suffers from an ill-posed problem in which a high-quality texture could be lost during extracting coarse predictions and averaging multiple output levels from different layers. Therefore, we choose the SRGAN that is the SR network based on GAN for designing the adaptive SR network because it achieves soft textures and smooth images [18].

The SRGAN consists of generator G, discriminator D, and feature extractor ϕ, and their model parameters are denoted by θ^G, θ^D, and θ^ϕ, respectively. G has a role of operating SR, i.e., enhancing the quality of an input video chunks, and its adversarial loss is derived as D evaluates the prediction result of G. Also, ϕ converts input images to a feature space to compute the content loss obtained by minimizing the Euclidean distance between feature $\phi_{\theta_\phi}(I^{HR})$ extracted from a high-resolution image and feature $\phi_{\theta_\phi}(I^{SR})$ extracted from an image created by G. Then, the perceptual loss is generated from both adversarial loss and content loss, and the SRGAN trained for minimizing the perceptual loss could avoid the ill-posed problem. Here, the VGG19 network [19] which is pretrained with ImageNet consisting of 14 million images [20] is adopted as ϕ.

To adaptively control the quality of output chunks and the required computational burden, we apply the idea of the ANN [17] to the SRGAN, in which auxiliary prediction results are extracted from the middle of the residual layers. Let the model of G have total U layers, and auxiliary prediction results can be obtained from the U_j-th residual layers for all $j \in \mathcal{J} \triangleq \{1, \cdots, J\}$, satisfying $U_1 < U_2 < \cdots < U_J$ and $U = U_J$. As shown in Fig. 1, additional output layers are used to extract auxiliary prediction results from every U_j-th residual layer, and the model parameter set of partial U_j residual blocks of G is denoted by θ_j^G. In order to allow quick and coarse prediction results from the middle of the model, we aim at training the adaptive SR network (i.e., G) by minimizing losses of multiple outputs from the U_j-the residual layers for all $j \in \mathcal{J}$.

According to [18], the total loss function is comprised of the MSE ℓ_{MSE}, Euclidean distance loss ℓ_D, and adversarial loss ℓ_{Gen}. Each loss component is defined as

$$\ell_{MSE} \triangleq \|I^{HR} - I^{SR}\|_2 \tag{1}$$

$$\ell_D \triangleq \|\phi(I^{HR}) - \phi(I^{SR})\|_2 \tag{2}$$

$$\ell_{Gen} \triangleq -\log D(I^{SR}), \tag{3}$$

where $\|.\|_2$ stands for the L2 loss, I^{HR} is the lossless high-resolution image, and I^{SR} is the output image of G. Particularly, ℓ_D measures the Euclidean distance between features of I^{HR} and I^{SR} extracted by ϕ, and L_{Gen} is based on failure probabilities of D for discriminating the output image of G and I^{HR}.

The auxiliary prediction results can be generated by θ_j^G for any $j \in \mathcal{J}$; therefore, we have to design the total loss function of the adaptive SR network using losses of output images from the U_j-th residual layers for all j as follows:

$$\ell_{ASR} = w_M \ell^{\star}_{MSE} + w_V \ell^{\star}_{D} + w_G \ell^{\star}_{Gen}, \tag{4}$$

where $\ell^{\star}_{MSE} \triangleq \sum_{j=1}^{J} \delta_j \ell_{MSE,j}$, $\ell^{\star}_{D} \triangleq \sum_{j=1}^{J} \delta_j \ell_{D,j}$, $\ell^{\star}_{Gen} \triangleq \sum_{j=1}^{J} \delta_j \ell_{Gen,j}$, and w_M, w_V and w_G are the scaling coefficients for $\ell^{\star}_{MSE}$, $\ell^{\star}_{D}$ and $\ell^{\star}_{Gen}$, respectively. Also, δ_j is the weight factor for the loss of the output from the U_j-th layer, satisfying $\sum_{j=1}^{J} \delta_j = 1$; therefore, $\ell^{\star}_{MSE}$, $\ell^{\star}_{D}$ and $\ell^{\star}_{Gen}$ are the weighted averages of ℓ_{MSE}, ℓ_D and ℓ_{Gen} generated by θ^G_j for all $j \in \mathcal{J}$. Here, $\ell_{MSE,j}$, $\ell_{D,j}$, and $\ell_{Gen,j}$ are the MSE loss, the Euclidean distance loss, and the adversarial loss of the output image generated by θ^G_j, respectively. According to [17], fore layers of G are strongly trained by minimizing multiple losses caused by different U_j-th residual layers; on the other hand, rear layers of G can be trained from the result predicted by the entire model θ^G. Accordingly, we use $\delta_i < \delta_j$ for any $i < j$ and $i, j \in \mathcal{J}$.

