Video Super-Resolution and Caching - An Edge-Assisted Adaptive Video Streaming Solution

Aoyang Zhang, Qing Li, *Member, IEEE,* Ying Chen, Xiaoteng Ma, Longhao Zou, *Member, IEEE,* Yong Jiang, *Member, IEEE,* Zhimin Xu, and Gabriel-Miro Muntean, *Senior Member, IEEE*

***Abstract*—Edge computing provides the potential to improve users' Quality of Experience (QoE) in ever-increasing video delivery. However, existing edge-based solutions cannot fully utilize the edge computing power and storage capacity. This paper proposes VIdeo Super-resolution and CAching (VISCA), an edge-assisted adaptive video streaming solution, which integrates super-resolution and edge caching to improve users' QoE. We design a novel edge-based ABR algorithm that makes bitrate and video chunk source decisions by considering network conditions, QoE objectives, and edge resource availability jointly. VISCA utilizes super-resolution to enhance the cached low-quality video at the edge. The super-resolution models used are trained for the most popular videos only in order to achieve quality improvements with a fraction of the computation. A novel cache strategy is also adopted to maximize caching efficiency. To assess the performance of VISCA, an implemented prototype of VISCA was deployed in synthetic and real network contexts. Compared with the existing video streaming solutions, VISCA improves video quality by 28.2%-251.2% and reduces rebuffering time by 16.1%-95.6% in all considered scenarios.**

***Index Terms*—DASH, edge computing, cache, QoE, super-resolution**

I. Introduction

Recently, video traffic has become the primary source of Internet traffic. It will account for about 82% of all Internet traffic by 2022 [1]. This surge in video traffic leads to severe pressure on network bandwidth. Although 5G promises to increase bandwidth at the last mile, the backhaul networks will still be the bottleneck for the massive video delivery in a long time [2]. Delivering videos with high Quality of Experience (QoE) to users has become the fundamental challenge in the video delivery scenario. Many studies have shown that users will quickly abandon video sessions if the quality is not sufficient, leading to significant losses in revenue for content providers [3], [4]. Therefore, it is crucial for content providers to improve users' QoE for better client engagement.

A. Zhang, Y. Chen and X. Ma are with the Tsinghua-Berkeley Shenzhen Institute, Tsinghua University, Shenzhen, China (Email: zhang-ay18@mails.tsinghua.edu.cn; chen-yin18@mails.tsinghua.edu.cn; maxt17@mails.tsinghua.edu.cn).

Q. Li and L. Zou are with the Southern University of Science and Technology and also with PCL Research Center of Networks and Communications, Peng Cheng Laboratory (PCL), Shenzhen, China (Email: liq8@sustech.edu.cn; zoulh@sustech.edu.cn).

Y. Jiang is with the Tsinghua Shenzhen International Graduate School, Shenzhen, China (Email: jiangy@sz.tsinghua.edu.cn).

Z. Xu is with the Multimedia Lab in Beijing Bytedance Technology Co., Ltd., Beijing, China (Email: xuzhimin@bytedance.com).

G.-M. Muntean is with the Performance Engineering Lab, Dublin City University, Ireland (Email: gabriel.muntean@dcu.ie).

Corresponding authors: Qing Li (liq8@sustech.edu.cn), Longhao Zou (zoulh@sustech.edu.cn)

In order to improve users' QoE under scarce bandwidth, Dynamic Adaptive Streaming over HTTP (DASH) [5] is provided as a standard to improve the utilization of bandwidth for efficient and easy video delivery. Based on DASH, clients employ Adaptive Bitrate (ABR) algorithms to pick the bitrate for the next DASH chunk according to the current network status [6], [7], users' buffer size [8], [9], or joint consideration of both aspects [10], [11]. These client-based ABR algorithms adaptively select the most suitable bitrate to improve QoE under the real-time network. However, each user chooses bitrate according to its network conditions and playback conditions locally, lacking a global view.

Super-resolution is also adopted in video delivery to break the strong dependency between the network condition and users' QoE. In recent studies, NAS [12] and SRAVS [13] integrates super-resolution into adaptive streaming at the client-side to improve users' QoE under limited bandwidth. Even though integrating super-resolution at the client-side improves video quality, the performance of super-resolution is fundamentally constrained by the computational capacity of terminal devices. Compared to terminal devices, an edge platform can provide sufficient computing power to reduce super-resolution latency significantly. Since the reconstructed results cannot be reused in the client-side scheme, the super-resolution can be utilized more effectively at the edge platform by providing the same reconstructed video for multiple clients.

The emerging edge computing [14] breaks the conventional end-to-end infrastructure for video streaming and brings more potential to improve video quality. Current edge-based solutions mainly focus on leveraging edge caching capability (i.e., CDN and LEAP [15]) to reduce the transmission redundancy or take advantage of edge computing power to make bitrate decisions more accurate and flexible [16], [17]. Even though the edge-based schemes are well developed, these solutions lack multi-dimensional joint optimization of network bandwidth resources and edge computing power. Thus improving users' QoE remains a challenging and active research topic under unstable and insufficient network bandwidth.

In this paper, we propose **VI**deo **S**uper-resolution and **CA**ching (VISCA) - an edge-assisted adaptive video streaming solution which integrates super-resolution and edge caching to improve users' QoE. We design a novel edge-based ABR algorithm that can make bitrate and video chunk source decisions by considering network conditions, QoE objectives, and edge resources jointly. VISCA enhances the cached low-quality video by super-resolution at the edge, which provides high video quality even under excessive backhaul network bandwidth pressure (leading to poor bandwidth conditions). In order to ensure the reliability of the enhanced quality provided by super-resolution, we train content-aware models for only the most popular videos according to highly skewed video popularity distribution based on the analysis of 31-days video data from Douyin, achieving most of the quality improvements with only a small fraction of the computation. We design a novel combined-utility based caching strategy to reduce the transmission redundancy. The combined-utility based caching strategy caches the most valuable chunks that quantify the video's contribution to improving the user's QoE by considering the video's popularity, video quality, and the potential benefits of performing super-resolution.

VISCA is instantiated in a prototype system based on Nginx [18], uWSGI [19], and Django [20]. Experiments over real-world videos and bandwidth traces show VISCA outperforms previously proposed methods. In particular, it improves user QoE between 71.7-149.1 compared to BOLA [8] and between 26.1-30.7 compared to MPC [10]. Also, we provide a performance analysis of individual system components. When the backhaul network is very limited (4Mbps), VISCA can reach 10.1 times the total network throughput perceived by users.

The main contributions of this paper are as follows:

- We design a comprehensive video delivery framework VISCA. To our best knowledge, our framework is the first to integrate video super-resolution with caching intelligently at the edge. With multi-dimensional joint optimization over network resources and edge resources, we enhance video quality and improve users' QoE under scarce network conditions.
- We propose an edge-based integrated ABR algorithm to make decisions intelligently according to dynamical network bandwidth, QoE objectives, edge caching status, and edge computing capacity.
- We design a novel combined-utility based cache strategy by thoroughly considering the caching benefit of a video and its potential enhancement by super-resolution.

II. Background and Related Works

Adaptive Streaming. Dynamic Adaptive Streaming over HTTP (DASH) has emerged as a key technology to enhance bandwidth utilization for video delivery. DASH encodes a video session into multiple chunks along with several bitrate levels. Adaptive Bitrate (ABR) algorithms are the primary tool that DASH integrates on the client-side to dynamically selects the most appropriate bitrate for each video chunk. These solutions can be roughly divided into three categories: (1) rate-based algorithms [6], [7], (2) buffer-based algorithms [8], [9] and (3) a combination of both rate and buffer information [10], [11]. Rate-based algorithms (i.e., RB [6]) select the highest possible bitrate based on the estimated available throughput. Buffer-based algorithms (i.e., BOLA [8]) make bitrate decisions according to the buffer occupancy level of video players. MPC [10] selects the most appropriate bitrate by solving a QoE maximization problem based on both rate and buffer data. The QoE-driven strategy [11] takes into account the instant bitrate of each segment and the buffer reservation for future streaming to optimize users' QoE. The authors of [7] proposed an ensemble rate adaption framework for DASH to adapt different network conditions, optimizing users' QoE. These client-based approaches make their own decision on bitrate selection to improve users' QoE. Therefore, they are not capable of optimizing users QoE globally over a region. For those users who share the same bandwidth, their QoE would still suffer when the network becomes unfavorable.

Super-resolution. Super-resolution refers to both single image super-resolution and video super-resolution. Single image super-resolution [21], [22], [23] uses a single image only to reconstruct the corresponding high-resolution image. Video super-resolution [24], [25] is an extension and development of the single image super-resolution approach, which maps low-resolution to high-resolution frames with the consideration of the temporal information. Video super-resolution models that consider temporal consistency can achieve better video reconstruction results than the single image super-resolution model, but video super-resolution also requires more computing power and inference time. Recent research using deep neural networks (DNN) based super-resolution [26] has demonstrated a significant performance improvement compared to non-DNN methods [27]. Frame-Recurrent Video Super-Resolution (FRVSR) [28] is a state-of-the-art video super-resolution approach that naturally encourages temporally consistent results without much increase in computational costs. In VISCA, we apply FRVSR at the edge to reduce the data transmission demand over the backhaul network by reconstructing low-quality videos.

