A Content-Aware Bitrate Selection Method Using Multi-Step Prediction for 360-Degree Video Streaming

GAO Nianzhen[1], YU Yifang[2], HUA Xinhai[2],
FENG Fangzheng[1], JIANG Tao[1]

(1. Huazhong University of Science and Technology, Wuhan 430074, China;
2. ZTE Corporation, Shenzhen 518057, China)

DOI: 10.12142/ZTECOM.202204012

https: //kns.cnki.net/kcms/detail/34.1294.TN.20221026.1118.002.html, published online October 26, 2022

Manuscript received: 2021-12-12

Abstract: A content-aware multi-step prediction control (CAMPC) algorithm is proposed to determine the bitrate of 360-degree videos, aiming to enhance the quality of experience (QoE) of users and reduce the cost of video content providers (VCP). The CAMPC algorithm first employs a neural network to generate the content richness and combines it with the current field of view (FOV) to accurately predict the probability distribution of tiles being viewed. Then, for the tiles in the predicted viewport which directly affect QoE, the CAMPC algorithm utilizes a multi-step prediction for future system states, and accordingly selects the bitrates of multiple subsequent steps, instead of an instantaneous state. Meanwhile, it controls the buffer occupancy to eliminate the impact of prediction errors. We implement CAMPC on players by building a 360-degree video streaming platform and evaluating other advanced adaptive bitrate (ABR) rules through the real network. Experimental results show that CAMPC can save 83.5% of bandwidth resources compared with the scheme that completely transmits the tiles outside the viewport with the Dynamic Adaptive Streaming over HTTP (DASH) protocol. Besides, the proposed method can improve the system utility by 62.7% and 27.6% compared with the DASH official and viewport-based rules, respectively.

Keywords: DASH; content-aware FOV prediction; bitrate adaptation; multi-step prediction; generalized predictive control

Citation (IEEE Format): N. Z. Gao, Y. F. Yu, X. H. Hua, et al., "A content-aware bitrate selection method using multi-step prediction for 360-degree video streaming," *ZTE Communications*, vol. 20, no. 4, pp. 96 – 109, Dec. 2022. doi: 10.12142/ZTECOM.202204012.

1 Introduction

With the rapid development of 5G technologies, the immersive viewing experience led by Virtual Reality (VR) is becoming increasingly popular. The 360-degree video, which is one of the most critical portions of VR applications, has attracted a good deal of attention on some commercial streaming platforms, such as Youtube and Bilibili. Although the 360-degree video brings a brand-new experience to users, it confronts new challenges as well. Compared with conventional 2D video streaming, the delivery of 360-degree video has much more bandwidth requirements due to its panorama feature.

Driven by the characteristic of the user field of view (FOV), researchers are exploring a solution that spatially divides a 360-degree video into small parts called tiles and transmits them in different video qualities to cope with the large consumption of bandwidth. In addition, benefitting from the Dynamic Adaptive Streaming over HTTP (DASH) Protocol, adaptive bitrate (ABR) algorithms have made an enormous contribution to video streaming, especially over dynamic wireless network conditions. By pre-coding the video into multi-bitrate, the client may request video segments of different qualities to meet the challenge of network fluctuations.

Therefore, tile-based hyper text transfer protocol (HTTP) adaptive streaming is a promising approach to achieving a better quality of experience (QoE) in a 360-degree video streaming system. The HTTP server usually crops the panoramic video into multiple tiles spatially, and then slices and encodes each tile into multi-bitrate segments. The client requests the most appropriate bitrate version of each tile based on his viewport and current network status, decodes these tiles, and then renders them into a 360-degree video for playback. In general, the tile that overlaps viewports is delivered in high quality, while other tiles outside the FOV are delivered in lower quality. Due to the human visual characteristics, the user can only see the FOV areas, so a significant reduction in the video bitrate outside the viewport will not affect the users' experience; on the contrary, it can save bandwidth and transmission costs, and avoid net-

This work was supported in part by ZTE Corporation under Grant No. 2021420118000065.

work congestion in the case of multiple users.

In fact, due to the randomness of the user's viewing angle and wireless network bandwidth, the low prediction accuracy will lead to an inappropriate bitrate version selected by the tiles in the viewport. To handle these prediction errors, the QoE is guaranteed by delivering tiles around the prediction viewport at high quality and keeping the buffer at a reasonable range without stopping the playback waiting buffer. In order to reasonably allocate the wireless network bandwidth resources, it is necessary to estimate the future viewport distribution and network state, and determine the bitrate version of the prefetched segments that match the overall network capacity.

Specifically, we propose a bandwidth resource scheduling algorithm content-aware multi-step predictive control (CAMPC) for 360-degree video streaming, which uses an online predictor to obtain throughput estimation. The content perception score obtained by the offline machine learning method and online user viewport trace is used to predict the probability distribution of future user viewport locations. For tiles with high viewport probability, the change of buffer occupation in the next period is predicted based on the throughput estimation. The bitrate decision is made by optimizing the predicted QoE within the viewport and the future buffer occupancy prediction. The remaining total bandwidth will be distributed according to the distribution probability for tiles with low viewport probability. The main contributions are shown as follows.

• We develop a content-aware method to predict the user's viewport location. The grayscale image is obtained by a trained semantic segmentation model with each frame of the video after spatial partition as input, and the numbers of different grayscale pixels are calculated to obtain the content richness of the current frame. Since users prefer to view the frame with richer content, the probability distribution of future viewing is obtained by the weighted content richness with the current user viewport.

• We propose a multi-step predictive bitrate adaptation algorithm to generate prospective bitrate decisions for players with high probability in the future viewport, which includes predicting network throughput using the Kalman filter, predicting buffer occupation, and solving predictive control problems using the generalized predictive control method.

• We provide experimental tests by building a 360-degree video streaming platform to implement the proposed bandwidth resource scheduling algorithm and evaluate the network status algorithm through the practical network. Experimental results show that compared with the existing online bandwidth resource scheduling algorithms, the proposed algorithm can save bandwidth while ensuring the quality of user experience. Compared with the complete transmission of 360-degree videos, the bandwidth can be saved by 83.5% in the tiles out of the viewport, and the CAMPC can improve the system utility by 62.7% and 27.6% compared with low-on-latency-plus (LOLP) and DYNAMIC solutions which have been integrated to the official DASH.js player in v3.2.0 and the most straightforward viewport-based bitrate adaptation algorithm.

The rest of the paper is organized as follows. Section 2 surveys related work on a tile-based 360-degree video streaming over DASH. Section 3 presents the system structure and QoE model for evaluation. Section 4 proposes the FOV prediction algorithm combining FOV and content priority. The bandwidth prediction and the bitrate selection algorithm are in Section 5. Section 6 describes the system implementation and throughput measurement in reality besides the performance evaluation. Finally, Section 7 concludes the paper and outlines future directions.

2 Background and Motivation

The 360-degree video is constructed by camera splicing. To play a 360-degree video, the client needs to run on a custom 360-degree video player or head-mounted displays (HMDs) to render the video. Some commercial 360-degree video content providers usually employ a simple approach that streams the entire panoramic content regardless of the viewport[1], such as the widely used equirectangular projection (ERP) format, which causes considerable waste of wireless bandwidth resource, as users always pay attention to only a tiny portion of the panoramic scene in their viewports.

Inspired by these observations, several studies have abandoned traditional flat video transmission methods and begun to propose tile-based solutions that adaptively fetch only the content inside the predicted FOV or fetch the content in FOV with higher quality than the parts out of FOV to meet the demand of 360-degree video streaming systems. XIE et al.[2] leveraged a probabilistic approach to prefetch tiles countering viewport prediction errors, apparently reduced the side effects caused by wrong head movement prediction, and designed a QoE-driven viewport adaptation system. QIAN et al.[3] adopted a viewport-adaptive approach and formulated an optimization algorithm to determine the tile quality, achieving high bandwidth reduction and video quality enhancement on Long Term Evolution (LTE). SONG et al.[4] proposed a two-tier streaming architecture using scalable video coding (SVC) techniques, which included two layers, namely, the basic layer (BL) and the enhanced layer (EL). In contrast, a fast-switching strategy was proposed by generating multiple video streams with different start times for each encoded enhanced layer chunk, which can be randomly accessed at any instant to adapt to the user viewport change immediately to achieve the optimal trade-off between video quality and streaming robustness.

According to the above research, tile-based 360-degree video transmission methods have been proven to save many bandwidth resources, whereas viewport prediction and bandwidth prediction are two of the most critical factors. To a great extent, the user's FOV would be influenced by the video content. Conventional viewport prediction approaches pay atten-

tion to the past viewing behavior of many users who have watched the same or similar videos, based on the head movement trajectory in the dataset. SUN et al.[5] developed a truncated linear prediction method by which we only use past samples that are monotonically increasing or decreasing for extrapolation. EPASS360[6] studied the similarity of multi-user viewing spatial locations, looking for similarities in patterns across a wide range of data through a deep learning LSTM network. These approaches apply only to the video on demand (VOD) case because the past viewing behavior is not available for live video streaming for the first time. The Pano[7] drew researchers' attention to the content of the video. FENG et al.[8] developed a new viewport prediction scheme for live 360-degree video streaming using video content-based motion tracking and dynamic user interest modeling. QIAO et al.[9] studied human attention over the viewport of 360-degree videos and proposed a novel visual saliency model to predict fixations over 360 videos through the multi-task deep neural networks (DNN) method. YUAN et al.[10] proposed a simple yet effective rate adaptation algorithm to determine the requested bitrate for downloading the current video segment and preserved both the quality and the smoothness of tiles in FoV. WEI et al.[11] proposed a hybrid control scheme presented for segment-level continuous bitrate selection and tile-level bitrate allocation for 360-degree streaming over mobile devices to increase users' quality of experience.