III. Video Streaming System using Adaptive Super-Resolution

This paper considers dynamic streaming with the help of adaptive SR. Let the transmitter have the capability of transcoding the high-quality video chunks and the user device have sufficient computation resources to operate adaptive SR. This section describes video transcoding and delivery at the transmitter, and quality enhancement of received video chunks at the receiver.

A. Video Transcoding and Delivery at Transmitter

Suppose that the transmitter has highest-quality videos only and can transcode them to lower quality versions with any transcoding rate $r \in \mathcal{R} \triangleq \{1, \cdots, R\}$. Each video file is partitioned into sequential chunks, and each chunk can be separately transcoded and delivered. The video chunks with the highest quality requested by the user are accumulated in the transmitter queue to be delivered, and its queue length is updated by $Q_{t+1} = \max\{Q_t - K_t + v_t, 0\}$ and $Q_0 = 0$, where Q_t is the queue length, K_t is the number of delivering chunks at slot t, and v_t is the number of desired chunks at slot t. We assume that v_t follows the uniform distribution, i.e., $v_t \sim \mathcal{U}[0, v_{\max}]$. Before delivering K_t chunks, the transmitter transcodes them to reduce communication latency with the rate r_t, as shown in Fig. 1. Denote $\mathcal{P}(r_t)$ and $M(r_t)$ as the quality measure and the size of a chunk transcoded with the rate r_t.

The Rayleigh fading channel model is assumed, and the channel gain between the transmitter and the receiver is denoted by $h = \sqrt{S}g$ where $S = 1/d^{\gamma}$ denotes the inverse of the path loss, γ is the path loss exponent, and $g \sim \mathcal{CN}(0, 1)$ is the fast fading component. Also, we assume block fading; therefore, h is static during one discretized time slot. Then, the channel capacity C_t limits the number of delivering chunks (i.e., K_t) depending on r_t as follows:

$$K_t M(r_t) \leq C_t = t_0 \mathcal{B} \log_2 \Big(1 + \frac{P_t |h_t|^2}{\sigma^2}\Big), \tag{5}$$

where h_t is the channel gain at slot t, t_0 is the time duration of each slot, $\mathcal{B}$ is the bandwidth, and σ^2 is the noise variance. The transmission power budget $P_{\max}$ is given, i.e., $P_t \leq P_{\max}$; therefore, we can assume $\mathbb{E}[K_t] \leq K_{\max}$. Also, for power efficiency, we limit the long-term expected power consumption by threshold P_0.

Thus, the transmitter determines 1) how many chunks to deliver (i.e., K_t), 2) how much to transcode them (i.e., r_t), and 3) how much to consume power for delivering (i.e., P_t), for limiting the queuing delay and saving power consumption.

B. Video Quality Enhancement at Receiver

The receiver has the pre-trained adaptive SR network explained in Section II and sufficient computation resources to operate it. When delivered video chunks arrive, the receiver first makes decision on the layer number l_t of the adaptive SR network to extract the output image, meaning that the quality of an input chunks is enhanced by $\theta^G_{l_t}$ and the auxiliary prediction result from the U_{l_t}-th residual layer of G is used as an output chunk. Here, $l_t \in \mathcal{J}$, and if $l_t = 0$, the receiver directly plays the arrived K_t chunks without enhancing their quality. Also, the receiver determines the number of CPU cycles denoted by c_t to use for operating adaptive SR. Then, the inference time of the adaptive SR network for each video chunk depends on the layer number to extract the output image, and the number of CPU cycles for operating SR, which is denoted by $\tau(r_t, l_t, c_t)$. We can experimentally measure the expected value of $\tau(l_t, c_t)$ for each combination of $\{l_t, c_t\}$ as shown in Table II. Note that $\tau(l_t, c_t) = 0$ when $c_t = 0$.