Video Delivery System with Super-resolution. NAS [12], and SRAVS [13] integrates super-resolution into adaptive streaming at the client-side to mitigate the influence of dynamic network conditions on users' QoE. NAS modifies MDSR [29] to adapt to the heterogeneous computing capabilities of clients. NAS requires a device with high computing power, such as a PC, to apply it. Considering the limited computing capacity of terminal devices, SRAVS chooses SRCNN [21], a simple, lightweight super-resolution model with only three convolutional layers. The key limitation of deploying super-resolution at the client-side is the computational capacity of terminal devices. Compared to terminal devices, an edge platform can provide enough computing capacity to reduce super-resolution latency. Furthermore, high-quality videos reconstructed by super-resolution at the client-side only serve a single client, but the integration of super-resolution at the edge can use the reconstructed high-quality video more effectively by providing the same reconstructed video for multiple clients.

Network-based Caching. Edge-based caching improves Internet video delivery, including solutions that use Content Delivery Networks (CDN) and cache proxies. To fully utilize this infrastructure, various caching algorithms have been proposed. These include offline cache schemes, which use complex algorithms to cache the most popular content [30], [31] and cache schemes that explore the request pattern to make cache decisions [32], [33]. However, these strategies do not consider the use of edge computing resources, and their performance highly depends on the backhaul network conditions. Aiming to increase user QoE, it is essential to employ edge computing

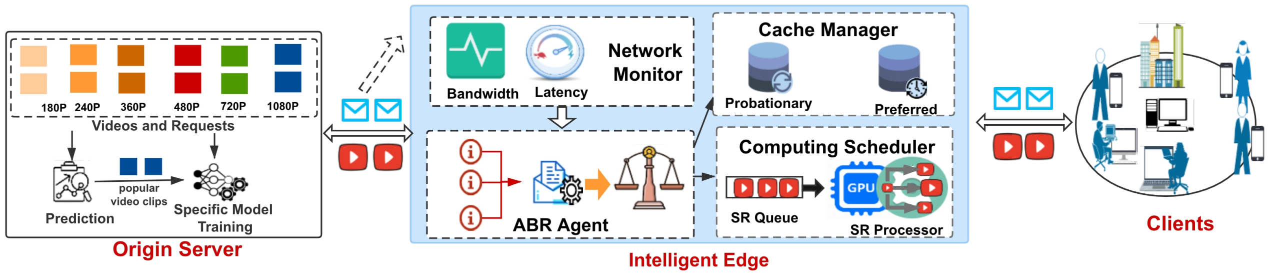


Fig. 1: The architecture of VISCA.

resources and optimize network resource utilization.

III. System Model Overview

VIdeo Super-resolution and CAching (**VISCA**) is an edge-assisted adaptive video streaming solution, which aims to improve users' QoE under limited backhaul bandwidth. As shown in Fig. 1, VISCA includes three components, which are introduced next.

Origin Server Functions. Raw videos are first uploaded to an origin server and are encoded into multiple streams with different predetermined bitrates. The server then splits each stream into a sequence of small video chunks. VISCA also includes a popularity prediction module and a super-resolution training module deployed on the origin server. Because the origin server has sufficient computing power, the super-resolution model training tasks are completed on the origin server. The specific super-resolution models are trained for the most popular videos only in order to achieve most of the quality improvements with a fraction of the computation. The functions of the origin server are described in detail in Section IV.

Intelligent Edge Functions. Intelligent Edge is the edge node with open and standard mechanisms to provide powerful computational resources and cache space. VISCA deploys the video super-resolution function at Intelligent Edge, aiming to improve users' QoE under poor network conditions by utilizing edge computing power and storage capacity. As illustrated in Fig. 1, Intelligent Edge of VISCA consists of four modules: Computing Scheduler, Cache Manager, Network Monitor, and ABR Agent. Computing Scheduler is responsible for computing resource scheduling for super-resolution tasks. Cache Manager records historical video information and manages the cached contents accordingly. Network Monitor measures the dynamic network status between edges and origin servers for ABR Agent to adapt to network changes. ABR Agent accesses client states, public network status, and edge resource information to handle all user requests appropriately. These modules cooperate and interact with each other to serve the users covered by the edge. The design of Intelligent Edge is detailed in Section V.

Client-side Behavior. There are various end devices. Clients send video chunk requests to the edge platform. Upon the arrival of a new client, the client shares the end device type and the highest display resolution of the device. Additionally, the end devices share the current playback status with the edge platform, including the current buffering level and the last chunk quality. The client is a *Simple Client* without ABR algorithms, which only send the request of video id and chunk number, and the edge makes the bitrate adaptation decision for each client. The selected bitrate will not exceed the maximum display quality of the end device.

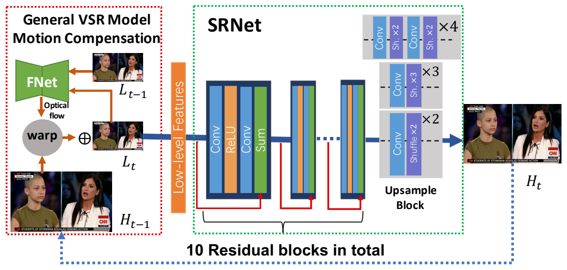


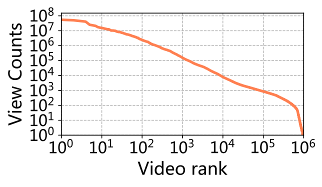
Fig. 2: Video super-resolution model architecture.

IV. Origin Server Design

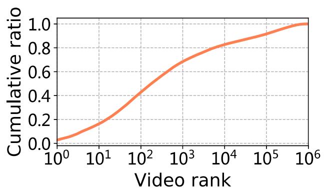
The origin server is responsible for raw video processing and super-resolution model training. Based on the DASH standard, a video is encoded to a set of standard resolution versions R and divided into chunks with the same duration. v_n denotes the n_{th} chunk of a video file v and $v_{n,r}$ denotes the n_{th} chunk encoded at resolution $r \in R$. In the case of limited computing resources, to improve the super-resolution model's training efficiency, we only train the specific super-resolution model for the most popular videos. This section introduces the architecture of our general super-resolution model, the process of video selection for specific super-resolution model training, and the design of specific super-resolution models.

A. Architecture of General Super-resolution Model

FRVSR [28] model is a state-of-the-art application of Deep Learning in the field of super-resolution, the architecture of which is shown in Fig. 2. The super-resolution model produces a high-resolution frame H_t from an input low-resolution frame L_t and recursively uses the previously generated output H_{t-1}. FNet estimates the optical flow m_t from the low-resolution (LR) frame L_{t-1} to L_t and hence yields the normalized LR flow map. The LR flow map is then upscaled to the corresponding high-resolution (HR) flow map. The warp operation uses the HR flow map to warp the previous HR-estimate frame onto the current LR frame. The high-level structure of the super-resolution model can be summarized as:



(a) View Counts



(b) Cumulative Ratio

Fig. 3: View counts distribution of Douyin

$$m_t = FNet(L_{t-1}, L_t), \quad (1)$$

$$H_t = SRNet(L_t, W(H_{t-1}, m_t)). \quad (2)$$

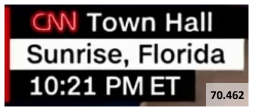
Here, W denotes warping operations. Inputting the previously estimated frame H_{t-1} can help SRNet reuse previously generated results more easily, thereby improving the accuracy of the reconstruction. We adopt three upsample scales x2, x3, x4 in FRVSR. For instance, the target resolution $r^{'}$, i.e., 720P, can be obtained from a base resolution r, i.e., 180P, with the upsample scale x4. Moreover, our super-resolution model supports multi-scale super-resolution (upsample scale x2, x3, x4) in a single network while sharing the intermediate layers to minimize the bandwidth resource consumption between origin and edge servers and the storage usage.

Video Multimethod Assessment Fusion (VMAF) [34] is used to evaluate the effect of video super-resolution on video quality improvement. VMAF estimates the subjective quality of users by combining multiple elementary quality metrics. These quality metrics include not only image quality metrics, but also the temporal characteristics of videos. Compared to traditional image quality metrics such as PSNR [35], and SSIM [36], VMAF correlates stronger with subjective scores and achieves more accurate results in estimating user perceived video quality [37]. A VMAF score is between 0 and 100: a score of 0-20 is considered as unacceptable, 20-40 as poor, 40-60 as fair, 60-80 as good, and 80-100 as excellent [34]. VMAF($r, r^{'}$) for $r \in R$, $r^{'} \in R$ and $r <= r^{'}$ represents the VMAF of reconstructed video in the target resolution $r^{'}$ from the base resolution r.