On the other hand, to optimize QoE in the DASH video streaming system, the bitrates decided by the client-side ABR algorithm should meet the bandwidth requirements. Throughput-based methods often employ various mechanisms to predict the end-to-end available bandwidth, such as Exponential Weighted Moving Average (EWMA) and Support Vector Regression. The estimation accuracy of throughput will affect the allocation decision. SOBHANI et al.[12] utilized Autoregressive-Moving-Average (ARMA)[13] and Generalized Autoregressive Conditional Heteroscedastic (GARCH) in order to predict the average and the variance of bandwidth. YUAN et al.[14] proposed an ensemble rate adaptation framework for DASH, which aims to leverage the advantages of multiple methods involved in the framework to improve the QoE of users. The buffer-based algorithm, such as BOLA[15], formulated bitrate adaptation as a utility maximization problem, devised an online control algorithm, and used Lyapunov optimization techniques to minimize rebuffering and maximize video quality.

However, to achieve a fast and smooth response among multiple players of the 360-degree video at the same time, ABR algorithms should quickly adapt to sustainable changes while avoiding the bit rate jitter caused by sudden throughput changes. Existing methods are inherently unable to achieve this goal because they cannot determine whether a current change is transient or persistent with a single step of predictive information. Thus, our work uses the idea of combining content awareness with the current viewport to calculate the viewing probability of spatial video blocks and provides efficient network state quantification and prediction algorithms.

3 Proposed Framework

3.1 System Architecture

As shown in Fig. 1, the framework of the 360-degree video transmission system consists of a server and a set of video players. The server includes a preprocessing module and a sending module. The preprocessing module converts a 360-degree video from the ERP format to the Cubemap Projection (CMP) and separates it into six tiles spatially so that each tile corresponds to a cube map. Then each tile is divided into a set of temporal segments and encoded at different bitrate levels according to DASH, and the information which describes the structure of bitrate representations for each tile is stored in the media presentation description (MPD). The sending module sends the segments at a specific bitrate selected by the ABR controller of the player.

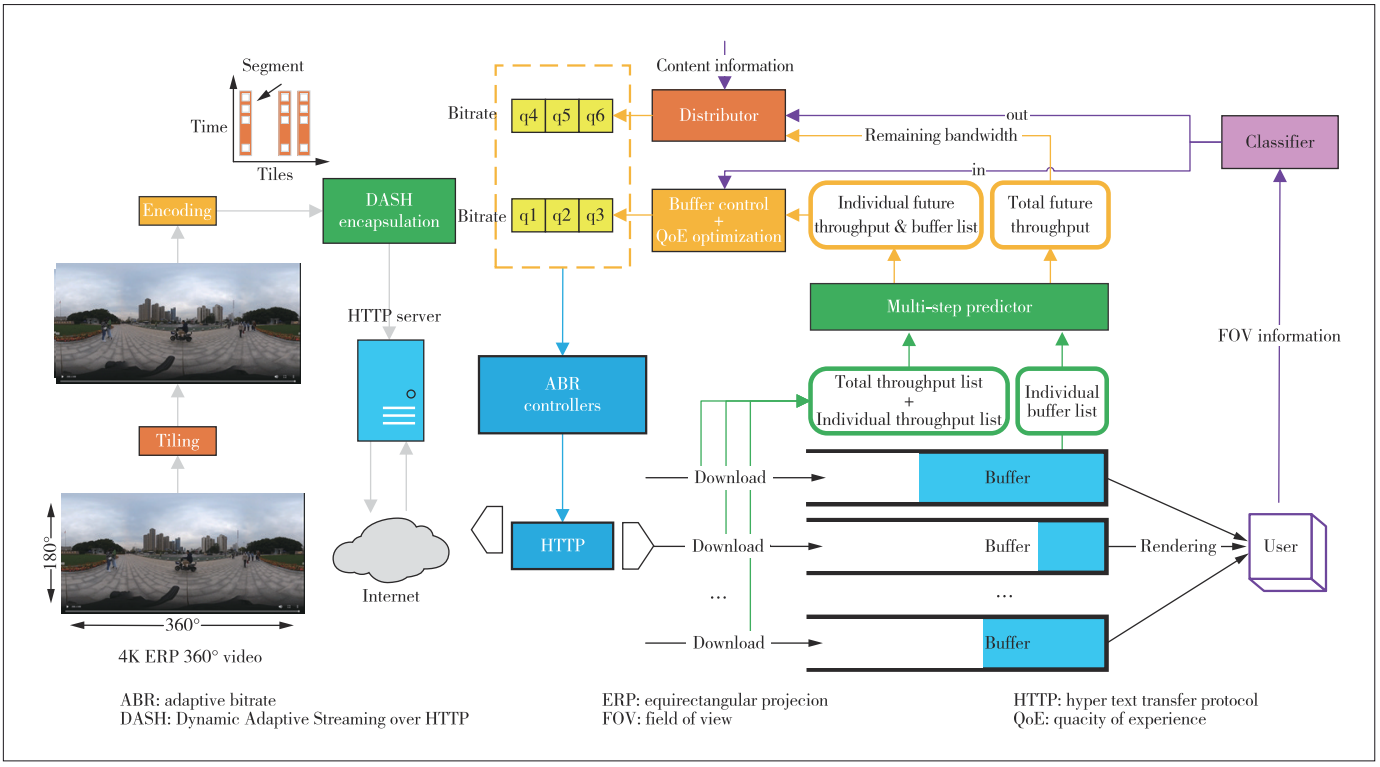
The client includes a receiving module, video players, a system monitor, multi-step predictors, and bitrate decision engines. The receiving module receives and decodes the tiles to reconstruct the 360-degree video. Players extract and display the viewport corresponding to the current viewing direction of the user. The system monitor is responsible for monitoring the viewport, network status (e.g., throughput), and player status (e.g., buffer occupancy). Multi-step predictors and bitrate decision engines assist ABR controllers to compute the bitrate level of the next segment and return it to the server. Each multi-step predictor module calculates the future throughput through the network status, and the bitrate decision engines obtain the bitrate level by optimizing the target based on the multi-step throughput and buffer occupancy prediction.

3.2 Problem Formulation

The server divides a 360-degree video into N tiles of the same size in space corresponding to a cube map, and each tile V is initialized to a DASH player. Then, each tile is sliced into $M \times K$ segments, indicating that there are M optimal bitrate versions divided into K segments in time, and each segment has the exact duration of L seconds. The system encapsulates and stores $N \times M \times K$ segments in an HTTP server for adaptive streaming.

The serial number of the tile V is represented by i, where $i \in \mathbb{Z}$. V^{in} and V^{out} indicate that the current tile is located inside and outside the viewport, respectively. For V^{in}, the quality of experience is determined by three factors: the selected bitrate version, the bitrate fluctuation range, and the rebuffering time. Spatial tiles that are not in the viewport will not affect QoE. For this reason, the value of QoE is given by:

$$\phi(V_i) = \sum_{k=1}^{K} r(V_{i,k}^{\text{in}}) - \sum_{k=1}^{K-1} \left| r(V_{i,k+1}^{\text{in}}) - r(V_{i,k}^{\text{in}}) \right| - \mu \sum_{k=1}^{K} T_{V_{i,k}^{\text{in}}}, \quad (1)$$



▲**Figure 1. Streaming video system structure**

where $r(V_{i,k}^{in})$ is the bitrate version when the player V_i starts to download chunk $V_{i,k}^{in}$, $T_{v_{i,k}}$ is the rebuffering time of chunk $V_{i,k}^{in}$, and μ is the rebuffering penalty which is generally set to $\mu = 4.3$.

The system ensures that the bandwidth is saved as much as possible when the QoE is the highest. The system utility includes the QoE and the bandwidth consumed compared with the situation where players request all chunks at the highest bitrate, which is:

$$\phi(V) = \omega_u \sum_{\substack{V_i \in V^{in} \\ i=1}}^{N} \alpha_i \phi(V_i) + \omega_o \sum_{\substack{V_j \in V^{out} \\ j=1}}^{N} \frac{\bar{D}_{V_j} - D_{V_j}}{\bar{D}_{V_j}}, \quad (2)$$

where $(\alpha_1,\alpha_2,\cdots,\alpha_N)$ is the importance of the tile depending on the percentage in the viewport, V_i indicates the tile inside the viewport, V_j indicates the tile outside the viewport, $\bar{D}_{V_j}$ and D_{V_j} respectively represent the bandwidth consumed by the player V_j if chunks are buffered at the highest bitrate and the bandwidth consumed by the actual download, and $\frac{\bar{D}_{V_j} - D_{V_j}}{\bar{D}_{V_j}}$ means the bandwidth saving rate of the player V_j. ω_u and ω_o are the weights of QoE and transmission cost respectively. A higher ω_u means more emphasis on the QoE and vice versa. We take $\omega_u = 0.5$ and $\omega_t = 0.5$.

We find a sequence of suitable bitrate versions for each tile V_i to schedule bandwidth resources that maximize the system utility and satisfy:

$$\begin{cases} \sum_{\substack{V_i \in V^{in} \\ i=1}}^{N} \sum_{k=1}^{K} r(V_{i,k}) \leqslant C \\ r(V_{i,k}) \in R \end{cases}. \quad (3)$$

Our solution consists of the following aspects:

- Prediction of the probability distribution of the viewport on spatial tiles, including viewport estimation and prediction based on content;
- Computation of multi-player total bandwidth and estimation of the bitrate constraint C (or throughput);
- Decision on the optimal version of each tile.