Consider the receiver buffer in which the expected inference time of adaptive SR for received video chunks is accumulated. The buffer dynamics is formulated by $B_{t+1} = \max\{B_t + \mu_t - t_0, 0\}$ and $Z_0 = 0$, where the arrival process μ_t is the expected inference time for enhancing the quality of video chunks arrived at slot t; therefore, it is defined as $\mu_t = K_t \cdot \tau(l_t, c_t)$. Because we assume that $\mathbb{E}[K_t] \leq K_{\max}$ and $\tau(l_t, c_t)$ is finite, $\mathbb{E}[\mu_t] \leq \mu_{\max}$ is satisfied. The receiver can justify whether it aggressively exploits the adaptive SR network for enhancing the quality of received chunks at the expense of the inference delay and CPU consumption or not by observing how long B_t is. If B_t is excessively large, the user could experience a playback stall because a SR task of the next chunk to be played may not be completed yet.

After finishing SR operation for the received K_t video chunks, the user can play these chunks. The final streaming quality depends on both transcoding at the transmitter and adaptive SR at the receiver, and its measure is denoted by $\mathcal{P}(r_t, l_t)$. Therefore, cooperative quality adaptation between the transmitter and the receiver is required for maximizing the streaming quality.

IV. Cooperative Quality Adaptation and Resource Allocation for Video Streaming

This section proposes the jointly optimized cooperative quality adaptation and resource allocation that maximizes the expected video quality while limiting the playback latency and resource consumption. However, since the transmitter dynamically controls amounts of delivering chunks at each slot, we cannot expect how many chunks will be delivered during the long-term time duration. Therefore, according to [6], we minimize the entire quality degradation compared to the highest-quality measure $\overline{\mathcal{P}}$, as follows:

$$\min_{\mathbf{\Psi}} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\Big[\Big(\overline{\mathcal{P}} - \mathcal{P}(r_t, l_t)\Big)K_t\Big] \tag{6}$$

$$\text{s.t.} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[Q_t] < \infty \tag{7}$$

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[B_t] < \infty \tag{8}$$

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[P_t] \le \overline{P} \tag{9}$$

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[c_t] \le \overline{c} \tag{10}$$

$$K_t M(r_t) \le t_0 C(P_t) \tag{11}$$

$$0 \le P_t \le P_{\text{max}} \tag{12}$$

$$r_t \in \mathcal{R},\ l_t \in \mathcal{J},\ c_t \in \mathcal{C}, \tag{13}$$

where $\mathbf{\Psi} = \{\mathbf{K}, \mathbf{r}, \mathbf{P}, \mathbf{l}, \mathbf{c}\}$ which is a decision parameter set. Here, $\mathbf{K} = [K_0, \cdots, K_T]$, and $\mathbf{r}$, $\mathbf{P}$, $\mathbf{l}$ and $\mathbf{c}$ are defined in a similar manner. We assume that there are finite number $|\mathcal{C}|$ of available CPU cores so that $\mathcal{C} \triangleq \{1, \cdots, c_{\text{max}}\}$. Also, $\overline{P}$ and $\overline{c}$ are target thresholds for the average transmit power and the average CPU usage, respectively, and P_{max} is the power budget. Expectations of (6)–(10) are with respect to random channel realizations. The constraints of (7) and (8) are for limiting the queueing delay and the inference delay of adaptive SR. The transmit power consumption and usage of CPU cores are limited by the constraints of (9) and (10), respectively, and the constraint (11) comes from (5), which demonstrates that decisions on K_t and r_t depend on the channel capacity.

Note that constraints (7) and (8) pursue stability of the transmitter queue and the buffer, respectively, rather than guaranteeing strict upper bounds. However, it is very difficult to make the instantaneous queuing delay bounded to a given value due to dynamic delivery decisions and time-varying channel conditions; therefore, we focus on limiting the time-average queue lengths because the average queueing delay is proportional to the average queue length [21]. Also, according to the Lyapunov optimization theory [21], the time-average queue length can be limited by pursuing strong stability of a queue, as shown in (7) and (8).