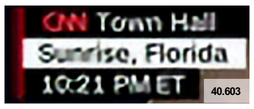
B. Video Selection for Specific Model Training

A generalized super-resolution model used for all videos introduces a large variance in the quality of the reconstructed videos. One promising direction is to train a super-resolution model for each content separately, which achieves more reliable and accurate reconstruction results for the corresponding content [38]. However, we cannot develop a super-resolution model for each video due to limited computing resources. To overcome this limitation, we exploit the video popularity distribution to achieve the greatest improvement using a small fraction of the computation.

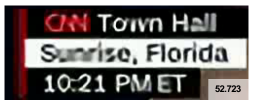
Based on our preliminary study, video popularity distribution is highly skewed. Fig. 3a shows the view counts of 1 million randomly sampled videos in 31 days that we collected from Douyin, the leading short video platform in China. The most popular video in the sample set has been viewed about 54.4 million times, while the $10,000^{th}$ most popular video has only 8,000 views. We analyze our video data using Zipf distribution [30] and Weibull distribution [39], as they are widely used for video popularity distribution. Our analysis shows that the Zipf distribution is more suitable for our video data as the Zipf can fit the long tail portion of our video data better than the Weibull distribution. The shown distribution of view counts follows a power-law with exponent $\alpha = 1.86$. The cumulative ratio of view counts represented by videos within a given rank is depicted in Fig. 3b. The top 1% of videos account for 82.7% of the total view counts. Thus, if we use the maximum computing resources on training specific super-resolution models for 1% of videos, we can benefit from increased streaming quality for 82.7% of all video requests. Therefore, the origin server focus on training specific super-resolution models for the most popular videos.



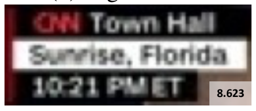
(a) LightSR-SP



(b) LightSR-GE



(c) FRVSR-GE



(d) 180P Input

Fig. 4: 180P to 720P super-resolution results.

TABLE I: Parameters and FLOPs over different resolutions

Method	Param	180P x4	240P x4	360P x4
FRVSR	2.811M	97.822G	172.252G	382.781G
LightSR	1.038M	86.792G	154.056G	347.169G

The super-resolution models are divided into two types: *General super-resolution model*, which is generalized to improve the quality for most videos. For training the general model, we build a dataset consisting of various types of videos, e.g., movie, drama, and talk-show. *Specific super-resolution model*, which is trained only using the corresponding video frames, leveraging the overfitting property of a deep neural network to further improve the reconstruction quality on a specific video. As shown in Fig. 1, we train specific super-resolution models for the most popular video clips obtained from our video popularity prediction model. The goal of the prediction model is to predict the future popularity, in terms of the total number of views of a video at the target day t_t, given data from the first t_r days ($t_t < t_r$). We use a multivariate linear regression model [40] to predict the future popularity based on the number of views per hour of each video over the first t_r days in the sample set.

C. Design of Specific Super-resolution Model

The introduction of optical flow estimation and warp operations in the motion compensation part of Fig. 2 increases model complexity. We found that for a specific super-resolution model, even if there is no motion compensation part, a good super-resolution result can still be obtained by leveraging the over-fitting property of DNN. Considering the high real-time requirements of video transmission, we train the specific model as a *LightSR* model without the motion compensation part, as shown in Fig. 2, which only uses L_t to generate the output. Table I reports the number of parameters and the FLOPs of FRVSR and LightSR super-resolution models. Compared with FRVSR, LightSR requires less computation and storage space.

Fig. 4 shows snapshots of video quality enhancement results of a 180P video by the upsample scale x4. Visually, with the general LightSR model (LightSR-GE) and the general FRVSR model (FRVSR-GE), the video quality has been significantly improved compared to the original 180P video. The specific LightSR model (LightSR-SP) achieves the highest video quality, which is closer to the original 720P video. The bottom right corner of each picture in Fig. 4 is the VMAF score, the quality of the 180P video has improved from the unacceptable level to the fair level by the general LightSR and the general FRVSR. The result of the specific LightSR achieves a good level, which improves the VMAF of the original 180P video by ten times. Therefore, training LightSR on a specific video can achieve higher reconstruction quality with less computing and storage costs. Hence, we can use the least amount of computing resources to maximize users' QoE. The specific and general models are trained on the origin server. Intelligent Edge uses the trained super-resolution model to enhance the video quality. The VMAF score of a reconstructed video is fixed and can be treated as prior knowledge for video streaming. The quality information of reconstructed videos is recorded in the corresponding MPD file and shares with the edges to distinguish between the original high-resolution video and the reconstructed video.

V. Intelligent Edge Design

Intelligent Edge is a key component of VISCA. Intelligent Edge utilizes its computing and caching resources to improve video quality while ensuring fast response time, ultimately providing higher users' QoE. This section describes the function of each module in Intelligent Edge and the process of cooperation between each module in detail.

A. Computing Scheduler

Computing Scheduler is responsible for intelligently scheduling edge computing resources to complete super-resolution (SR) tasks, consisting a SR waiting queue and a SR processor. We first store the general super-resolution model at the edge side. When the cumulative amount of requests for a video received by the edge exceeds a threshold, and the bandwidth is sufficient, Computing Scheduler sends a request to fetch the specific model from the origin server and save it at the edge.

We build a waiting queue Q for the SR processor, following the First In First Out rule, which ensures the sequential execution of tasks. Once ABR Agent decides to respond to this video request by enhancing the cached chunk, the corresponding cached lower resolution chunk is then extracted from the cache space and moved to Computing Scheduler. If the SR processor is available, the current task will be executed immediately and then returned to the user. Otherwise, the task will be added to the waiting queue until there are available computing resources. The entire super-resolution task includes three steps: the SR processor first decodes the video chunk into frames, then puts them into the video super-resolution model, and lastly re-encodes the processed frames.

The inference time of a video chunk is only related to the base resolution and the upsample scale (x2, x3, x4). For a certain base resolution, the higher the target resolution is, the longer the inference time. For a fixed upsample scale model, the higher the input resolution, the longer the inference time. We denote $\phi(r, r^{'})$ as the processing time of reconstructing a chunk from resolution r to resolution $r^{'}$ and F represents the set of all legal reconstruction pairs. All reconstruction time of $(r, r^{'}) \in F$ is derived offline.

B. Cache Manager

In VISCA, the performance of super-resolution is also influenced by the cache strategy. Super-resolution can be performed only if the low-resolution video chunk has already been cached. Thus, it is significant to envision a more suitable cache strategy to cooperate with the super-resolution method, so that we construct a combined-utility based cache strategy.

1) Deriving Cache Utility: The combined-utility is used to represent the contribution of a chunk to improving the average QoE among covered clients.

Video quality is included in the combined-utility as it is a key factor affecting users' QoE. Other factors (such as rebuffering, smoothness) are not only related to the video itself but also affected by bandwidth conditions, so they are not included in the utility. As a key factor affecting users' QoE, video quality is included in the combined-utility. Other key components of QoE objectives (such as rebuffering, smoothness) are not included in the utility, since they are not only related to the video itself but also affected by bandwidth conditions. The popularity of a video is another key factor in the combined-utility. Only a few videos are popular and account for the majority of viewers, so caching the most popular videos can reduce latency and download time [41].

Combined-utility also considers the potential super-resolution benefit. There are two sides of super-resolution processing: the potential gain and the potential cost. The video quality obtained by super-resolution is positively correlated with the base video chunk quality, which brings a potential quality gain on the base video. For a given base resolution, the larger the upsample scale, the greater the video quality, but the longer the super-resolution process takes. The cost of super-resolution can be represented by the inference time of super-resolution. Therefore, we denote γ the super-resolution potential benefit, which is negatively correlated to super-resolution processing time. The decision of caching a video chunk is based on its combined-utility. A higher combined-utility value of a video chunk indicates a higher possibility of being cached. All super-resolution video chunks are treated in the same way, but these chunks obtained by super-resolution cannot be super-resolved again because a reconstructed video may lose some feature information of the original video due to a learning error. The utility of chunk $v_{n,r}$ is defined as:

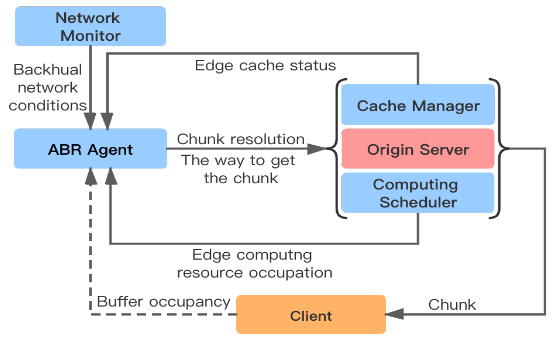


Fig. 5: ABR Agent workflow

$$CU_{v_{n,r}} = (1+\gamma) \times (\alpha q(v_{n,r}) + \beta Pop(v_{n,r})),$$
$$\gamma = \begin{cases} e^{-\frac{\sum \phi(r,r')}{3}} & \forall (r,r') \in F \\ 0 & \forall (r,r') \notin F \end{cases} \tag{3}$$

where $q(v_{n,r})$ denotes the chunk quality (VMAF score) and $Pop(v_{n,r})$ denotes the total number of chunk requests of $v_{n,r}$ collected by the edge over a period P, as the popularity of videos is found to follow a periodic pattern in [42]. $q(v_{n,r})$ and $Pop(v_{n,r})$ are normalized to compute cache utility. For a given base resolution in our system, it consists of three legal pairs based on the number of upsample scales we adopted. We compute the average super-resolution processing time over these three legal sets. α and β are the weight of video quality and popularity, respectively.