In the following section, we will address each of these aspects.

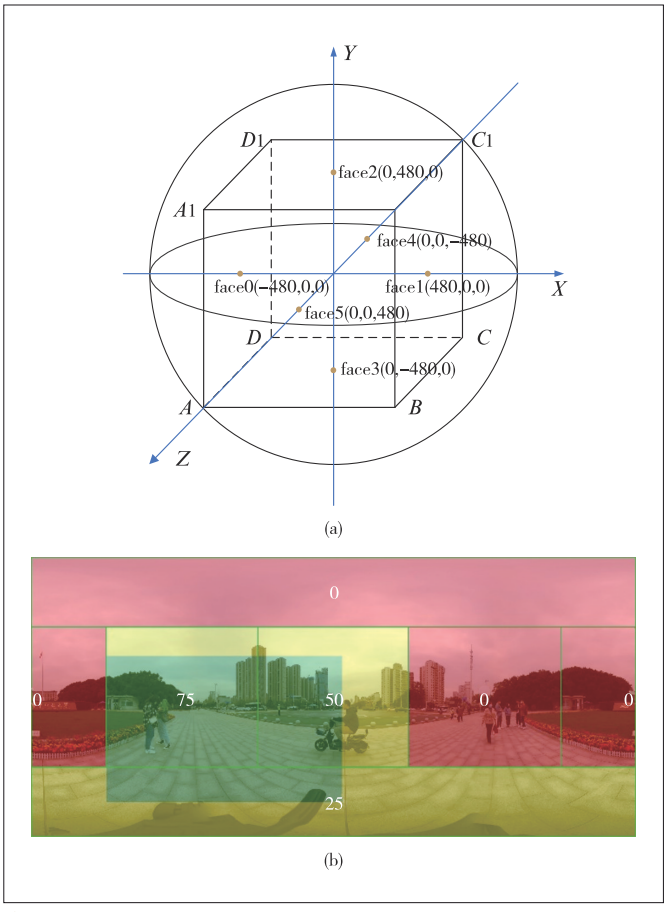
4 Viewport Prediction

A distinctive feature of 360-degree video is that the attention is not evenly distributed. Therefore, the viewport-driven stream is an effective solution to the improvement of the quality of the 360-degree video, but it is always challenging to predict the viewpoint trajectory accurately. Recent studies have proposed adaptive bitrate algorithms based on FOV prediction, but these algorithms have the following shortcomings. Firstly, it lacks consideration for the video content itself. Secondly, there is a strong dependence on the FOV, and any FOV prediction error

probably causes a decline in video quality and even significant rebuffering. Therefore, in this section, we propose a viewport prediction approach based on the priority of FOV and content.

4.1 FOV priority

The system divides the 360-degree video into six faces corresponding to a cube map as in Fig. 2(a), where six video tiles are placed on the six faces of the cube. The browser renders it into a sphere, and the VR user is located in the center of the sphere to observe the surface of the sphere. The red point is the center point on each surface of the cube, with the Cartesian coordinates of the point in brackets. The spherical coordinates of each tile mapped on the sphere are shown in Table 1, expressed in the form of latitude and longitude, and the latitude and longitude centers mean the spherical coordinate of the center of the tile. In Fig. 2(b), the green area shows the user's FOV, the yellow area indicates that a part of the current tile is inside the FOV and may be viewed later, and the red area illustrates that the current tile is absolutely outside the FOV. We calculate the overlap between the tile and FOV according to Table 2 to get the priority of the tile.



▲ **Figure 2. (a) 360-degree video segmentation that the content-aware multi-step prediction control algrithm (CAMPC) uses to judge the importance of tiles; (b) example of FOV priority allocation according to FOV at a certain moment**

The higher the value, the higher the proportion of the tile in the FOV, and the higher the bitrate version should be buffered later.

The FOV of common head-mounted devices is about 110 degrees. Since the final verification scene is a browser window, obviously it is easier to obtain the spherical coordinates of the center of the window. We define the priority of each tile based on the relative position of each face and the center of the viewport. As shown in Table 2, the system divides the priority into five levels: 100, 75, 50, 25, and 0. In order to obtain the final FOV priority, the adaptive bitrate algorithm traverses latitude and longitude in order from high score to low score to find the mapping interval of the two dimensions.

▼**Table 1. Spherical coordinates of tiles**

	Latitude Range	Latitude Center	Longitude Range	Longitude Center
V_1	[225°, 315°]	270°	[45°, 135°]	90°
V_2	[45°, 135°]	90°	[45°, 135°]	90°
V_3	[0°, 360°]	*	[0°, 45°]	0
V_4	[0°, 360°]	*	[135°, 180°]	180°
V_5	[135°, 225°]	180°	[45°, 135°]	90°
V_6	[315°, 45°]	0°	[45°, 135°]	90°

▼**Table 2. Tile priority and spherical coordinates mapping relations**

Priority	100	75	50	25	0
V_1	(90°, 270°)	(45°~135°, 225°~315°)	(10°~170°, 225°~315°) (45°~135°, 180°~360°)	(10°~170°, 180°~360°)	Others
V_2	(90°, 90°)	(45°~135°, 45°~135°)	(10°~170°, 45°~135°) (45°~135°, 0°~180°)	(10°~170°, 0°~180°)	Others
V_3	(0°~5°,*)	(5°~10°,*)	(0°~45°,*)	(0°~90°,*)	(90°~180°,*)
V_4	(175°~180°,*)	(170°~175°,*)	(135°~180°,*)	(90°~180°,*)	(0°~90°,*)
V_5	(90°, 0°)	(45°~135°, 135°~225°)	(10°~170°, 135°~225°) (45°~135°, 270°~90°)	(10°~170°, 270°~90°)	Others
V_6	(90°, 180°)	(45°~135°, 315°~45°)	(10°~170°, 315°~45°) (45°~135°, 90°~270°)	(10°~170°, 90°~270°)	Others

4.2 Content Priority

Different from predicting the future viewport based only on the online viewport, the system server will convert the original video into frames in advance, and perform operations such as gradient calculation and semantic segmentation through the pre-trained neural network model, to get the priority of bitrate allocation shown in Fig. 2(b).

In this work, we mainly use FC-DenseNets with U-Net structure as a model for extracting features. Through the 56-layer network, the model analyzes content features and then classi-

fies and slices them. After training 300 samples in 300 epochs through the model, a better checkpoint is obtained, and the following semantic segmentation results are obtained by using the own dataset as the test data. Among them, the segmented category must be consistent with the category applied in training.

The original image shown in Fig. 3(a) is inputted into the trained network model to obtain segmentation results as in Fig. 3 (b) and then transformed into an image with only 0 and 255 grayscales as shown in Fig. 3(c). Despite some errors and the defect that there are unclear boundaries of categories, the results are generally sufficient to judge the priority of content perception. We count the number of black and white pixels on the obtained grayscale image. The whiter pixels there are, the richer the image content. The higher priority of content perception means that in the MPD file of the video tiles, the bandwidth attribute corresponding to each bitrate level will be relatively higher. In other words, the video segment has a larger file size than the segment with a lower perception priority, containing complex content that may grab the user's attention. After the above process is performed on each frame of the original video, the average value of the content richness of each video segment can be calculated, and this information is stored in the JavaScript Object Notation (JSON) file used for the request for the terminal to read while allocating bandwidth among multi-players.

4.3 Probability Distributions of Future FOV

The viewport distribution probability α_i of any tile V_i in the future is determined by the FOV priority S_F and content perception priority S_C. The total priority α_i of a tile can be calculated by $S = \omega_F S_F + \omega_C S_C$, where ω_F represents the weight of S_F, ω_C represents the weight of S_C, and they depend on the current occupancy of the buffer $B_{f,k}$:

- $B_{f,k} \geqslant B_r - L$ (when $\omega_F = 0.3, \omega_C = 0.7$) means that when the video buffer is sufficient, more emphasis is placed on the score based on the richness of video content, where B_r represents the length of the safe buffer that players want to keep, and L represents the duration of the video segment.
- $B_{f,k} \leqslant L$ (when $\omega_F = 0.8, \omega_C = 0.2$) means that the FOV score is more emphasized while the insufficient buffered video is facing the danger of rebuffering.
- $L \leqslant B_{f,k} \leqslant B_r - L$ and $\omega_F = 0.5, \omega_C = 0.5$ represent that when the video buffer occupancy is within the normal range, the content score and the FOV score are jointly influenced by the priority.

S represents the importance of the impact of the current face on the QoE. Since the calculation method for the total score of each tile may be different, it is necessary to normalize the score S by $\alpha_i = \frac{S_i}{\sum_{j=1}^{N} S_j}$, and finally, get a set of probability distributions $(\alpha_1, \alpha_2, \cdots, \alpha_N)$ of future FOV.