To deal with (9) and (10), we present the virtual queues $W(t)$ and $V(t)$ whose update equations are given by

$$W_{t+1} = \max\{W_t - \overline{P} + P_t, 0\} \tag{14}$$

$$V_{t+1} = \max\{V_t - \overline{c} + c_t, 0\}. \tag{15}$$

According to [22], the strong stability of W_t and V_t pushes the long-term time average of P_t and c_t to be smaller than $\overline{P}$ and $\overline{c}$, respectively. Then, let $\Theta_t = [Q_t, B_t, W_t, V_t]^T$ and define $\mathfrak{L}(\Theta_t) = \frac{1}{2}\{Q_t^2 + k_B B_t^2 + k_W W_t^2 + k_V V_t^2\}$, where k_B, k_W and k_V are scaling coefficients. Then, let $\Delta(.)$ be a conditional quadratic Lyapunov drift on t that is formulated as $\mathbb{E}[\mathfrak{L}(\Theta_{t+1}) - \mathfrak{L}(\Theta_t)|\Theta_t]$. According to Lyapunov optimization theory [22], the dynamic policy is designed to solve the given optimization problem by observing Θ_t and choosing Ψ_t to minimize a bound on *drift-plus-penalty* which is given by

$$\Delta(\Theta_t) + A \cdot \mathbb{E}\Big[\Big(\overline{\mathcal{P}} - \mathcal{P}(r_t, l_t)\Big)K_t\Big], \tag{16}$$

where A is a system parameter that gives a weight to the objective function of the problem in (6)–(13).

Then, the upper bound on the Lyapunov drift can be obtained as

$$\begin{aligned}\mathfrak{L}(\Theta_{t+1}) - \mathfrak{L}(\Theta_t) \le \frac{1}{2}\Big\{ & K_t^2 + v_t^2 + k_B(\mu_t^2 + t_0^2) \\ & + 2Q_t(\mu_t - K_t) + 2k_B B_t(\mu_t - t_0) + k_W(P_t - \overline{P})^2 \\ & + k_V(V_t - \overline{c})^2 + 2k_W W_t(P_t - \overline{P}) + 2k_V V_t(c_t - \overline{c})\Big\} \\ \le H + \Big\{ & Q_t(v_t - K_t) + k_B B_t(\mu_t - t_0) \\ & + k_W W_t(P_t - \overline{P}) + k_V V_t(c_t - \overline{c})\Big\},\end{aligned} \tag{17}$$

where H is chosen to satisfy the following inequality:

$$v_{\text{max}}^2 + K_{\text{max}}^2 + \mu_{\text{max}}^2 + t_0^2 + \overline{P}^2 + P_{\text{max}}^2 + \overline{c}^2 + c_{\text{max}}^2 \le 2H. \tag{18}$$

According to (16) and (17), when Θ_t is observed, minimizing a bound on drift-plus-penalty is consistent with minimizing

$$\begin{aligned}&-\mathbb{E}[Q_t K_t] + \mathbb{E}[k_B B_t \mu_t] + \mathbb{E}[k_W W_t P_t] \\ &\quad + \mathbb{E}[k_V V_t c_t] + A \cdot \mathbb{E}\Big[(\overline{\mathcal{P}} - \mathcal{P}(r_t, l_t))K_t\Big],\end{aligned} \tag{19}$$

Here, we use the concept of opportunistically minimizing the expectations; therefore, (19) is minimized by independently choosing $\Psi_t = \{K_t, r_t, P_t, l_t, c_t\}$ to minimize

$$\begin{aligned}\mathcal{D}(K_t, r_t, P_t, l_t, c_t) &= -Q_t K_t + k_B B_t K_t \tau(l_t, c_t) \\ &\quad + k_W W_t P_t + k_V V_t c_t + A \cdot K_t(\overline{\mathcal{P}} - \mathcal{P}(r_t, l_t)).\end{aligned} \tag{20}$$

Thus, we can reformulate the long-term time-average problem of (6)–(13) into the opportunistic problem as follows:

$$\min_{\Psi_t} \mathcal{D}(K_t, r_t, P_t, l_t, c_t) \tag{21}$$

$$\text{s.t.}\ (11), (12), (13). \tag{22}$$

The appropriate initial values of k_B, k_V, k_w and A need to be obtained experimentally because they depend on the channel environments and constraints on performances (i.e., $\overline{P}$ and $\overline{c}$). Also, all weight parameters should be positive.