2) Cache Replacement Algorithm: We also considered the efficiency of cache replacement. A study shows that most chunks cached upon the first request had never been requested again, which reduces the efficiency of caching and leads to frequent cache replacement [33]. We address the issue by dividing the cache storage into two parts: a probationary cache C_{pb} and a preferred cache C_{pre}. C_{pb} caches video chunks for a short term, whereas C_{pre} is designed for long-term caching. Cached content in C_{pb} and C_{pre} are recorded in $Clist_{pb}$ and $Clist_{pre}$, respectively. The job of the combined-utility based cache strategy has two parts:

- C_{pb}: Upon the arrival of a video chunk from the origin server, the cache manager caches it in C_{pb}. If the cache space of C_{pb} is insufficient, the oldest items will be replaced by using the Least Recently Used (LRU) cache replacement algorithm. For all $v_{n,r} \in Clist_{pb}$, we accumulate the number of requests $CNT_{v_{n,r}}$, and keep track of the information related to the combined utility over this period. The video chunk $v_{n,r}$, with $CNT_{v_{n,r}} > 1$, has greater chances to be requested repeatedly [33] will be moved to C_{pre}.
- C_{pre}: C_{pre} stores chunks that have a high possibility to be frequently requested. To move these chunks in C_{pb} into the long-term cache space, we first calculate $CU_{v_{n,r}}$ for all cached chunks in the preferred part and the video chunks $v_{n,r} \in C_{pb}$ with $CNT_{v_{n,r}} > 1$. Next, Cache Manager sorts these chunks in descending order according to their combined utilities and selects the cached chunks sequentially until there is no caching space left in the preferred space.

C. Network Monitor

Network Monitor is responsible for monitoring the status of the backhaul network between Intelligent Edge and Origin Server, including 1) *Request List*: It records all active video request sessions that have been sent to the origin server. The length of Request List K represents the number of active video sessions on the backhaul network. 2) *Estimated Throughput*: The throughput from the edge to the origin server of past k time-slots $\vec{b_k}$ is recorded. We use a one-dimensional convolutional neural network to predict the current available edge-server bandwidth B. 3) RTT_{eo}: the round-trip-time between the edge and the origin server. This information is provided to the ABR agent to assist decision-making.

D. Edge-based ABR Agent

VISCA integrates two decisions into its ABR algorithm for QoE optimization: 1) the response chunk's resolution r, and 2) the way to get the chunk ω. Upon the arrival of a new chunk request, ABR Agent decides on which way to respond and selects the resolution. As shown in Fig. 5, there are three potential ways of ABR Agent to get the corresponding chunk: a) fetch from the local edge cache; b) fetch from the origin server; c) reconstruct a high-resolution chunk by super-resolution. Four key factors are affecting the decision made by ABR Agent: a) the edge cache status; b) the backhaul network bandwidth; c) the computing occupation status of the edge; d) the buffer occupancy level of the client. As illustrated in Fig. 5, ABR Agent takes into account four parts of information to make a response decision (ω, r) with regards to maximizing users' QoE.

1) Problem Formulation: t_0 is the time when the user sends the request, t_1 is the time when the edge receives the request, t_2 is the super-resolution process start time, and v_n is the n_{th} chunk of video v. When receiving a request of v_n at time t_1, ABR Agent parses the user information from the HTTP request, including the timestamp t_0 when the user sends the request, the current buffering level b_{t_0}, the quality of last chunk $q(v_{n-1,r})$, and the round-trip-time between edge and client RTT_{ce}. Cache information is obtained from Cache Manager, and utilization information of computing power is provided by Computing Scheduler. Network Monitor measures the available bandwidth from the edge to the origin server. It is worth mentioning that the bandwidth from edge-to-client is sufficient compared to the backhaul links between edges and origin servers. Thus, the extra delay caused by transmitting the video chunk from the edge to clients is negligible. ABR Agent aims to get the maximum QoE under the following restrictions:

- For cache status, we define a binary video cache status variable $a(v_{n,r})$. $a(v_{n,r}) = 1$ denotes $v_{n,r}$ is cached at the edge and vice versa. The rebuffering time of returning a cached chunk can be written as

$$T_{LC} = max(RTT_{ce} - b_{t_0}, 0). \tag{4}$$

Algorithm 1 Edge-based Intelligent ABR algorithm (EIABR)

```
Input: $CList$, $v_n$, $R$, $F$.
Output: $(\omega, r)$.
1: Initialize $M \leftarrow \varnothing$, $CAN \leftarrow \varnothing$, $flag \leftarrow 0$
2: for $r \in R$ do
3:     $CAN \cup QoE(v_{n,(\omega=0,r)})$
4:     if $a(v_{n,r})$ = 1 then
5:         $CAN \cup QoE(v_{n,(\omega=1,r)})$ , $flag \leftarrow 1$
6:         if $z(v_{n,r})$ = 0 then
7:             $M \leftarrow M \cup v_{n,r}$
8: end for
9: $M \leftarrow M \cap F$
10: for $(r, r^{'})$ in $M$ do
11:     $CAN \cup QoE(v_{n,(\omega=2,r)})$
12: end for
13: $(\omega, r) \leftarrow \arg\max_{(\omega,r)} QoE(v_{n,(\omega,r)})$,
            $QoE(v_{n,(\omega,r)}) \in CAN$
14: if $\omega = 0$ then
15:     if not Request Management Algorithm($flag$) then
16:         $(\omega, r) \leftarrow \arg\max_{(\omega,r),\omega!=0} QoE(v_{n,(\omega,r)})$,
                $QoE(v_{n,(\omega,r)}) \in CAN$
17: return $(\omega, r)$
```

Besides, we define a binary variable $z(v_{n,r})$ to keep track of the reconstructed chunk obtained from the super-resolution process. $z(v_{n,r}) = 1$ represents a reconstructed chunk and vice-versa.

- For computing status, if ABR Agent decides to enhance the cached chunk $v_{n,r}$ to resolution $r^{'}$ and return to the client, and it will send the corresponding super-resolution command to Computing Scheduler. g is the ongoing super-resolution task to enhance $v_{n,r}$ in the SR processor. The total processing time of chunks in the waiting queue Q at time $t_1(v_{(n)})$ can be estimated as

$$\Psi(t_1(v_n)) = \sum_{v_{n,r} \in Q} \phi(r, r^{'}), \tag{5}$$

where $t_1(v_n)$ is the timestamp when the edge receives the request v_n. l_g is the remaining processing time of the super-resolution task g in progress, which is depicted by:

$$l_g = t_2(g) + \phi(g(r, r^{'}))) - t_1(v_n), \tag{6}$$

where $t_2(g)$ is the super-resolution process start time for the super-resolution task g, $\phi(g(r, r^{'}))$ is the processing time of task g which reconstructs the chunk from resolution r to resolution r'. Thus, the total time $C(v_{n,r})$ to finish the super-resolution process of $v_{n,r}$ can be deduced by:

$$C(v_{n,r}) = \Psi(t_1(v_n)) + l_g + \phi(r, r^{'}). \tag{7}$$

The rebuffering time of obtaining a reconstructed chunk from resolution r to $r^{'}$ can be written as:

$$T_{VSR} = max(C(v_{n,r}) + RTT_{ce} - b_{t_0}, 0). \tag{8}$$

- For the backhaul network status, $K(t_1)$ represents the length of Request List, and $B(t_1)$ represents the backhaul bandwidth at time t_1. The current download throughput

Algorithm 2 Request Management Algorithm

```
Input: $Pop\_video(v)$, $H$, $K$, $flag$
1: if $flag = 0$
2:     return True
3: $Pop_{max} \leftarrow max(Pop\_video(v), Pop_{max})$
4: if $K + 1 < \frac{H}{2}$ then
5:     return True
6: else if $Pop\_video(v) > (1 + log_2(\frac{K}{H})) \times Pop_{max}$ then
7:     return True
8: else
9:     return False
```

of chunk $v_{n,r}$ can be approximated as $B(t_1)/K$ [43]. The time delay d of fetching a new chunk $v_{n,r}$ from the origin server can be deduced by :

$$d = \frac{S(v_{n,r}) \times (K(t_1) + 1)}{B(t_1)} + RTT_{eo}, \tag{9}$$

where $S(v_{n,r})$ denotes the chunk size. The rebuffering time of fetching from origin server is

$$T_{OS} = max(d + RTT_{ce} - b_{t_0}, 0). \tag{10}$$

QoE definition: We refer to the QoE model defined in Pensieve [38]. We regard perceived video quality, rebuffering time, and smoothness as the critical elements in our QoE calculation, which is defined as:

$$QoE(v_{n,(\omega,r)}) = q(v_{n,r}) - \mu T_n - \lambda \left| q(v_{n,r}) - q(v_{n-1,r}) \right|, \tag{11}$$

where $q(\cdot)$ denotes the VMAF score quantified the video quality perceived by the client, T_n represents the rebuffering caused by downloading this chunk, and the last term penalizes the changes in video quality to favor smoothness. Coefficient μ and λ are weight factors.