5 Adaptive Bitrate

5.1 Video Streaming Model

The download process of video slices is modeled as a time sequence[16], with the start download time of chunk $V_{i,k}$ as the sampling point. At this time, the buffer occupancy is represented as $b(V_{i,k})$. After $V_{i,k}$ is downloaded, the buffer is consumed for $\frac{r(V_{i,k})L}{c(V_{i,k})}$ as the user watches the video and is filled with L seconds when the time has passed. Therefore, when the download of $V_{i,k}$ has been completed, the buffer occupancy can be expressed as:

$$b(V_{i,k+1}) = b(V_{i,k}) - \frac{r(V_{i,k})L}{c(V_{i,k})} + L. \tag{4}$$

The premise is $b(V_{i,k}) > \frac{r(V_{i,k})L}{c(V_{i,k})}$ to ensure that the content of the chunks in the buffer is enough to play without rebuffering. Similarly, in the case before downloading chunk $V_{i,k}$, we have

$$b(V_{i,k}) = b(V_{i,k-1}) - \frac{r(V_{i,k-1})L}{c(V_{i,k-1})} + L. \tag{5}$$

(a) Original image (b) Segmentation image (c) Binary image

▲ **Figure 3. Analyzing the priority of content perception under the semantic segmentation model**

In order to describe the relationship between the bitrate change and the buffer occupancy evolution, we make a subtraction between Eqs. (4) and (5). Since previous studies showed that the throughput of a cellular network would not change significantly for a period of time, we assume $c(V_{i,k+1}) = c(V_{i,k})$, and $\xi(V_{i,k})$ is used to ex-

press the impact of this assumption, which obtainins that

$$b(V_{i,k+1}) = 2b(V_{i,k}) - b(V_{i,k-1}) - \frac{\Delta r(V_{i,k})L}{c(V_{i,k})} + \xi(V_{i,k}) . \quad (6)$$

The adaptive bitrate algorithm among multi-players needs to balance a set of conflicting QoE elements such as video quality maximization, rebuffering events minimization, and quality fluctuations. For chunks in V^{in} that directly affect the QoE, we optimize the QoE metrics over the multi-step prediction horizon and at the same time control the future buffer occupancy. And for chunks in V^{out} of which the importance is obviously less than chunks inside FOV, we subtract the predicted bandwidth and the used bandwidth to obtain the currently available bandwidth, and then according to $(\alpha_1,\alpha_2,\cdots,\alpha_N)$ we allocate the remaining bandwidth to each tile. Note that the tiles contained in sets V^{in} and V^{out} are updated in real-time as the FOV changes.

Since the network status, in reality, is hard to predict accurately, it requires the algorithm performance to be robust to the prediction error. Therefore, minimizing buffer control error achieves quality maximization and rebuffering minimization simultaneously by keeping buffer occupancy at a constant level B_r. However, due to changes in throughput, accurate control of buffer occupancy requires frequent quality switching. Since buffer occupancy changes will not affect QoE directly and switching bitrates reduces QoE, it should tolerate some buffer occupancy fluctuations but limit the variability of video quality. Therefore, the optimization goal is designed to minimize the buffer control error and the weighted combination of the bitrate change $\Delta r(V_{i,k}) = r(V_{i,k}) - r(V_{i,k-1})$ between two consecutive video chunks. Then the total cost function from chunks 1 to K for any tiles in V^{in} is expressed as

$$J_1^K = E\{\sum_{k=1}^{K}[b(V_{i,k+1}) - b_r(V_{i,k+1})]^2 + \sum_{k=1}^{K}[\lambda_k \Delta r(V_{i,k})]^2\}, \quad (7)$$

where λ_k is the penalty of changing the bitrate at $V_{i,K}$. The algorithm controls $b(V_{i,k+1})$ to smoothly reach $b_r(V_{i,k+1})$ along the trajectory $b_r(V_{i,k+1}) = \beta b(V_{i,k}) + (1-\beta)B_r$, where $\beta \in [0,1)$. A smaller β means moving towards $b_r(V_{i,k+1})$ faster. The definition of the cost function allows us to meet the requirements of different users for video playback. A larger λ is employed if users are more concerned about smooth playback. A trimmer is adopted in cases where they do not care about bitrate variations.

When the average throughput $c(V_{i,k}),\cdots,c(V_{i,K})$ of downloading K chunks in the future is known, the bitrate $r(V_{i,k}),\cdots,r(V_{i,K})$ allocated to each chunk can be obtained by minimizing the cost function. Therefore, the bitrate selection for $V_{i,K}$ in V^{in} can be formulated as an optimal predictive control problem over an N-step horizon.

$$\min \hat{J}_{k+1}^{k+T} = E\{\sum_{t=1}^{T}[\hat{b}(V_{i,k+t}|V_{i,k}) - b_r(V_{i,k+t})]^2 + \sum_{t=1}^{T}[\lambda_t \Delta r(V_{i,k+t-1})]^2\}$$

$$\text{s.t.}\begin{cases} b(V_{i,k+1}) = 2b(V_{i,k}) - b(V_{i,k-1}) - \dfrac{\Delta r(V_{i,k})L}{c(V_{i,k})} + \xi(V_{i,k}) \\ b_r(V_{i,k+1}) = \beta b(V_{i,k}) + (1-\beta)B_r, \beta \in [0,1) \\ b(V_{i,1}) = L, b(V_{i,k}) \in [L, B_m] \\ r(V_{i,k}) \in R, \lambda_t = \lambda(V_{i,T-t+1}) \end{cases}, \quad (8)$$

where the $\hat{b}(V_{i,k+t}|V_{i,k})$ is the t-step ahead estimated value of the buffer occupancy while downloading $V_{i,k+1}$ up to $V_{i,k+t}$, and λ_t is the discount rate for switching bitrate from $r(V_{i,k+t}^{\text{in}})$ to $r(V_{i,k+t+1}^{\text{in}})$. And as the number of prediction steps increases, λ_t is gradually reduced to $\lambda(V_{i,T-t+1})$, representing that the far future will have less impact on the current cost.

With the bandwidth prediction $\hat{c}[V_{i,k}, V_{i,k+T-1}]$ of the future K chunks as the input, the prediction optimization controller outputs the bitrate $r(V_{i,k})$ selected for $V_{i,K}$ so that the QoE indicators achieve the expected balance.

5.2 Link Bandwidth Predictor

The algorithm first gets the real-time throughput and then uses the Kalman filter method to obtain the predicted value used in the bitrate selector for control optimization.

The Kalman filter dynamically adjusts parameters to output the estimated value for online prediction. The prediction is based on two equations: a dynamic state equation that describes the dynamics hidden state (e.g., the predicted bandwidth) and a static output equation describing the relationship between the hidden state and the measured value (e.g., throughput). The Kalman filter method, which can filter out temporary throughput fluctuations and reflect the stable change, matches the observations in the previous work that the evolution of the throughput within a session exhibits stateful characteristics and the throughput is essentially Gaussian within each state.

The throughput prediction model based on the Kalman filter employs a classic time series model and an auto-regressive model assuming $c(V_{i,k+1}) = \sum_{j=0}^{p} a_j c(V_{i,k-j}) + w(V_{i,k})$, where $\{a_j\}_{j=0}^{p}$ is the weight parameter and $w(V_{i,k})$ is Gaussian noise with zero mean, satisfying $W = E[w(V_{i,k})^2]$. Existing works[17-18] have shown that the network throughput is piecewise stationary. The statistical properties, including mean and variance, do not change over tens of seconds or minutes. We assume the dynamic state model is $c(V_{i,k+1}) = c(V_{i,k}) + w(V_{i,k})$, where $c(V_{i,k})$ represents the average download speed that needs to be estimated during the downloading process of

the video chunk $V_{i,k}$.

Let $v(V_{i,k})$ denote the video throughput measurement when downloading chunk $V_{i,k}$. Recent studies show that the observed throughput fluctuates around the link capacity following Gaussian. Therefore, $v(V_{i,k})$ is modeled as the summation of capacity $c(V_{i,k})$ and measurement noise $q(V_{i,k})$ denoted by $Q = E[q(V_{i,k})^2]$. Finally, the whole system model is given by:

$$\begin{cases} c(V_{i,k+1}) = c(V_{i,k}) + w(V_{i,k}) \\ v(V_{i,k}) = c(V_{i,k}) + q(V_{i,k}) \end{cases}. \tag{9}$$

The Kalman filter consists of model prediction and measurement correction. In the prediction stage, the Kalman filter uses the estimated value of the previous link capacity $\hat{c}(V_{i,k-1})$ to predict the current state:

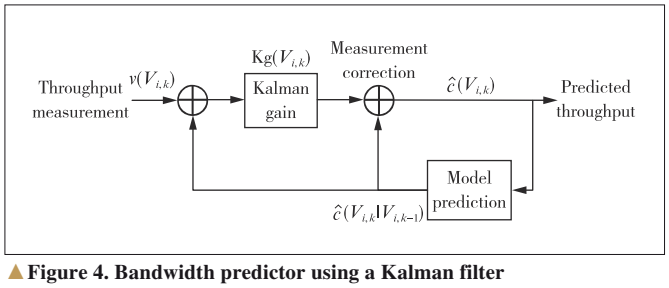
$$\hat{c}(V_{i,k}|V_{i,k-1}) = \hat{c}(V_{i,k-1}). \tag{10}$$

Then the initial estimate of capacity $\hat{c}(V_{i,k}|V_{i,k-1})$ is corrected to a new estimated value $\hat{c}(V_{i,k})$ by the measurement correction model. The correction equation is:

$$\hat{c}(V_{i,k}) = \hat{c}(V_{i,k}|V_{i,k-1}) + Kg(V_{i,k})\left[v(V_{i,k}) - \hat{c}(V_{i,k}|V_{i,k-1})\right]. \tag{11}$$

Fig. 4 shows how to obtain a new estimate $\hat{c}(V_{i,k})$. The difference between the initial estimate $\hat{c}(V_{i,k}|V_{i,k-1})$ and the measured throughput $v(V_{i,k})$ is multiplied by the Kalman gain $\mathrm{Kg}(V_{i,k})$, which then serves as a correction to the initial estimate $\hat{c}(V_{i,k})$. The estimation produced by the Kalman filter is attributed to two terms: the previous estimate $\hat{c}(V_{i,k-1})$ and the throughput $v(V_{i,k})$. The Kalman gain $\mathrm{Kg}(V_{i,k})$ balances contributions of the two terms: a larger one means more weight is given to the measured value; conversely, a smaller one denotes that the model trusts the estimated value more. The updated process of the Kalman gain is:

$$\mathrm{Kg}(V_{i,k}) = \frac{P(V_{i,k-1}) + W}{P(V_{i,k-1}) + W + Q}, \tag{12}$$