To efficiently exploit the channel condition, if r_t and P_t are given, the transmitter is better to deliver as many chunks as possible because the average queuing delay is proportional to the expected queue length [21]. Therefore, the inequality constraint on the data rate in (11) can be converted into the equality constraint. Then, according to (11), the optimal power to deliver K_t transcoded with rate r_t is

$$P_t^* = \frac{\sigma^2}{|h_t|^2}\Big(2^{\frac{K_t M(r_t)}{t_0 B}} - 1\Big). \tag{23}$$

Then, we formulate the subproblem with respect K_t and c_t

for given $r_t = r$ and $l_t = l$ as follows:

$$\min_{K_t, c_t} k_W W_t \frac{1}{|h_t|^2}\left(2^{\frac{K_t r_t}{t_0 \mathcal{B}}} - 1\right) + k_V V_t c_t$$
$$+ K_t\Big[k_B B_t \tau(l_t, c_t) - Q_t + A \cdot (\overline{\mathcal{P}} - \mathcal{P}(r_t, l_t))\Big]. \quad (24)$$

Denote the optimal solutions of (24) by K'_t and c'_t without any constraints. Then, the problem in (24) is convex and can be solved by using Karush–Kuhn–Tucker (KKT) conditions.

If $k_B B_t \tau(l_t, c_t) - Q_t + A \cdot (\overline{\mathcal{P}} - \mathcal{P}(r_t, d_t)) < 0$, K'_t and c'_t should satisfy the following equations for achieving KKT conditions:

$$k_W \frac{W_t}{|h_t|^2} \frac{M(r_t)}{t_0 \mathcal{B}} \ln 2 \cdot 2^{\frac{M(r_t)}{t_0 \mathcal{B}} K'_t} + A \cdot (\bar{\mathcal{P}} - \mathcal{P}(r_t, l_t)) - Q_t$$
$$+ k_B B_t c_t \tau(l_t, c_t) \sqrt{\frac{k_B B_t c_t \tau(l_t, c_t) K'_t}{k_V V_t}} = 0 \quad (25)$$

$$c'_t = \sqrt{\frac{k_B B_t c_t \tau(l_t, c_t) K'_t}{k_V V_t}}. \quad (26)$$

Otherwise, $K'_t = 0$ and $u'_t = 0$, which means that no chunk is delivered and the receiver does not consume CPU resources.

Since $c_t \in \mathcal{C} = \{0, 1, \cdots, c_{\max}\}$ and K_t is a nonnegative integer, we have to compare boundary conditions depending on the value of c'_t for obtaining the optimal K^*_t and c^*_t with consideration of the boundary constraints on K_t and c_t. If $0 \leq c'_t \leq c_{\max}$, four possible combinations of $K^*_t \in \{\lfloor K'_t \rfloor, \lceil K'_t \rceil\}$ and $c*_t \in \{\lfloor c'_t \rfloor, \lceil c'_t \rceil\}$ are compared. Meanwhile, if $c'_t \geq c_{\max}$, we compare the following three conditions as follows: 1) $K^*_t = 0$ and $c^*_t = 0$, 2) $K^*_t = \lfloor K'_t \rfloor$ and $c^*_t = c_{\max}$, and 3) $K^*_t = \lceil K'_t \rceil$ and $c^*_t = c_{\max}$. Then, we find the optimal solution of the problem in (21)–(22) by greedily testing all joint combinations of decisions on r_t and l_t. The details are given in Algorithm 1.

V. Performance Evaluation

This section first demonstrates the proposed adaptive SR network, and provides performances of the proposed quality adaptation scheme for a delay-sensitive dynamic streaming.

A. Adaptive Super-Resolution Network

Youtube 8K MPEG-DASH dataset is used to train and test the adaptive SR network [23]. The preprocessed dataset is randomly cropped with the size of $144 \times\ 144 \times 3$. Note that video images with resolutions of 360p, 480p, 720p, and 1080p are bicubic interpolated from the highest-resolution video (i.e., 2160p) with transcoding rate $r = \{6, 4, 3, 2\}$, respectively. When training the adaptive SR network, the training set and batch contain all pre-processed images regardless of r, and parameters of $w_M = 3.03$, $w_V = 3.03$, $w_G = 10^{-2}$, $\delta_5 = 0.0165$, $\delta_{10} = 0.0660$, $\delta_{15} = 0.264$, $\delta_{20} = 0.323$, and $\delta_{25} = 0.330$ are used.

The adaptive SR network consists of $U_J = 25$ residual blocks, and all residual blocks have 64 filters with a 3×3 kernal-size, the stride of 1 and the padding of 1. The *Adam* optimizer is used for both generator and discriminator with a learning rate of 0.001.