The goal of VISCA is to maximize the overall users' QoE. According to the system model, the problem can be expressed as follows:

$$\textbf{P1}: \max_{(\omega,r)} \sum_{n=1}^{N} QoE(v_{n,(\omega,r)}) \quad \textbf{subject to } \text{Eq. (4)-(10)}. \tag{12}$$

The optimization provides the following as output: 1) resolution decision r, and 2) the way ω to get the chunk. When $\omega = 0$, the edge requests $v_{n,r}$ from the original server and response to the user. When $\omega = 1$, the edge sends the cached chunk $v_{n,r}$ to the user. When $\omega = 2$, the edge returns the chunk $v_{n,r}$ enhanced by super-resolution.

2) Edge-based ABR Algorithm: Considering the system goal of maximizing users' QoE, we propose an Edge-based Intelligent ABR algorithm (EIABR), presented in Algorithm 1. ABR Agent is responsible for making the online response decision (ω, r) for request v_n. $CList$ is the local cache list which includes $Clist_{pre}$ and $Clist_{pb}$, R is the set of standard resolutions, and F is the legal resolution pairs for super-resolution processing. We initialize M to record the cached resolution versions of v_n, CAN to record the estimated QoE of $v_{n,(\omega,r)}$ and $flag$ to denote whether there are cached

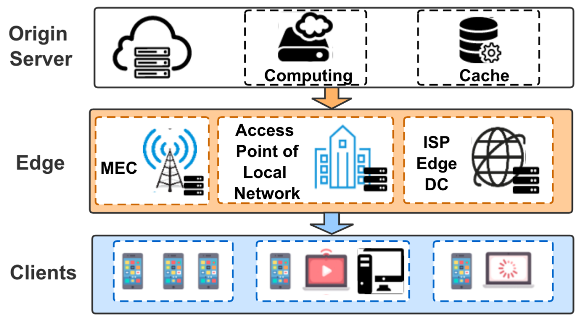
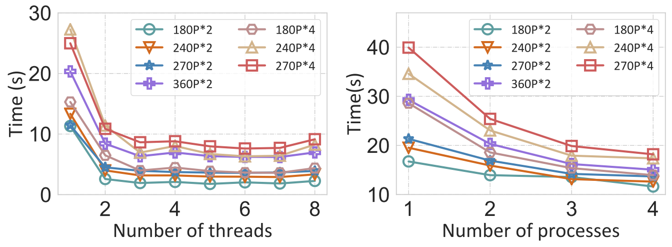


Fig. 6: Streaming architecture with edges.

resolution versions of v_n. As lines 2-8 in Algorithm 1 show, upon a video request v_n arrival, ABR Agent first calculates QoE of fetching available resolution versions from the origin server. Meanwhile, it extracts all the cached resolution versions and calculates the corresponding QoE of directly returning the cached chunk. Then ABR Agent checks whether those cached chunks are suitable for the super-resolution task by comparing against F. Lines 10-12 calculate QoE of obtaining video chunks from super-resolution processing. Next, as lines 14-17 show, ABR Agent chooses the method that maximizes users' QoE. If the choice is fetching from the origin server, we check whether the new request can be added to the Request List according to the Request Management Algorithm. If the fetch request meets the Request Management Algorithm constraints, a new request is then sent to the origin server. Otherwise, ABR Agent chooses the (ω, r) with the largest QoE among the response methods that can be completed at the edge.

Request Management Algorithm checks whether the request of fetching from the origin server can be sent. Under limited backhaul bandwidth, sending a new request to the origin server will increase the access latency for the request in the *Request List*, resulting in the drop of users' QoE. Therefore, we divide the requests sent to the origin server into two categories: 1) cache-miss request ($flag = 0$): The edge does not cache any version of the requested video chunk and can only fetch video chunk from the origin server; 2) cache-hit request ($flag = 1$): There exist cached video chunks that can be enhanced or directly returned to users, but higher-resolution videos can be obtained from the origin server which achieves higher QoE. All the cache-miss requests must be sent to the origin server, while not all cache-hit requests should be sent, which compete for the backhaul bandwidth with cache-miss requests. Thus, we set a threshold H to limit the impact of new coming requests to the active downloading sessions. We assume the available backhaul bandwidth is B, the total number of requests in the *Request List* before adding a new request is K. All the requests in the *Request List* share B equally [43]. The download throughput f of $v_{n,r}$ in the *Request List* can be approximated as $\frac{B}{K}$. If x new requests are added in the *Request List* the new throughput can be written as $f^{'} = \frac{B}{K+x}$. For a request $v_{n,r}$ in the *Request List*, the original download time $\delta(v_{n,r})$ is $\frac{S(v_{n,r})}{f}$, where $S(v_{n,r})$ is the chunk size. The difference between the new download time and the original download time can be expressed as:



(a) Multiple threads execution (b) Multiple processes execution

Fig. 7: Comparison of GPU computing efficiency.

$$\delta^{'}(v_{n,r}) - \delta(v_{n,r}) = \frac{S(v_{n,r}) \times x}{B}. \tag{13}$$

We set the maximum additional delay as ε. All newly added request should satisfy $\delta^{'}(v_{n,r}) - \delta(v_{n,r}) < \varepsilon$, so the maximum number of new requests x can be derived as follows:

$$x = \min_{v_{n,r} \in RList} \frac{\varepsilon \times B}{S(v_{n,r})}. \tag{14}$$

Then if new requests are added, the maximum capacity of *Request List* is $H = K + x$. Request Management Algorithm considers the popularity of the requested video chunk and the upper limit of the *Request List* length to restrict the addition of new requests. Lines 3-9 in Algorithm 2 detail this process. If the *Request List* length is less than half of the threshold, we add the new request to the *Request List*. Otherwise, a fetch request can be sent to the origin server if and only if

$$Pop\_video(v) > \theta \times Pop_{max}, \theta \in (0, 1], \tag{15}$$

where $Pop\_video(v)$ is the the total number of requests of a video v. $\theta = log_2(\frac{K}{H})$ is derived from the intuition that when the number of requests approaches the upper limit, we should pick fetch requests more carefully. We maintain the maximum number of requests overall requested video in the Request List, denoted by Pop_{max}, which is updated upon the arrival of each origin server fetch request.

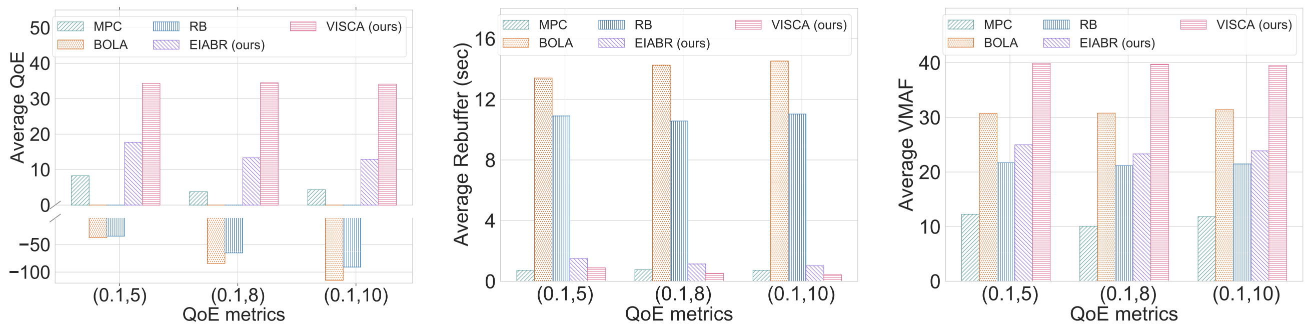
VI. Implementation

A. Practical deployment of VISCA

VISCA is implemented on top of the current HTTP adaptive streaming. Fig. 6 illustrates the proposed streaming system with edge support. To watch videos, the clients download the chunks sequentially with HTTP requests. An edge handles the DASH requests from clients based on network conditions and edge resource utilization. As Fig. 6 shows, video content providers can deploy the Intelligent Edge at the mobile edge computing (MEC) node, the access point of the local network, or the edge data center of ISPs. In this way, the content provider can improve the QoE for its users and reduce the required bandwidth for video delivery.