▲ **Figure 4. Bandwidth predictor using a Kalman filter**

where $P(V_{i,k})$ is the system error defined as $P(V_{i,k}) = E\left[\left(\hat{c}(V_{i,k}) - c(V_{i,k})\right)^2\right]$. This value will be recursively modified at each step:

$$P(V_{i,k}) = \left(1 - \mathrm{Kg}(V_{i,k})\right)\left(P(V_{i,k-1}) + W\right). \tag{13}$$

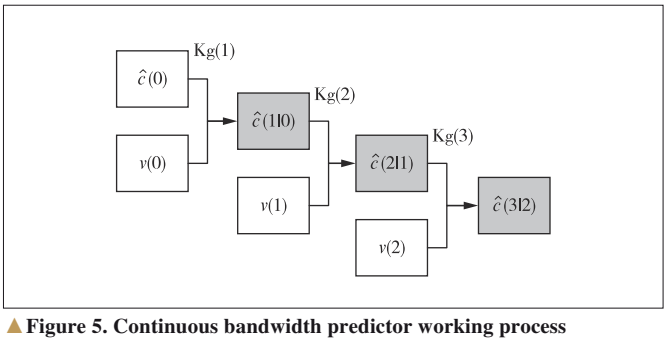
The above process is executed continuously, obtaining the predicted throughput as the gray blocks in Fig. 5 while buffering the next chunk every time. In order to obtain the multi-step bandwidth prediction value, the average throughput of five video chunks in the future is predicted at each step. Since the average throughput measurements $v(V_{i,k+1}), v(V_{i,k+2}), v(V_{i,k+3})$, and $v(V_{i,k+4})$ are unknown when downloading chunks $V_{i,k}$, the network throughput can be approximately stable for a while[17-18]. Therefore, it is assumed that the measurement throughput $v(V_{i,k})$ of the next chunks meets $v(V_{i,k+1}) = v(V_{i,k+2}) = v(V_{i,k+3}) = v(V_{i,k+4}) = v(V_{i,k})$. Fig. 6 shows the prediction iteration process, which is the continuation of the one-step Kalman filter. Blocks of the same color have the same predicted value, and the gray blocks have the same value as the same block in Fig. 5. Various parameter initialization problems involved in the process are described in Section 6.

5.3 Bitrate Selector

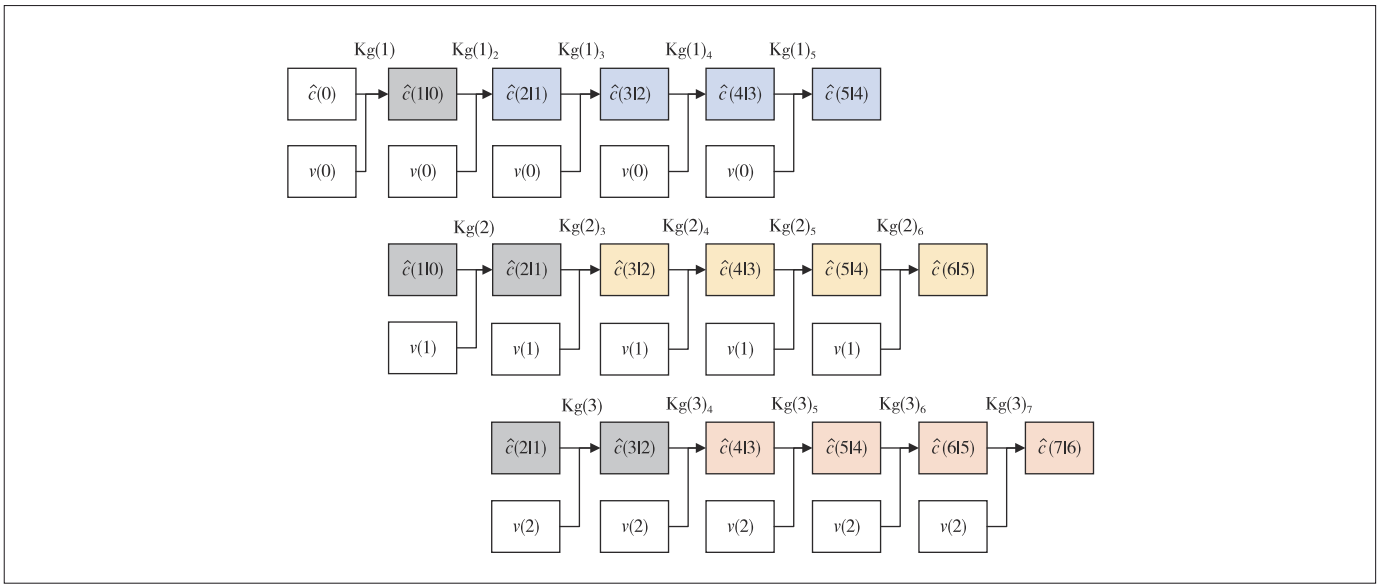
The CAMPC selects the bitrate according to the minimization cost function in Eq. (8) within the prediction range. To solve it, we first obtain the multi-step prediction of buffer occupancy based on the predicted link bandwidth. With the future buffer occupancy expressed by a bitrate function, the cost function can be derived to be only relevant to the variable video bitrate. Then, the bitrate that minimizes the cost function can be obtained.

After obtaining the average throughput $c(V_{i,k}),\cdots,c(V_{i,K})$ of downloading K chunks in the future, the buffer occupation of the t-step is defined as $b(V_{i,k})$ and $b(V_{i,k-1})$, experiencing a recursive process:

$$\hat{b}(V_{i,k+t}|V_{i,k}) = (t+1)b(V_{i,k}) - tb(V_{i,k-1}) - \frac{N\Delta r(V_{i,k})L}{c(V_{i,k})} - \cdots - \frac{2\Delta r(V_{i,k+t-2})L}{c(V_{i,k+t-2})} - \frac{\Delta r(V_{i,k+t-1})L}{c(V_{i,k+t-1})}. \tag{14}$$



▲ **Figure 5. Continuous bandwidth predictor working process**



▲ **Figure 6. Multi-step bandwidth predictor working process**

Then the buffer occupation of the player V_i is finally given by the following:

$$\hat{B}=\begin{bmatrix}\hat{b}(V_{i,k+1})\\ \hat{b}(V_{i,k+2})\\ \cdots\\ \hat{b}(V_{i,k+K})\end{bmatrix}=\begin{bmatrix}2b(V_{i,k})-b(V_{i,k-1})\\ 3b(V_{i,k})-2b(V_{i,k-1})\\ \cdots\\ (T+1)b(V_{i,k})-Tb(V_{i,k-1})\end{bmatrix}+$$
$$\begin{bmatrix}-\dfrac{L\Delta r(V_{i,k})}{c(V_{i,k})}\\ -\dfrac{2L\Delta r(V_{i,k})}{c(V_{i,k})}-\dfrac{L\Delta r(V_{i,k+1})}{c(V_{i,k+1})}\\ \cdots\\ -\dfrac{TL\Delta r(V_{i,k})}{c(V_{i,k})}\cdots-\dfrac{2L\Delta r(V_{i,k+T-2})}{c(V_{i,k+T-2})}-\dfrac{L\Delta r(V_{i,k+T-1})}{c(V_{i,k+T-1})}\end{bmatrix}. \tag{15}$$

Among them, $\Delta r(V_{i,k+T-2})$ is replaced by $z^{-1}\Delta r(V_{i,k+T-1}),\cdots$, and $\Delta r(V_{i,k+1})$ is replaced by $z^{-T+2}\Delta r(V_{i,k+T-1})$. Similarly $b(V_{i,k})$ is replaced by $z^{-1}b(V_{i,k})$, and then the $\hat{b}(V_{i,k+T})$ is formulated as:

$$\hat{b}(V_{i,k+T})=(T+1-Tz^{-1})b(V_{i,k})+$$
$$(-\frac{L}{\hat{c}(V_{i,k+T-1})}-\frac{2L}{\hat{c}(V_{i,k+T-2})}z^{-1}-\cdots-\frac{TL}{\hat{c}(V_{i,k})}z^{-T+1})=$$
$$G_T(z^{-1})b(V_{i,k})+F_T(z^{-1})\Delta r(V_{i,k+t-1})\ . \tag{16}$$

The vector of future buffer occupancy is $\hat{B}=\boldsymbol{GB}(V_{i,k})+\boldsymbol{F}\Delta R$, which can be written as:

$$\hat{B}=\begin{bmatrix}2&-1\\ 3&-2\\ 4&-3\\ \vdots&\vdots\\ T+1&-T\end{bmatrix}\begin{bmatrix}b(V_{i,k})\\ b(V_{i,k-1})\end{bmatrix}+$$
$$\begin{bmatrix}-\dfrac{L}{c(V_{i,k})}&0&\cdots&0\\ -\dfrac{2L}{c(V_{i,k})}&-\dfrac{L}{c(V_{i,k+1})}&\cdots&0\\ \vdots&\vdots&\ddots&\vdots\\ -\dfrac{TL}{c(V_{i,k})}&-\dfrac{(N-1)L}{c(V_{i,k+1})}&\cdots&-\dfrac{L}{c(V_{i,k+T-1})}\end{bmatrix}$$
$$\begin{bmatrix}\Delta r(V_{i,k})\\ \Delta r(V_{i,k+1})\\ \cdots\\ \Delta r(V_{i,k+T-1})\end{bmatrix}. \tag{17}$$