Algorithm 1 Solving problem in (21)–(22)

```
1: At slot $t$, $Q_t$, $B_t$, $W_t$, $V_t$, and $\mathcal{D}^* = 10^{10}$ are given.
2: for $r^*_t \in \mathcal{R}$ and $l^*_t \in \mathcal{J}$ do
3:     if $k_B B_t \tau(l_t, c_t) - Q_t + A \cdot (\overline{\mathcal{P}} - \mathcal{P}(r_t, d_t)) \geq 0$ then
4:         $K'_t = 0$ and $c'_t = 0$
5:     else
6:         Obtain $K'_t$ and $c'_t$ by (25) and (26)
7:         if $0 \leq c'_t \leq c_{\max}$ then
8:             Compare four combinations of $K^*_t \in \{\lfloor K'_t \rfloor, \lceil K'_t \rceil\}$ and $c^*_t \in \{\lfloor c'_t \rfloor, \lceil c'_t \rceil\}$ and pick one of them minimizing (24)
9:         else
10:            Compare the following conditions and pick one of them minimizing (24): 1) $K^*_t = 0$ and $c^*_t = 0$, 2) $K^*_t = \lfloor K'_t \rfloor$ and $c^*_t = c_{\max}$, and 3) $K^*_t = \lceil K'_t \rceil$ and $c^*_t = c_{\max}$
11:        end if
12:        Obtain $P^*_t$ by (23) and $\mathcal{D}(\Psi^*_t)$ by (20)
13:        if $\mathcal{D}(\Psi^*_t) < \mathcal{D}^\star$ then
14:            $\mathcal{D}^\star \leftarrow \mathcal{D}(\Psi^*_t)$, $K^\star_t \leftarrow K^*_t$, $r^\star_t \leftarrow r^*_t$
15:            $P^\star_t \leftarrow P^*_t$, $l^\star_t \leftarrow l^*_t$, $c^\star_t \leftarrow c^*_t$
16:        end if
17:    end if
18: end for
```

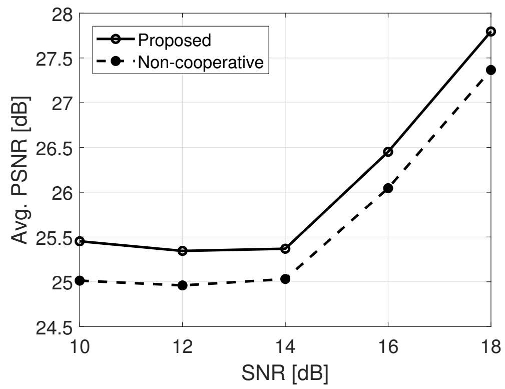

Fig. 2: Average PSNR

Table I summarizes PSNR and SSIM measures of output images of adaptive SR depending on the layer number to extract the prediction result and the transcoding rate. Also, the required CPU cycles and inference time to extract the output image from different layers are shown in Table II. As explained in Section II, as the deeper layer generates the prediction result, the quality of an input image becomes more enhanced at the expense of CPU consumption and processing delay. Based on this specification of the adaptive SR network, we simulate the proposed cooperative quality adaptation algorithm for dynamic video streaming.

B. Numerical Results of Dynamic Video Streaming

In this subsection, $\overline{P} = 2.5$, $\overline{c} = 2.5$, $A = 0.01$, $\mathcal{B} = 3$ MHz, $t_0 = 1$ sec, $K_{\mathbf{max}} = 30$, $c_{\mathbf{max}} = 10$, $k_B = 1.0$,

TABLE I: Quality measure (i.e., PSNR, and SSIM) of output images of the adaptive SR network

PSNR/SSIM	Baseline	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
$r = 2$	30.60/0.859	30.66/0.866	31.01/0.885	31.87/0.899	32.20/0.904	32.26/0.906
$r = 3$	27.45/0.749	27.51/0.773	28.42/0.795	29.08/0.816	29.39/0.823	29.45/0.826
$r = 4$	25.82/0.672	26.29/0.701	26.85/0.724	27.49/0.749	27.73/0.759	27.79/0.762
$r = 5$	24.81/0.620	25.22/0.641	25.74/0.663	26.23/0.688	26.46/0.698	26.54/0.702

TABLE II: Computational cost metrics of adaptive SR

Metric	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
Clocks $[10^9]$	1.007	1.441	1.864	2.283	2.669
Inference time [ms]	8.6	12.3	15.9	19.4	22.8