B. Experimental Setup

We implement the prototype of VISCA based on Nginx [18], uWSGI [19], and Django [20] architecture, which is a common deployment in the production environments. The



(a) QoE comparison (b) Rebuffering comparison (c) Quality comparison

Fig. 8: Performance of different video streaming solutions

origin server in Fig. 1 is Nginx-based, which contains 70 two-minute videos, occupying 20GB. Each video is divided into 4-second chunks, which are encoded into nine discrete bitrates from 300Kbps to 4800Kbps.We select the most appropriate bitrate for each resolution, which gives a good level of video quality with a low bandwidth cost. Intelligent Edge is equipped with two RTX 2080Ti GPUs and two CPUs (Intel(R) Xeon(R) Silver 4210). We implement four modules of Intelligent Edge using Django Framework. The video quality weight α and the video popularity weight β of cache-utility are configured to 0.2 and 0.8, respectively. The cache size is set to 400MB. Our test's legal reconstruction base resolutions include 180p, 240p, 270p, and 360p, which can be super-resolved by three upsample scales, 2, 3, and 4. The super-resolution model is implemented by Pytorch [44] based on FRVSR [28] and trained on the origin server. We use the Vimeo dataset [45] as the training dataset for our general model. We only train specific super-resolution models for the top 5 popular videos among our test dataset. We create 100 virtual DASH players to simulate the clients. The distribution of video requests follows Zipf's Law with $\alpha = 1.86$ according to video popularity analysis based on video data we collected from Douyin.

C. Super-resolution Inference Process

In order to meet the real-time transmission requirements and cost-effectively apply super-resolution, we have conducted several experiments with four GeForce RTX 2080 Ti GPUs to investigate the performance of super-resolution. To explore the maximal processing capacity of a single GPU, the experiments were run with different numbers of threads. GPU resource utilization improved with the increase in the number of threads. However, excessive parallelization damages overall efficiency. Fig 7a shows that the average processing speed of 2-threads and 3-threads super-resolution is 2.6 and 3.6 times faster than that of a single thread, respectively, and the reduction in the processing time of more than 3-threads levels off. We have also tested the performance of super-resolution tasks using multiple parallel processes assigned to different GPUs. However, the cooperation between GPUs involves disk reading and writing, and process creation and termination are associated with higher costs than that of the thread. The results of using multiple processes are worse than using multiple threads. As shown in Fig. 7b, multi-process

TABLE II: Video processing speed.

Method	SR Speed (FPS)	Total Processing Time (sec)
FRVSR	6.27	16.205
FRVSR-parallel	24.12	5.045
LightSR	40.49	3.436

tasks took much longer than multi-threads with one GPU. Therefore, a parallel multi-threading super-resolution method was adopted to optimize the overall GPU utilization efficiency and reduce the latency of video super-resolution.

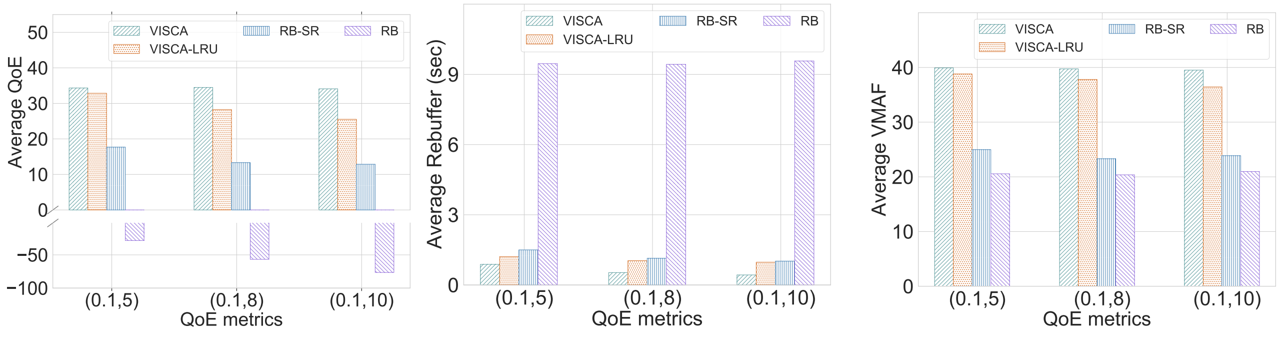
D. Comparison of Super-resolution Models

We have conducted experiments to investigate the performance of different super-resolution models. We have implemented FRVSR without parallelism, FRVSR-parallel with 3-thread parallelism, and LightSR with 3-thread parallelism. The second column in Table II represents the super-resolution speed to enhance a 180p video to 720p with an RTX 2080Ti GPU. The LightSR achieves the fastest super-resolution inference speed as 40.49 FPS. The third column in the table II reports the total processing time of a four-second video chunk at 24fps with three super-resolution models. The processing time consists of three parts: decoding, video super-resolution, and encoding time. SR processor decodes and re-encodes video chunk in H.264 with FFmpeg using the fastest option [46]. Through our parallelism processing, the time spent in the whole super-resolution task reduces from 16.205s to 5.045s. Moreover, the processing time of LightSR can be reduced by 31.89% compared to FRVSR-parallel.

E. Network Traces

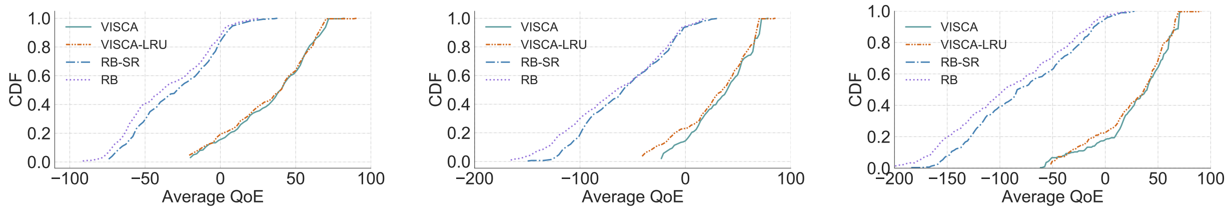
Real network traces: To simulate the dynamic changes of the backhaul network in a realistic network, we used the Linux Traffic Control tool [47] to control the sending rate of the original server with 100 throughput traces from FCC [48]. To highlight the advantages of VISCA in the case of excessive bandwidth competition at the same access point resulting in insufficient backhaul bandwidth, we only used traces with average throughput between 1Mbps and 40Mbps. Besides, the RTT between the edge and the origin server was set to 200ms.

Synthetic traces: We conducted a small set of experiments on synthetic network conditions to stress test VISCA under



(a) QoE comparison (b) Rebuffering comparison (c) Quality comparison

Fig. 9: Component-wise comparison.



(a) QoE with (0.1, 5) (b) QoE with (0.1, 8) (c) QoE with (0.1, 10)

Fig. 10: QoE values comparison under different QoE metrics.

poor (e.g., 4 Mbps) versus good (e.g., 60 Mbps) network conditions.

Metrics: To test the performance of VISCA under different users' QoE preferences, we considered three sets of QoE coefficients (λ, μ) to indicate different tolerance for latency, namely (0.1, 5), (0.1, 8), (0.1, 10).

VII. Experimental Testing and Results

The evaluation employed three machines: one origin server, one edge server, and one client server. The origin server utilizes the Nginx Web Server to serve video requests from the edge server. Django and Nginx are used to implement the Intelligent Edge to serve video requests. At the client server, we ran 100 DASH virtual players as different users to download videos. We set each streaming experiment under the same network trace with 70 available video clips. We conducted each streaming experiment for five epochs over three types of users' QoE preferences. For each epoch, we looped the trace until videos were completely downloaded for all DASH players.

A. Comparison with Existing Video Streaming Solutions

We compare the performance of VISCA with existing video streaming solutions. Also, we investigate the performance of our proposed ABR algorithm (EIABR), which does not use SR Processor. The existing video solutions deploy the corresponding ABR algorithm on the client-side, and there is no super-resolution module at the edge, and the cache replacement strategy is LRU [49].

- *MPC* [10], which uses buffer occupancy observations and throughput predictions to select the bitrate, which maximizes a given QoE metric over a horizon of 5 future chunks.
- *BOLA* [8], which uses Lyapunov optimization to select bitrates solely considering buffer occupancy observations.
- *Rate-Based (RB)* [6], which selects the highest available bitrate that is below the predicted throughput.

We first evaluate the performance of VISCA and EIABR compared to the existing streaming solutions under three QoE objectives. We choose network traces whose average throughputs are scarce to simulate poor network conditions. Fig. 8a shows the average QoE that each ABR algorithm achieves. As shown in the figure, our proposed EIABR algorithm outperforms other algorithms on all three QoE coefficients, with an average improvement of 17.0, 78.2, and 93.6 over MPC, RB, and BOLA, respectively. With super-resolution and our combined-utility based strategy, VISCA achieves a greater QoE than EIABR, with an average improvement of 19.7.

Fig. 8b and Fig. 8c show the average rebuffering time and the average video quality of each ABR algorithm under three QoE objectives. As we can see, BOLA attempts to fetch high-quality video, which results in a long rebuffering time under limited bandwidth. RB, MPC, and EIABR can perceive network condition changes, which leads to a lower quality of video and a shorter rebuffering time. EIABR achieves higher video quality than RB and MPC, which indicates that our proposed edge-based ABR algorithm can better adapt to network changes. On top of EIABR, VISCA integrates super-resolution and combined-utility strategy, which yields the highest video quality, with an average improvement of 28.2%, 85.2%, 251.2% from BOLA, RB, and MPC, respectively. VISCA also achieves the shortest rebuffering time, with an average reduction of 16.1%, 94.3%, 95.6% from MPC, RB, and BOLA, respectively. This result shows that VISCA can well-adapt to different network changes and different QoE objectives, making it efficient for video streaming in real-world applications.

