CA-Live360: Crowd-Assisted Transcoding and Delivery for Live 360-degree Video Streaming

Yunxiao Ma[a], Changqiao Xu[a,*], Zhonghui Wu[a], Renjie Ding[a], Han Xiao[a], Lujie Zhong[b], Yirong Zhuang[c], Gabriel-Miro Muntean[d]

[a]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, P. R. China.*
[b]*Information Engineering College, Capital Normal University, Beijing 100048, P. R. China.*
[c]*China Telecom Research Institute, Guangzhou 510630, P.R. China.*
[d]*Performance Engineering Laboratory, School of Electronic Engineering, Dublin City University, Dublin 9, Ireland.*

Abstract

In the burgeoning field of digital media, the popularity of live 360-degree video has witnessed an upward trajectory, underpinned by its immersive experience. In live 360-degree video streaming services, transcoding enormous video tiles and providing low-latency services for global viewers require significantly more computing and communication resources than traditional live services. Inspired by the huge crowdsourcing computational resources available at crowd, this paper designs CA-Live360, a Crowd-Assisted Live 360-degree video streaming solution, in which transcoding and delivery are supported by crowdsourcing devices. Due to the dynamic availability of resources, it is challenging to achieve low-latency live services, while guaranteeing fair transcoding. Therefore, firstly an innovative fairness-guaranteed transcoding task assignment scheme is proposed. This scheme employs a novel Fair Bandit algorithm based on the combinatorial multi-armed bandit approach to achieve low-latency transcoding while considering time-varying available resources and fairness constraints. Secondly, to reduce delivery latency, we design a transcoding-aware delivery scheme, in which a provider-requester matching algorithm is introduced to realize effective requester scheduling. Real-world trace-driven experiments demonstrate the improved performance of the proposed CA-Live360 solution in terms of system delay and fairness.

Keywords: Live 360-degree video transmission system, task assignment, video delivery, fairness, multi-armed bandit, matching theory.

1. Introduction

1.1. Background

In the new media ecology era, live streaming has become a necessity in our daily life[1]. Concurrently, the metaverse's expansion propels virtual reality (VR) live streaming, observable in the surge of VR live streaming services that offer immersive experiences. VR services are premised on 360-degree video streaming [2], steadily gaining popularity. According to a Statista report[3], the global VR market size is projected to increase from less than 12 billion U.S. dollars in 2022 to more than 22 billion U.S. dollars by 2025.

Due to the diversity of networking conditions and device capabilities, viewers require live videos encoded at different resolutions. Accordingly, live streaming platforms must transcode original video content into multiple versions. Transcoding 360-degree video services, however, demands substantially more computational resources and time than traditional live streaming, due to the large amount of 360-degree video data [4]. Furthermore, the ultra-low-latency (0.2 to 1 second) demand of live streaming [5] brings a greater challenge to panoramic video transcoding. In previous works, cloud-based paradigms are used to transcode live streaming[6–13]. However, they have inherent drawbacks to live 360-degree video streaming: 1) broadcasters and a large number of viewers generate a massive number of 360-degree streaming transcoding tasks, which consumes more computing resources than ordinary video transcoding. It makes the cloud-based transcoding significantly expensive[15]; 2) the cloud server is often geographically remote, which leads to high round-trip latency and affects the Quality of Service (QoS) in the delay-sensitive live streaming services.

In live streaming systems, there are huge crowdsourcing computational resources. Several attempts have been made to employ fog-based transcoding which offloads transcoding tasks to nearby devices, especially to personal devices[15–19], overcoming the disadvantages of cloud-based transcoding. For example, He *et al.* proposed CrowdTranscoding [16], which focused on providing low-latency services by detecting qualified stable transcoders for source channels.

Fortunately, crowd-based computing provides a natural opportunity for distributed tiled 360-degree video transcoding, in which tiles are transcoded in parallel on multiple devices. Dis-

*Corresponding author: Changqiao Xu
Email addresses: myx@bupt.edu.cn (Yunxiao Ma), cqxu@bupt.edu.cn (Changqiao Xu), zhwu@bupt.edu.cn (Zhonghui Wu), 2021140795@bupt.edu.cn (Renjie Ding), xiaohan@bupt.edu.cn (Han Xiao), zhonglj@cnu.edu.cn (Lujie Zhong), 13316094433@chinatelecom.cn (Yirong Zhuang), gabriel.muntean@dcu.ie (Gabriel-Miro Muntean)

tributed transcoding reduces a lot of time consumption than centralized transcoding. Meanwhile, the economic cost of crowd-based computing is lower than cloud-based computing in general. Motivated by the above factors, we proposed to accomplish tiled 360-degree video transcoding through crowd-based distributed computing.