Substituting $\hat{B}=\boldsymbol{GB}(V_{i,k})+\boldsymbol{F}\Delta R$ into the cost function Eq. (8), we get:

$$\min \hat{J}_{k+1}^{k+T}=E\{\sum_{t=1}^{T}[\boldsymbol{GB}(V_{i,k})+\boldsymbol{F}\Delta R-B_r]^T$$
$$[\boldsymbol{GB}(V_{i,k})+\boldsymbol{F}\Delta R-B_r]+\Delta R^T\Lambda\Delta R\}$$
$$\text{s.t.}\begin{cases}\hat{B}=[\hat{b}(V_{i,k+1}),\hat{b}(V_{i,k+2}),\cdots,\hat{b}(V_{i,k+T})]^T\\ \Delta R=[\Delta r(V_{i,k}),\Delta r(V_{i,k+1}),\cdots,\Delta r(V_{i,k+T-1})]^T\\ B(V_{i,k})=[b(V_{i,k}),b(V_{i,k-1})]^T\\ B_r=[B_r,B_r,B_r,B_r,B_r]^T\\ \Lambda=\mathrm{diag}(\lambda_1,\lambda_2,\cdots,\lambda_T)\ .\end{cases} \tag{18}$$

In order to minimize the cost, let $\frac{\partial \hat{J}_{k+1}^{k+T}}{\partial \Delta R} = 0$, and then we get $\Delta R = (F^T F + \Lambda)^{-1} F^T (B_r - GB(V_{i,k}))$. The final bandwidth allocation takes the first vector of ΔR, that is $\Delta r(V_{i,k})$, and the bitrate finally selected for the chunk $V_{i,k}$ is $h(V_{i,k})$. We assume $H(V_{i,k}) = \{h_1, h_2, h_3, \cdots, h_m\}$ to be the set of available bitrate versions for chunk $V_{i,k}$, where $h(V_{i,k}) = h_i$ satisfies $h_i < r(V_{i,k-1}) + \Delta r(V_{i,k})$ and $h_{i+1} > r(V_{i,k-1}) + \Delta r(V_{i,k})$.

The above bandwidth scheduling method helps spatial tiles V^{in} with high probability in the future viewport allocated bandwidth. For tiles V^{out} segmentation with low probability in the FOV, the system first uses the one-step bandwidth predictor to obtain the predicted link capacity. After the bandwidth allocation of the V^{in}, the difference between the predicted value and the consumed value indicates the currently available bandwidth, which will be scheduled to each tile out of FOV according to the weight $(\alpha_1, \alpha_2, \cdots, \alpha_N)$ given in Section 4.

6 Evaluation

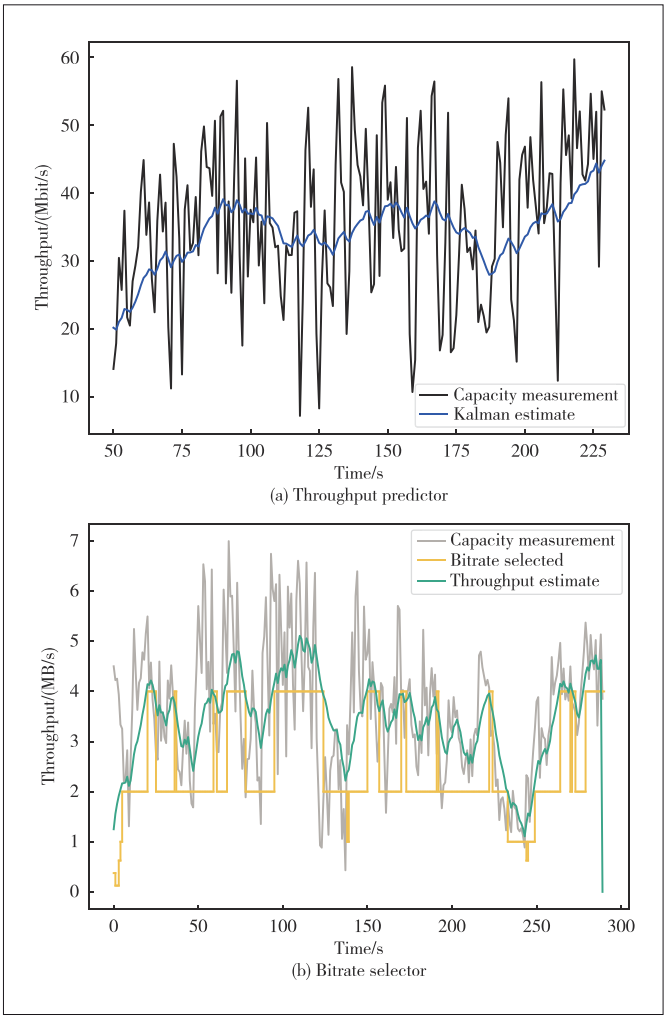
6.1 Methodology

6.1.1 Experimental Setup

Our emulated DASH system consists of a server and a video player based on Dash.js (version 3.2), an open-source implementation of the DASH standard. The client video player is a Google Chrome browser. The throughput is computed by the method in the next section in real time. Key classes of adaptive streaming-related functions are modified. First of all, we modify the media-player settings and add attributes that indicate the current chunk and tile serial number. ABR algorithms are implemented in the AbrController class, and HTTP requests for throughput measurement are collected from the Throughput History function. At the same time, we modify the buffer threshold of each player when the FOV is switched. For the tile in the viewport, the maximum buffer size is 10 s, and the safety threshold is 6 s. While the tile is outside FOV, the above two values are 4 s and 2 s, respectively. The purpose is to respond to this change faster when the viewing angle is switched. We use the Angular.js framework to unify the front-end communication of the platform, monitoring system state information such as buffer occupancy, bitrates, rebuffering time, and the predicted/actual capacity. These also are logged for the performance analysis.

6.1.2 Link Capacity Traces

To verify the effectiveness of the throughput predictor and control optimizing model on a single media player in realistic network conditions, we use link capacity traces based on the public dataset, which collects throughput measurements in 4G/LTE networks. Belgium 4G/LTE dataset records the available bandwidth while downloading a large file in and around Ghent, Belgium. Figs. 7(a) and 7(b) show the single simulator's predictor and selector, respectively, running status on a 4G/LTE trace, which can verify the algorithm accuracy on one player.



▲ **Figure 7. Single simulator running status over Belgium 4G/Long Term Evolution (LTE) dataset**

6.1.3 Video Parameters

We transform a 4K 360-degree panoramic video into six faces corresponding to a cube map for evaluation. Each video tile with 1 080 resolution is split into 123 chunks of 1 s and is encoded by VP9 codec in six bitrate versions, i.e., {0.18, 0.45, 0.91, 3.10, 4.55, 6.05} Mbit/s. The specific bitrate mapping to the same level among different video tiles varies with the video content richness; that is, a video about the sky is often simpler and has a lower bitrate than a portrait video.

6.1.4 Adaptation Algorithms

We evaluate CAMPC and the following widely adopted ABR algorithms and simple ABR rules for a 360-degree video:

• DASH. js. DYNAMIC dynamically switches between ThroughputRule and BolaRule. These two algorithms are based on rate and buffer to select bitrate.

• DASH.js.LoL+ is based on the learning adaptive bitrate algorithm Low-on-Latency (LoL), with improving adjustment of the weight for the self-organizing mapping (SOM) features and controlling the playback speed and taking into account latency and buffering levels.

• FOV is an adaptive bitrate algorithm based on the real-time viewport implemented by our experiment platform. The bandwidth allocation weight is calculated according to the angle deviation between each tile in the viewport and the center point of the FOV.

• FOVCONTENT is an adaptive bitrate algorithm based on the real-time viewport and content perception implemented by our experiment platform. It predicts the viewport based on the user's current viewport and the content richness obtained from offline training, and the average throughput within a period is used for bit rate allocation.

6.2 Choice of CAMPC Parameters

In the bandwidth predictor, we set the initial system error variance $P\left(V_{i,0}\right)$ to 7 Mbit/s and the process noise variance W to 3 Mbit/s, which denotes that the prior estimate of the network bandwidth fluctuates by 3 Mbit/s. When the initial value is set, it is necessary to ensure that the system error variance is not less than the process noise variance. A higher variance of the system error will make the prediction process more trustworthy in the throughput measurement at the initial stage, resulting in a better fitting curve. The measuring noise is updated by $Q(k) = \alpha Q(k-1) + (1-\alpha)[v(k) - \hat{c}(k|k-1)]^2, \alpha = 0.8$. The initial value of the throughput estimate $c\left(V_{i,0}\right)$ is set to 8 Mbit/s.

6.3 Real-Time Throughput Measurement

Different from the single player, which can directly measure the throughput according to the application programming interface (API) in DASH to get the throughput, multiple players share the link for synchronous transmission, with inaccuracy in the throughput obtained by any player from API. According to DASH, at the end of transmission for a chunk, the start timestamp d, the end timestamp, and the number of data transferred can be obtained by the player. Therefore, the transmission state of the video chunks, as in Fig. 8, shows a possible state.

The outermost black box represents a time slot, which is the smallest unit of our timing measurement throughput. The blue blocks in Fig. 8 indicate the downloading process of each player. The value on the blue block indicates which player is downloading. Although they request videos in sequential order $V_{i,k}, V_{i,k+1}, V_{i,k+2}, \cdots$, the order in which the players appear in the picture is random since each player is an independent download process. The blue blocks marked by the dotted line represents the download process that players' monitor cannot capture at the end of the time slot. Therefore, when calculating the throughput, the time should remove the part that has no data transmission from a slot and the total number of data should be all the data that can be sensed.