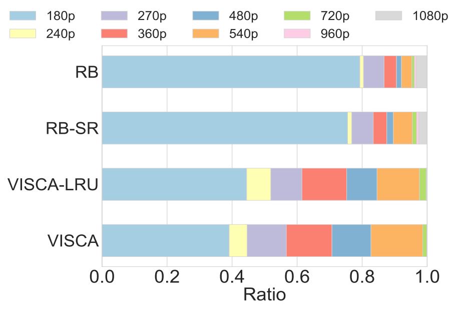


Fig. 11: Video quality distribution

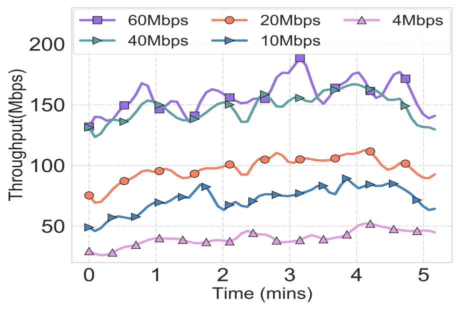


Fig. 12: Throughput variation

B. Component-wise Analysis

The key designs of VISCA include the edge-based intelligent ABR strategy (EIABR), the combined-utility based cache strategy, and super-resolution. In order to evaluate the contribution of each design component to the QoE improvement, we compare the following four schemes:

- *RB*, which deploys the rate-based ABR algorithm on the client-side, employs LRU caching strategy at the edge and does not use super-resolution.
- *RB-SR*, which deploys the rate-based ABR algorithm on the client-side, applies LRU caching strategy and utilizes super-resolution at the edge.
- *VISCA-LRU*, which deploys EIABR and LRU caching strategy and utilizes super-resolution at the edge.
- *VISCA*, our proposed framework, which integrates the combined-utility based cache strategy, the edge-based intelligent ABR, and super-resolution.

1) QoE Improvement: We investigate the effects of all schemes on QoE. Fig. 9a shows the average QoE that each scheme achieves over our entire test corpus under three QoE objectives. Fig. 10 provides more detailed results in the form of full CDFs under each QoE metric. We observe that the edge-based intelligent ABR strategy, the combined-utility based cache strategy, and super-resolution all have positive impacts on the improvement of QoE. First, we study how super-resolution can help improve the video transmission quality. After video quality enhancement at the edge, the QoE of RB-SR increased by 9.8 on average compared to RB. Second, with EIABR, VISCA-LRU can better use network bandwidth, edge cache, and computing resources, thus achieving higher QoE. The huge gap between VISCA-LRU and RB-SR widens from 71.3 to 102.0, reflecting that the edge-based ABR algorithm delivers a significant QoE gain. Finally, with a caching strategy specially designed for the edge with the super-resolution module, the QoE of VISCA exceeds the QoE of VISCA-LRU by 4.5%-33.6%. Overall, VISCA achieves the highest QoE across all QoE metrics, which indicates that VISCA can improve the quality of video services in the case of insufficient network resources.

2) Fine-grained Analysis: To better understand the QoE gains obtained by VISCA, we analyzed the rebuffering and video quality among all considered schemes.

Rebuffering. Fig. 9b compares VISCA with other schemes in terms of average rebuffering time. As shown in the figure, RB yields the highest rebuffering time, as the client-based ABR algorithm can not adapt well to the perceived throughput variation caused by edge cache hits. Because some requests can be resolved at the edge by super-resolution, RB-SR reduces rebuffering time by at least 10.8% compared to RB. VISCA and VISCA-LRU achieve lower rebuffering with edge-based ABR algorithm with an average reduction of 93.5% and 88.7% from RB-SR, respectively, and the reduction of 94.3% and 90.1% from RB, respectively. Fig.9b shows that the processing time of super-resolution does not increase the rebuffering time because ABR Agent can adaptively choose whether to respond to the user through super-resolution on the user's buffering level. The rebuffering time of VISCA is on average 44 % lower than that of VISCA-LRU, which shows that our combined-utility based cache strategy copes better with super-resolution.

Quality. Fig. 9c shows the average quality under each scheme. The average quality of RB-SR and RB is about half of VISCA quality due to the poor performance of client-side based ABR. The average VMAF of VISCA is 5.5% higher than that of VISCA-LRU, which is attributed to the combined-utility based strategy that successfully caches video content with high potential values. Fig. 11 shows the distribution of video resolutions over four schemes under the QoE objective (0.1,10). With the assistance of EIABR, VISCA-LRU obtains more high-resolution video chunks than RB and RB-SR under limited bandwidth. With our proposed cache strategy, the cache space can be fully utilized. Therefore, the proportion of high-resolution video chunks of VISCA is higher than that of VISCA-LRU.

C. Different Network Conditions

VISCA is specifically designed to work well in case of insufficient bandwidth resources. We compare VISCA performance under six synthetic network scenarios—4Mbps, 10Mbps, 20Mbps, 40Mbps, and 60Mbps bandwidth of the backhaul network at the same access point. Fig. 12 shows the total users perceived throughput under these network bandwidth. VISCA sees more benefits under poorer networks as VISCA can make full use of the super-resolution technique under such a bad network condition. VISCA achieves by 2.6×, 3.7×, 4.8×, 7.2×, and 10.1× the total users' perceived throughput of the backhaul network bandwidth under 60Mbps, 40Mbps, 20Mbps, 10Mbps, and 4Mbps, respectively.

VIII. Conclusion

As video traffic surges, the network load increases. Users' QoE suffers inevitably in the case of limited backhaul bandwidth resources. This paper proposes VISCA, an edge-assisted adaptive video streaming framework with joint VIdeo Super-resolution and CAching, aiming to improve users' QoE under bad network conditions. With the employment of video super-resolution on edge, VISCA can achieve high video quality even under poor network conditions. An edge-based intelligent ABR algorithm is proposed to allocate edge resources and network resources to maximize users' QoE dynamically. Additionally, a novel cache strategy, combined-utility based cache strategy, is introduced to cache the most valuable video considering the potential benefits of super-resolution.

A comprehensive evaluation of the proposed solution is performed, which demonstrates that VISCA delivers substantial benefits. Under 4 Mbps backhaul bandwidth, VISCA provides users with 40 Mbps throughput. VISCA outperforms the existing video streaming algorithms under the real-world trace by at least 17.0 and up to 93.6 on average QoE. Due to its greater flexibility and scalability of edge resource utilization, VISCA provides an excellent service to users by improving response speed and video quality.

Acknowledgment

This work is supported by Guangdong Province Key Area RD Program under grant No. 2018B010113001, National Natural Science Foundation of China under grant No. 61972189, the project "PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications (LZC0019)" and the Shenzhen Key Lab of Software Defined Networking under grant No. ZDSYS20140509172959989. G.-M. Muntean acknowledges Science Foundation Ireland's support under grants 12/RC/2289_P2 (Insight) and 16/SP/3804 (ENABLE).

References

[1] V. Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022," *White Paper*, vol. 1, 2018.

[2] M. M. Ahamed and S. Faruque, "5g backhaul: requirements, challenges, and emerging technologies," *Broadband Communications Networks: Recent Advances and Lessons from Practice*, vol. 43, 2018.

[3] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 362–373, 2011.

[4] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs," *IEEE/ACM Trans. on Networking*, vol. 21, no. 6, pp. 2001–2014, 2013.

[5] T. Stockhammer, "Dynamic adaptive streaming over http –: Standards and design principles," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, 2011, p. 133–144.

[6] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, 2012, pp. 97–108.

[7] H. Yuan, X. Hu, J. Hou, X. Wei, and S. Kwong, "An ensemble rate adaptation framework for dynamic adaptive streaming over http," *IEEE Transactions on Broadcasting*, vol. 66, no. 2, pp. 251–263, 2019.

[8] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," in *The 35th IEEE Int. Conference on Computer Communications (INFOCOM)*. IEEE, 2016, pp. 1–9.

[9] W. Huang, Y. Zhou, X. Xie, D. Wu, M. Chen, and E. Ngai, "Buffer state is enough: Simplifying the design of qoe-aware http adaptive video streaming," *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 590–601, 2018.

[10] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 325–338.

[11] L. Yu, T. Tillo, and J. Xiao, "Qoe-driven dynamic adaptive video streaming strategy with future information," *IEEE Transactions on Broadcasting*, vol. 63, no. 3, pp. 523–534, 2017.

[12] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han, "Neural adaptive content-aware internet video delivery," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018, pp. 645–661.

[13] Y. Z. Yinjie Zhang, Y. T. Yi Wu, P. Z. Kaigui Bian, and H. T. Lingyang Song, "Improving quality of experience by adaptive video streaming with super-resolution," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020.