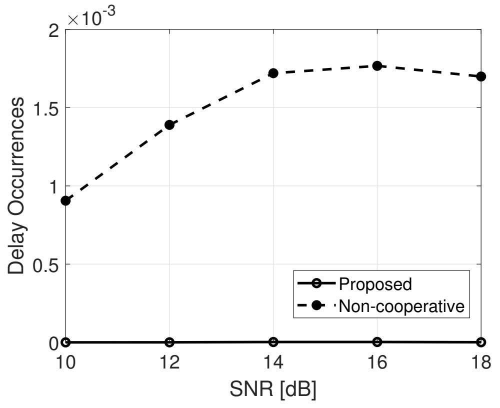

Fig. 3: Average delay incidence

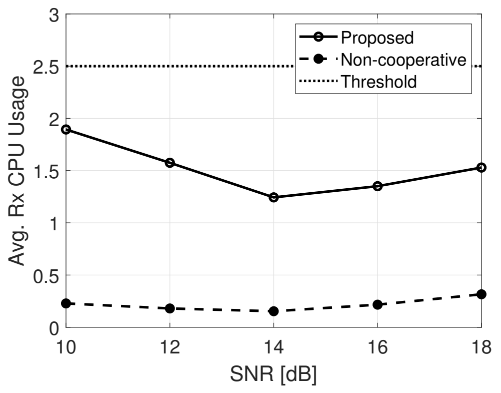

Fig. 5: CPU consumption

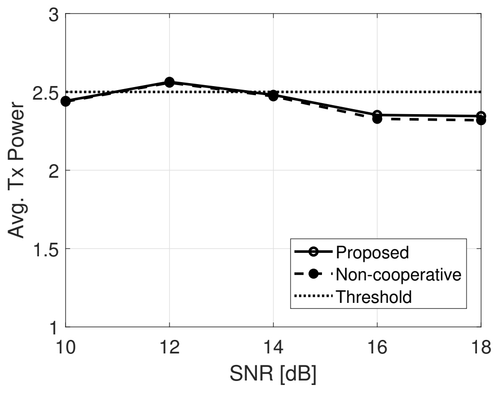

Fig. 4: Transmission power consumption

$k_W = 1.0$, and $k_V = 1.0$ are used. We consider the non-adaptive SR scheme that fully enhances the quality of received chunks always and joint optimization process for other decision parameters (i.e., K_t, r_t, P_t, and c_t) is identical to that of the proposed one as a comparison technique. Figs. 2–5 show the plots of average PSNR, delay incidence, transmit power consumption, and CPU usage versus the SNR, respectively. Because the final quality of video chunks depends on both transcoding at the transmitter and SR at the receiver, the proposed scheme that jointly controls both decisions provides better a PSNR performance than the non-cooperative scheme overall. The transmitter and the receiver of the non-cooperative scheme separately control their decisions in sequence, the receiver decisions (i.e., layers of the adaptive SR network and CPU cycles) are strictly dependent on the transmitter decisions (i.e., number of delivering chunks, transcoding rate, transmit power). Therefore, we can observe that transmit power consumption of both schemes is very similar; meanwhile, the proposed scheme outperforms the non-cooperative one in terms of the performance metrics related to receiver decisions (i.e., delay incidence and CPU consumption).

Both techniques satisfy the target thresholds for transmit power and CPU resources; however, the non-cooperative scheme spends much lower CPU cycles than the available amounts. From this point, we realize that our scheme efficiently controls the tradeoff among the video quality, delay incidence, consumption of transmitter and receiver resources. The proposed scheme spends more CPU cycles but not larger than a threshold value to improve the final quality of video chunks while even guaranteeing almost zero playback delay.

VI. Conclusion

This paper proposed an adaptive SR network and a cooperative quality adaptation technique for delay-sensitive video streaming using adaptive SR. Inspired by ANN [17], we build an adaptive SR network that can adaptively control a quality enhancement level and amounts of required computations. With the help of adaptive SR at the receiver, the transmitter can aggressively transcode video chunks to reduce the queuing delay and to save transmit power. This paper jointly optimizes transcoding at the transmitter, adaptive SR operation at the receiver, and resource allocations to maximize the video quality while limiting the playback delay. Numerical results show that the proposed scheme strikes a balance among a variety of conflicting performance metrics of video streaming carefully, i.e., video quality, playback delay incidence, transmit power consumption, and CPU usage.

ACKNOWLEDGMENT

This work was in part supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2021-0-02201, Federated Learning for Privacy-Preserving Video Caching Networks) and the National Research Foundation of Korea(NRF) grant funded by the Korea govenment(MSIT) (NRF-2022R1C1C1010766).