Assuming that the start time of the slot is t and the end time is $t + d$, in order to avoid the impact of the chunks that cannot be captured as much as possible, we move the start and end timestamps forward for a short period d_{back}, then the actual start time of the time slot for calculating the throughput is $t - d_{\text{back}}$, and the actual end time is $t + d - d_{\text{back}}$.

The calculator cyclically judges whether the response starts timestamp req_k and finishes timestamp fin_k in the HTTP request list satisfy $\text{req}_k < t + d - d_{\text{back}}$ and $t - d_{\text{back}} < \text{fin}_k$; if satisfied, it indicates that at least part of the download process within the time gap needs to be further judged and calculated:

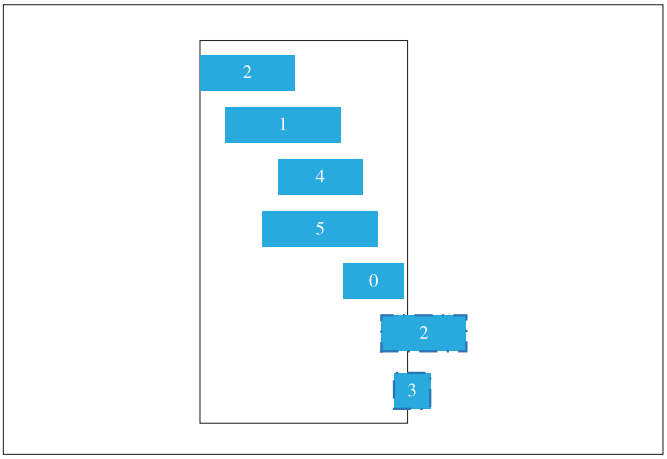
• When the start and end time of the request response is both within the timestamp, that is $t - d_{\text{back}} < \text{req}_k$ and $\text{fin}_k \leqslant t + d - d_{\text{back}}$, the total number of downloaded data D_k is directly added to the total number of downloaded data in this time slot;

• When the end time of the request response is outside the right timestamp, that is $t - d_{\text{back}} < \text{req}_k$ and $\text{fin}_k > t + d - d_{\text{back}}$, all the downloaded data are in proportion $\frac{t + d - d_{\text{back}} - \text{req}_k}{\text{fin}_k - \text{req}_k}$ to the total downloaded data in this time slot.

• When the request response start time is outside the time stamp on the left, that is $\text{req}_k < t - d_{\text{back}}$ and $\text{fin}_k \leqslant t + d - d_{\text{back}}$, add all the downloaded data D_k in proportion $\frac{\text{fin}_k - (t - d_{\text{back}})}{\text{fin}_k - \text{req}_k}$ are added to the total downloaded data in this time slot.

• When the start time and end time of the request response are outside the timestamp, that is $\text{req}_k < t - d_{\text{back}}$ and $\text{fin}_k > t + d - d_{\text{back}}$, the total number of downloaded data D_k in proportion $\frac{d}{\text{fin}_k - \text{req}_k}$ are added to the total downloaded data in this time slot.

In addition, it is necessary to avoid gaps in which no data is transmitted to the middle, beginning, and end of the slot due to buffer control rather than the current link capacity being 0. In the process, we record the current minimum request-



▲ **Figure 8. Time sequence of downloading process that may occur in multiple players**

response start timestamp req_{min}, which means there is no gap after the timestamp. If fin_k is less than req_{min}, then we subtract the gap time $req_{min} - fin_k$ from d. Finally, the throughput of the current time slot can be obtained by dividing the actual time interval by the total number of actual transmitted data.

6.4 Performance

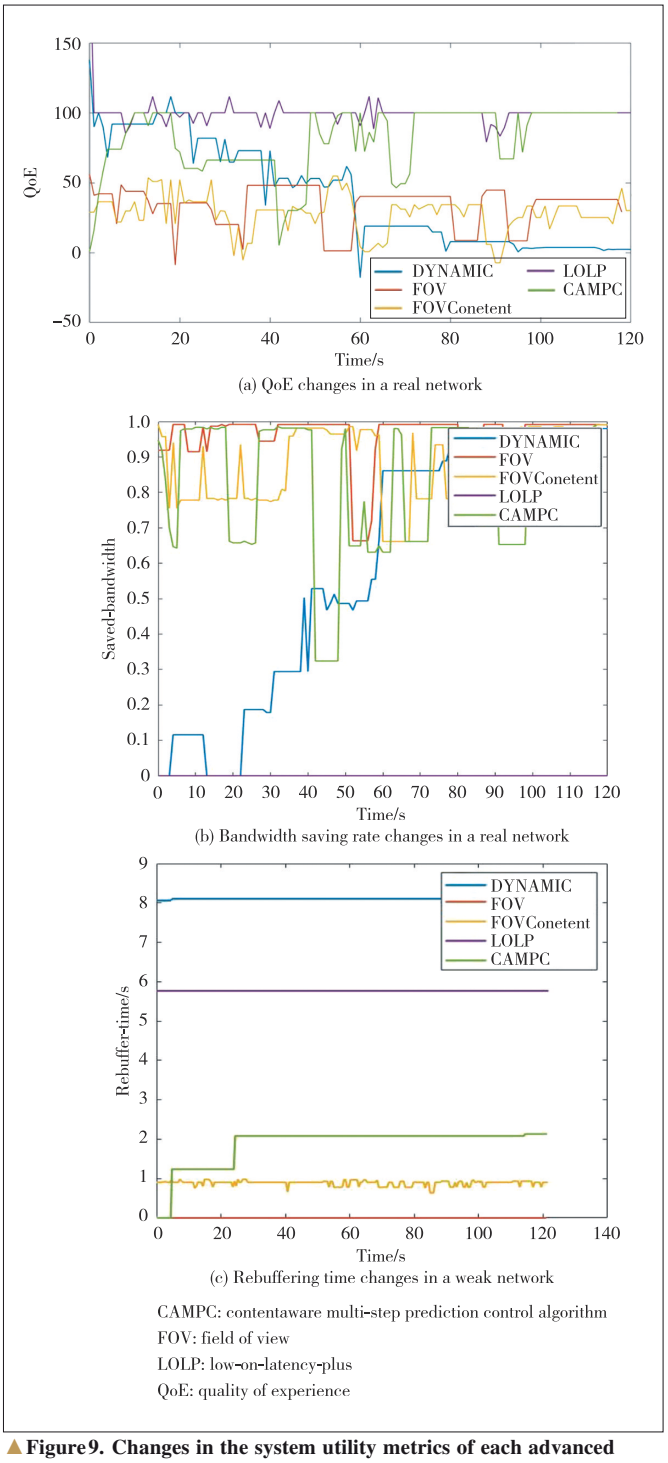
We compare the performance of ABR algorithms in a real and a weak network environment limited by the NetLimiter 4.

As shown in Fig. 9(a), the momentary fluctuation of QoE occurs when the viewport changes. Because the two rules of Dash. js treat each video player fairly, even if the viewport changes suddenly, the QoE fluctuation is minimal. For other rules, the tiles not in the FOV are often transmitted at a lower bitrate to save bandwidth, resulting in severe (30 – 50 s) QoE fluctuation when the FOV suddenly changes to a player outside the original viewport. However, the CAMPC rule can recover to the highest achievable QoE and remain stable in about 5 s, while the FOV and FOVContent rules need about 10 s to deal with this sudden change.

Fig. 9(b) shows the time-varying bandwidth saving rate compared with full video transmission. Although the CAMPC rule is slightly inferior to the LoL+ rule on QoE, it can achieve a bandwidth saving rate of 83% for tiles not in the viewport. More costs for transmissions are saved and congestion is reduced when multiple users request the same video source. No rebuffering event has occurred in the actual network environment.

In the actual network environment, the performance comparison of each ABR algorithm is shown in Table 3. The user QoE of the LoL+ rule can reach 100; the DYNAMIC rule shows that the initial bitrate can be reached in multiple tests, but the bitrate requested gradually decreases owing to the vicious circle of continuously requesting lower bitrates; the average QoE of the CAMPC rule proposed by this paper can be maintained at 80. The FOV and FOVContent rules cannot respond to the change of the user viewport in time, resulting in a sudden decrease in quality every time the user viewport changes. The bitrate is allocated according to the weight directly, leading to the case that the bandwidth within the user viewport is not made full use of, and the average QoE is low. The FOV and FOVContent rules have the highest bandwidth savings, reaching about 90%, and CAMPC can achieve 83% bandwidth savings. If both bandwidth savings and QoE are given a weight of 0.5, CAMPC can get the highest system utility of 81.9. The weight can be changed according to the importance of the cost and the QoE.