[14] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[15] W. Shi, Q. Li, C. Wang, G. Shen, W. Li, Y. Wu, and Y. Jiang, "Leap: learning-based smart edge with caching and prefetching for adaptive video streaming," in *Proceedings of the International Symposium on Quality of Service*, 2019, pp. 1–10.

[16] X. Ma, Q. Li, J. Chai, X. Xiao, S.-t. Xia, and Y. Jiang, "Steward: smart edge based joint qoe optimization for adaptive video streaming," in *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2019, pp. 31–36.

[17] A. Bentaleb, A. C. Begen, and R. Zimmermann, "Qoe-aware bandwidth broker for http adaptive streaming flows in an sdn-enabled hfc network," *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 575–589, 2018.

[18] "Nginx," http://nginx.org/, 2019.

[19] "uwsgi documents," https://uwsgi-docs.readthedocs.io/en/latest/tutorials/, 2019.

[20] "Django," https://www.djangoproject.com/, 2020.

[21] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*. Springer, 2014, pp. 184–199.

[22] Y. Guo, J. Chen, J. Wang, Q. Chen, J. Cao, Z. Deng, Y. Xu, and M. Tan, "Closed-loop matters: Dual regression networks for single image super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5407–5416.

[23] Z. Jiang, H. Zhu, Y. Lu, G. Ju, and A. Men, "Lightweight super-resolution using deep neural learning," *IEEE Transactions on Broadcasting*, vol. 66, no. 4, pp. 814–823, 2020.

[24] J. Caballero, C. Ledig, A. Aitken, A. Acosta, J. Totz, Z. Wang, and W. Shi, "Real-time video super-resolution with spatio-temporal networks and motion compensation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4778–4787.

[25] Y. Jo, S. Wug Oh, J. Kang, and S. Joo Kim, "Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3224–3232.

[26] 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, 2015.

[27] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1. IEEE, 2004, pp. I–I.

[28] M. S. Sajjadi, R. Vemulapalli, and M. Brown, "Frame-recurrent video super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6626–6634.

[29] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 136–144.

[30] L. Tang, Q. Huang, A. Puntambekar, Y. Vigfusson, W. Lloyd, and K. Li, "Popularity prediction of facebook videos for higher quality streaming," in *USENIX Annual Technical Conf. (USENIX ATC)*, 2017, pp. 111–123.

[31] S. S. Tanzil, W. Hoiles, and V. Krishnamurthy, "Adaptive scheme for caching youtube content in a cellular network: Machine learning approach," *Ieee Access*, vol. 5, pp. 5870–5881, 2017.

[32] H. Pang, J. Liu, X. Fan, and L. Sun, "Toward smart and cooperative edge caching for 5g networks: A deep learning based approach," in *IEEE/ACM Intl. Symp. on Quality of Service (IWQoS)*, 2018, pp. 1–6.

[33] Z. Akhtar, Y. Li, R. Govindan, E. Halepovic, S. Hao, Y. Liu, and S. Sen, "Avic: a cache for adaptive bitrate video," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, 2019, pp. 305–317.

[34] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "Toward a practical perceptual video quality metric," *The Netflix Tech Blog*, vol. 6, p. 2, 2016.

[35] Q. Huynh-Thu and M. Ghanbari, "Scope of validity of psnr in image/video quality assessment," *Electronics letters*, vol. 44, no. 13, pp. 800–801, 2008.

[36] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *Intl. Conference on Pattern Recognition*. IEEE, 2010, pp. 2366–2369.

[37] R. Rassool, "Vmaf reproducibility: Validating a perceptual practical video quality metric," in *IEEE international symposium on broadband multimedia systems and broadcasting (BMSB)*. IEEE, 2017, pp. 1–2.

[38] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 197–210.

[39] F. Jiang, Z. Liu, K. Thilakarathna, Z. Li, Y. Ji, and A. Seneviratne, "Transfetch: A viewing behavior driven video distribution framework in public transport," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*. IEEE, 2016, pp. 147–155.

[40] H. Pinto, J. M. Almeida, and M. A. Gonçalves, "Using early view patterns to predict the popularity of youtube videos," in *Proc. ACM Intl. Conference on Web Search and Data Mining*, 2013, pp. 365–374.

[41] H. S. Goian, O. Y. Al-Jarrah, S. Muhaidat, Y. Al-Hammadi, P. Yoo, and M. Dianati, "Popularity-based video caching techniques for cache-enabled networks: A survey," *IEEE Access*, vol. 7, pp. 27 699–27 719, 2019.

[42] F. Wang, F. Wang, J. Liu, R. Shea, and L. Sun, "Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2499–2508.

[43] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis, "What happens when http adaptive streaming players compete for bandwidth?" in *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video*, 2012, pp. 9–14.

[44] "Pytorch documents," https://pytorch.org/, 2019.

[45] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *International Journal of Computer Vision (IJCV)*, vol. 127, no. 8, pp. 1106–1125, 2019.

[46] "H.264 video encoding guide," https://trac.ffmpeg.org/wiki/Encode/H.264/, 2020.

[47] B. Hubert *et al.*, "Linux advanced routing & traffic control howto," *Netherlabs BV*, vol. 1, 2002.

[48] F. C. Commission., "Raw data - measuring broadband america." https://www.fcc.gov/reports-research/reports/, 2016.

[49] A. Dan and D. Towsley, "An approximate analysis of the lru and fifo buffer replacement schemes," in *Proceedings of the 1990 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, 1990, pp. 143–152.

Aoyang Zhang received the B.Eng. degree in Communication Engineering from Beijing Institute of Technology in 2018. She is currently a Master student with Tsinghua-Berkeley Shenzhen Institute. She majors in Data Science and Information Technology. Her research interests include adaptive multimedia and multimedia streaming, resource allocation and user quality of experience.

Qing Li (S'10-M'14) received the B.S. degree (2008) from Dalian University of Technology, Dalian, China, the Ph.D. degree (2013) from Tsinghua University, Beijing, China. He is currently a Research Associate Professor with Southern University of Science and Technology, and also with Peng Cheng Laboratory, Shenzhen, China. His research interests include reliable and scalable routing of the Internet, in-network caching/computing, intelligent self-running network, edge computing, etc.

Ying Chen received the B.Eng. (Hons) degree in Engineering Science from The University of Auckland in 2016. She is currently a Master student with Tsinghua-Berkeley Shenzhen Institute. She majors in Data Science and Information Technology. Her research interests include adaptive multimedia and multimedia streaming, resource allocation and user quality of experience.

Xiaoteng Ma is a Ph.D. student with Tsinghua-Berkeley Shenzhen Institute. He received his B.Eng. degree in Electrical Engineering from Huazhong University of Science and Technology in 2017. He majors in Data Science and Information Technology. His research interests include edge-assisted multimedia delivery, adaptive multimedia and multimedia streaming, and resource allocation on hybrid cloud-edge-client network.

Longhao Zou (S'12-M'19) received the B.Eng and Ph.D degrees from Beijing University of Posts and Telecommunications (BUPT), Beijing, China and Dublin City University (DCU), Ireland in 2011 and 2016, respectively. He was a postdoctoral researcher with the EU Horizon 2020 NEWTON Project at DCU. Now he is Research Assistant Professor with Southern University of Science and Technology, and also with Peng Cheng Laboratory, Shenzhen, China. His research interests include mobile and wireless communications, adaptive multimedia and multimedia streaming, resource allocation and user quality of experience.

Yong Jiang is currently a Full Professor with Tsinghua Shenzhen International Graduate School. He received his B.S. degree and Ph.D. degree both from Tsinghua University, respectively in 1998 and 2002. He mainly focuses on the future Internet, edge computing, multimedia transmission, AI for networks, etc.

Zhimin Xu is currently an algorithm engineer in Multimedia Lab in Beijing Bytedance Technology Co., Ltd., Beijing, China. He received the B.S. degree in network engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2016, the M.S. degree in Wangxuan Institute of Computer Technology, Peking University, Beijing, China, in 2019. His research interests include video streaming, multimedia communication, QoE, 360-degree video, playback experience optimization. He is the Runner-up of the IEEE International Conference on Multimedia and Expo (ICME) DASH-IF Grand Challenge Award on Dynamic Adaptive Streaming over HTTP (DASH) both in 2017 and 2018.

Gabriel-Miro Muntean (M'04, SM'17) is a Professor with the School of Electronic Engineering, Dublin City University (DCU), Ireland, and Co-Director of the DCU Performance Engineering Laboratory. Prof. Muntean was awarded the PhD degree by DCU for research on adaptive multimedia delivery in 2004. He has published over 400 books, chapters and papers in top-level international venues. His research interests include quality, performance, and energy saving issues related to rich media delivery, technology-enhanced learning, and other data communications over heterogeneous networks. Prof. Muntean is an Associate Editor of the IEEE Transactions on Broadcasting, the Multimedia Communications Area Editor of the IEEE Communications Surveys and Tutorials, and chair and reviewer for top international journals and conferences.