REFERENCES

[1] Cisco Visual Networking Index: Forecast and Trends, 2017–2022. Accessed: Mar. 3, 2020.

[2] Y. Chen, K. Wu and Q. Zhang, "From QoS to QoE: A Tutorial on Video Quality Assessment," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 1126-1165, Second quarter 2015.

[3] T. Stockhammer. 2011. Dynamic adaptive streaming over HTTP – standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems (MMSys '11)*. Association for Computing Machinery, New York, NY, USA, 133–144.

[4] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer and R. Zimmermann, "A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 562-585, First quarter 2019.

[5] M. Choi, A. F. Molisch and J. Kim, "Joint Distributed Link Scheduling and Power Allocation for Content Delivery in Wireless Caching Networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 12, pp. 7810-7824, Dec. 2020.

[6] M. Choi, A. No, M. Ji and J. Kim, "Markov Decision Policies for Dynamic Video Delivery in Wireless Caching Networks", *IEEE Transactions on Wireless Communications*, vol. 18, no. 12, pp. 5705-5718, Dec. 2019.

[7] W. J. Yun, D. Kwon, M. Choi, J. Kim, G. Caire and A. F. Molisch, "Quality-Aware Deep Reinforcement Learning for Streaming in Infrastructure-Assisted Connected Vehicles," *IEEE Transactions on Vehicular Technology*W, vol. 71, no. 2, pp. 2002-2017, Feb. 2022.

[8] R. Bhattacharyya, A. Bura, D. Rengarajan, M. Rumuly, S. Shakkottai, D. Kalathil, Ricky K. P. Mok, and A. Dhamdhere. 2019. QFlow: A Reinforcement Learning Approach to High QoE Video Streaming over Wireless Networks. In *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc '19)*. Association for Computing Machinery, New York, NY, USA, 251–260.

[9] C. Ye, M. C. Gursoy and S. Velipasalar, "Power Control for Wireless VBR Video Streaming: From Optimization to Reinforcement Learning," *IEEE Transactions on Communications*, vol. 67, no. 8, pp. 5629-5644, Aug. 2019.

[10] H. Zhou, X. Wang, Z. Liu, Y. Ji and S. Yamada, "Resource Allocation for SVC Streaming Over Cooperative Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 7924-7936, Sept. 2018.

[11] D. Kim and M. Choi, "Impacts of Device Caching of Content Fractions on Expected Content Quality," *IEEE Wireless Communications Letters*, vol. 11, no. 5, pp. 1022-1026, May 2022.

[12] Y. Guo, F. R. Yu, J. An, K. Yang, C. Yu and V. C. M. Leung, "Adaptive Bitrate Streaming in Wireless Networks With Transcoding at Network Edge Using Deep Reinforcement Learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 3879-3892, April 2020.

[13] C. Dong, C. C. Loy, K. He and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, June 2015.

[14] P. Hu, R. Misra, and S. Katti. 2019. Dejavu: Enhancing Videoconferencing with Prior Knowledge. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications (HotMobile '19)*. Association for Computing Machinery, New York, NY, USA, 63–68.

[15] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han, "Neural adaptive content-aware internet video delivery," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 645–661.

[16] J. Yi, S. Kim, J. Kim and S. Choi, "Supremo: Cloud-Assisted Low-Latency Super-Resolution in Mobile Devices," *IEEE Transactions on Mobile Computing*, Early Access.

[17] H. Hu, D. Dey, M. Hebert, J. A. and Bagnell, "Anytime neural network: a versatile trade-off between computation and accuracy," *arXiv preprint arXiv:1708.06832 (2018)*.

[18] C. Ledig, L. Theis, F. Husz´ar, J. Caballero, A. Cunningham, A. Acosta, A. Alejandro and A. Andrew, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4681–4690, 2017.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *in Proc. International Conference on Learning Representations (ICLR)*, 2015.

[20] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, Li. Fei-Fei, "Imagenet: A large-scale hierarchical image database," *in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Miami, FL, USA, June 2009, pp. 248–255.

[21] D. Bertsekas and R. G. Gallager, *Data Networks*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1992.

[22] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Claypool, 2010.

[23] M. Choi, W. J. Yun, J. Kim "ABR-ASRGAN," GitHub repository, 2021, available at: https://github.com/WonJoon-Yun/ABR-ASRGAN