We conducted the same test in a weak network environment (the speed limit is 0.5 MB/s). Changes in the system utility metrics of each ABR rule over time show the same laws as in the natural environment, but the average QoE is lower and unstable. The performance comparison of the ABR rules in a weak network environment is shown in Table 4, where the bandwidth saving rate of each rule is about 90%. Rebuffering time changes are shown in Fig. 9(c). The FOV rule does not have rebuffering, the FOVContent rule has a rebuffering time of less than 1 s, and the CAMPC algorithm is the next, but all of these are below 2 s, which accounts for less than 1.64% of the total length of the video. The LoL+ and DYNAMIC rules have a relatively high rebuffering time, accounting for 4.73%



(a) QoE changes in a real network

(b) Bandwidth saving rate changes in a real network

(c) Rebuffering time changes in a weak network

CAMPC: contentaware multi-step prediction control algorithm
FOV: field of view
LOLP: low-on-latency-plus
QoE: quality of experience

▲**Figure 9. Changes in the system utility metrics of each advanced adaptive bitrate (ABR) rule over time**

▼ **Table 3. Performance comparison of ABR algorithms in a real network environment**

	CAMPC	DYNAMIC	LoL+	FOV	FOVContent
QoE	80.356	41.111	100.011	33.630	28.222
Saved-bandwidth	0.834 6	0.596	0	0.947	0.884
Utility	81.908	50.356	50.006	64.165	58.311

ABR: adaptive bitrate CAMPC: content-aware multi-step prediction control algorithm
FOV: field of view LoL+: Low-on-Latency-plus
QoE: quality of experience

and 6.64% of the total video length, respectively. The FOV rule has the highest system utility because, with the current rate limit of 0.5 MB/s, it happens to be enough for the player with the highest weight in FOV to request a video chunk with a quality of 6. If the network speed is lower than 0.5 MB/s, the FOV rule must have a severe rebuffering event. However, what is certain is that the FOV rule has better system utility in the range of about 0.5 MB/s. CAMPC rules are better than LoL+ and DYNAMIC rules in QoE, bandwidth saving rate, and rebuffering time. The overall utility is slightly inferior to the FOV and FOVContent rules.

▼ **Table 4. Performance comparison of ABR algorithms in a weak network environment**

	CAMPC	DYNAMIC	LoL+	FOV	FOVContent
QoE	12.051	4.690	6.565	23.526	6.904
Saved-bandwidth	0.975	0.954	0.941	0.926	0.790
Rebuffer-time	1.867	8.103	5.767	0	0.886 9
Utility	54.776	52.39	50.332	64.165	58.311

ABR: adaptive bitrate CAMPC: contentaware multi-step prediction control algorithm
FOV: field of view LoL: Low-on-Latency
QoE: quality of experience

7 Conclusions

Existing adaptive bitrate algorithms cannot provide smooth video quality for the 360-degree video in a network with high dynamic characteristics because of uncertain viewport prediction and bitrate selection. In order to achieve good QoE, the algorithm proposed in this paper considers the future multi-step network status and combines the richness of video content and the real-time user viewport to predict the future FOV, which can effectively save bandwidth resources. CAMPC uses a multi-step predictive control formulation that selects bitrate by controlling the buffer occupancy and optimizing QoE metrics over the prediction horizon. The formulation can select the bitrate level with the highest QoE and high fault tolerance. Through the above two prediction algorithms and control optimization, a content-aware 360-degree video ABR algorithm has been designed. The algorithm is implemented on the DASH video player and evaluated in reality. Experimental results show that CAMPC can save 83.5% of bandwidth resources compared with the scheme that completely transmits the tiles outside the viewport with the DASH protocol. Besides, the proposed method can improve the system utility by 62.7% and 27.6% compared with official and viewport-based rules, respectively.

References

[1] AFZAL S, CHEN J S, RAMAKRISHNAN K K. Characterization of 360-degree videos [C]//Proceedings of the Workshop on Virtual Reality and Augmented Reality Network. ACM, 2017: 1 – 6. DOI: 10.1145/3097895.3097896
[2] XIE L, XU Z M, BAN Y X, et al. 360ProbDASH: improving QoE of 360 video streaming using tile-based HTTP adaptive streaming [C]//Proceedings of the 25th ACM international conference on Multimedia. ACM, 2017: 345 – 323. DOI: 10.1145/3123266.3123291
[3] QIAN F, HAN B, XIAO Q Y, et al. Flare: practical viewport-adaptive 360-degree video streaming for mobile devices [C]//Proceedings of the 24th Annual International Conference on Mobile Computing and Networking. ACM, 2018: 99 – 114. DOI: 10.1145/3241539.3241565
[4] SONG J R, YANG F Z, ZHANG W, et al. A fast FoV-switching DASH system based on tiling mechanism for practical omnidirectional video services [J]. IEEE transactions on multimedia, 2020, 22(9): 2366 – 2381. DOI: 10.1109/TMM.2019.2957976
[5] SUN L Y, DUANMU F, LIU Y, et al. Multi-path multi-tier 360-degree video streaming in 5G networks [C]//Proceedings of the 9th ACM Multimedia Systems Conference. ACM, 2018: 162 – 173. DOI: 10.1145/3204949.3204978
[6] ZHANG Y X, GUAN Y S, BIAN K G, et al. EPASS360: QoE-aware 360-degree video streaming over mobile devices [J]. IEEE transactions on mobile computing, 2021, 20(7): 2338 – 2353. DOI: 10.1109/tmc.2020.2978187
[7] GUAN Y, ZHENG C Y, ZHANG X G, et al. Pano: optimizing 360° video streaming with a better understanding of quality perception [C]//Proceedings of the ACM Special Interest Group on Data Communication. ACM, 2019: 394 – 407. DOI: 10.1145/3341302.3342063
[8] FENG X L, SWAMINATHAN V, WEI S. Viewport prediction for live 360-degree mobile video streaming using user-content hybrid motion tracking [J]. Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies, 2019, 3(2): 1 – 22. DOI: 10.1145/3328914
[9] QIAO M L, XU M, WANG Z L, et al. Viewport-dependent saliency prediction in 360° video[J]. IEEE transactions on multimedia, 2021, 23: 748 – 760. DOI: 10.1109/TMM.2020.2987682
[10] YUAN H, ZHAO S Y, HOU J H, et al. Spatial and temporal consistency-aware dynamic adaptive streaming for 360-degree videos [J]. IEEE journal of selected topics in signal processing, 2020, 14(1): 177 – 193. DOI: 10.1109/JSTSP.2019.2957981
[11] WEI X K, ZHOU M L, KWONG S, et al. A hybrid control scheme for 360-degree dynamic adaptive video streaming over mobile devices [J]. IEEE transactions on mobile computing, 2022, 21(10): 3428 – 3442. DOI: 10.1109/TMC.2021.3058099
[12] SOBHANI A, YASSINE A, SHIRMOHAMMADI S. A video bitrate adaptation and prediction mechanism for HTTP adaptive streaming [J]. ACM transactions on multimedia computing, communications, and applications, 2017, 13(2): 1 – 25. DOI: 10.1145/3052822
[13] ZHOU C, LIN C W, ZHANG X G, et al. Buffer-based smooth rate adaptation for dynamic HTTP streaming [C]//Proceedings of 2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference. IEEE, 2013: 1 – 9
[14] YUAN H, HU X Q, HOU J H, et al. An ensemble rate adaptation framework for dynamic adaptive streaming over HTTP [J]. IEEE transactions on broadcasting, 2020, 66(2): 251 – 263. DOI: 10.1109/TBC.2019.2954074
[15] SPITERI K, URGAONKAR R, SITARAMAN R K. BOLA: Near-optimal bitrate adaptation for online videos [C]//The 35th Annual IEEE International Conference on Computer Communications. IEEE, 2016: 1 – 9. DOI: 10.1109/INFOCOM.2016.7524428

[16] YUAN H, FU H Y, LIU J, et al. Non-cooperative game theory based rate adaptation for dynamic video streaming over HTTP [J]. IEEE transactions on mobile computing, 2018, 17(10): 2334 – 2348. DOI: 10.1109/TMC.2018.2800749
[17] SUN Y, YIN X Q, JIANG J C, et al. CS2P: improving video bitrate selection and adaptation with data-driven throughput prediction [C]//Proceedings of the 2016 ACM SIGCOMM Conference. ACM, 2016: 272 – 285. DOI: 10.1145/2934872.2934898
[18] AKHTAR Z, NAM Y S, GOVINDAN R, et al. Oboe: auto-tuning video ABR algorithms to network conditions [C]//Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. ACM, 2018: 44 – 58. DOI: 10.1145/3230543.3230558

Biographies

GAO Nianzhen received her bachelors' degree in computer science and technology from Sichuan University, China in 2020. She is currently working toward the PhD degree with the Research Center of 6G Mobile Communications, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, China. Her research interests include multimedia transmission and mobile edge computing.

YU Yifang received his MS degree in engineering from Xi'an Jiaotong University, China. He currently serves as the Senior Vice President of ZTE Corporation and President of the Cloud Video and Energy Product Operation Division. He has engaged in market planning and operations management in the telecommunications industry for over 20 years. His research interests include traditional telecom networks as well as the emerging fields such as cloud computing and the mobile Internet.

HUA Xinhai received his PhD degree from Nanjing University, China. He is currently the Vice President of ZTE Corporation and General Manager of Cloud Video Product Department. His research interests include cloud computing, IP-based video product technology and solutions, video business security solutions, content distribution network technology, product solutions, etc.

FENG Fangzheng received his bachelors' degree in communication engineering from Hunan University, China in 2019. He is currently working toward the PhD degree with the Research Center of 6G Mobile Communications, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Hubei, China. His research interests include wireless communication and mobile multimedia transmission.

JIANG Tao (taojiang@hust.edu.cn) received his PhD degree in information and communication engineering from Huazhong University of Science and Technology, China in 2004. He is currently a Distinguished Professor with the Research Center of 6G Mobile Communications and School of Cyber Science and Engineering, Huazhong University of Science and Technology. He has authored or coauthored more than 300 technical papers in main journals and conferences, and nine books or chapters in the areas of communications and networks. He has served or is serving as an associate editor of some technical journals in communications, including the *IEEE Network*, *IEEE Transactions on Signal Processing*, *IEEE Communications Surveys and Tutorials*, *IEEE Transactions on Vehicular Technology*, and *IEEE Internet of Things Journal*, and he is an associate editor-in-chief of *China Communications*. His main research directions include wireless communication, mobile multimedia transmission, etc.