1.2. Problem Statement

In crowd-based distributed transcoding, one crowd consists of a large number of nodes with computing capacity, which are called Computational Nodes (CNs). It is important to note that the computing and communication resource provision of a CN is time-varying and unpredictable. The dynamic environment makes the transcoding efficiency of a CN unstable, which cannot be ignored in a delay-sensitive live 360-degree video streaming system. Besides, the existing articles [15–17, 20] focus on decreasing the streaming latency by detecting and selecting qualified stable devices. However, the researchers ignored the fact that crowd's fairness is a key point to ensuring the healthy and long-term development of the crowdsourcing system. If the system greedily chooses CNs who have the superior transcoding capability to minimize the time cost, it will inevitably lead to ordinary CNs having few chances to participate in transcoding. This situation will seriously impact the satisfaction and participation of most CNs. Therefore, it is indispensable to guarantee CNs selection fairness while pursuing low latency.

In addition, video streaming to viewers must occur after transcoding. Given the large bandwidth requirement of 360-degree video streaming and the heterogeneous and time-varying network status, low-latency video delivery is a formidable challenge. Existing research seldom focuses on the delivery process in crowd-based transcoding systems. Although there are two works[16, 20] that mentioned video delivery briefly in their transcoder selection framework, both of them put the server in charge of video delivery. Specifically, a centralized server is responsible for recollecting transcoded videos and their delivery. This method has a potential problem, which is that the centralized server has to wait for the transcoder to upload the transcoded videos and then their delivery to viewers. The extra transmission delay between the transcoder and server can not be overlooked.

1.3. Contributions

Motivated by the aforementioned aspects, we design CA-Live360, a novel **C**rowd-**A**ssisted **Live 360**-degree Streaming solution, which includes a fair transcoding scheme and a transcoding-aware delivery scheme, proposed to support low-latency live 360-degree video streaming services with fair assistance from the crowd.

To achieve the fairness-guaranteed transcoding task assignment, there are two main challenges to overcome: 1) achieving the minimum delay with no prior information in a dynamic environment; 2) striking a balance between fairness and latency. Related to **the first challenge**, since there is no information on crowd and the network condition is dynamic and unpredictable, most traditional optimization methods (e.g., convex optimization) become infeasible. Reinforcement learning is a possible method, however, it has some problems, such as high computational complexity, unstable convergence rate, etc. In this situation, we consider the multi-armed bandit technique, which is a promising solution for continuous decision-making problems. It learns information about CNs in the crowd by interacting with the environment and makes decisions based on this knowledge. To address **the second challenge**, we have to solve two problems: capture the dynamics of fairness, and strike a balance. A fairness queue mechanism is designed considering the evolving nature of fairness. We then utilize a fair bandit algorithm, balancing delay and fairness based on the fairness queue.

In CA-Live360, transcoders are geographically located close to massive viewers. In this context, transcoders are the best content providers due to their location superiority. Therefore, to take full advantage of the crowd-assisted transcoding, we design a transcoding-aware live streaming delivery scheme, where the requesters will get live streaming from transcoders in priority. If a requester can not get the desired video tiles from any transcoder quickly, it will be scheduled to the Base Station (BS). To tackle the requester scheduling problem in the delivery process, we proposed a provider-requester matching algorithm, which assigns each requester to the appropriate providers. Part of this work has been published in [21].

The main contributions are summarized as follows:

- *A crowd-assisted live 360-degree video streaming solution (CA-Live360).* We designed CA-Live360 to provide low-latency 360-degree live streaming services with assistance from crowd. It consists of a crowd-based distributed tile transcoding process and a tile delivery process.
- *Fairness-guaranteed transcoding task assignment scheme.* To meet the fairness constraints in transcoding task assignment, a **F**air **B**andit (FB) algorithm is designed to strike a balance between latency and fairness in a dynamic environment. To the best of our knowledge, this is the first work to consider fairness in crowd-assisted transcoding.
- *Transcoding-aware live streaming delivery scheme.* To capitalize on the transcoder's proximity to viewers, we proposed transcoder-aware streaming delivery. To schedule requesters, we proposed a **P**rovider-**R**equester **M**atching (PRM) algorithm based on matching theory. The stability and complexity of PRM are analyzed.
- *A series of trace-driven experiments* are conducted to evaluate the performance of the proposed CA-Live360 based on a real-world dataset. The experimental results show CA-Live360's superiority to state-of-art schemes in terms of delay and fairness.

The rest of this paper is organized as follows. In Section 2, we introduced the background. The system model of the proposed CA-Live360 and problem formulation are presented in Section 3. The fairness-guaranteed transcoding task assignment is introduced in Section 4, followed by the transcoding-

aware delivery in Section 5. The experimental results of CA-Live360 are presented in Section 6. Finally, conclusions and future works are included in Section 7.

2. Background and Related Works

2.1. Video transcoding

In mainstream live streaming platforms such as Twitch, Douyu, etc., cloud computing is widely applied, as large numbers of stable computing resources are at their disposal. In academia, a number of scholars were attracted to study live video transcoding with cloud computing [6–9] in the last decade. Authors of [8] try to solve the problems of cost-effective adaptive live video streaming, by making real-time transcoding and offloading decisions dynamically in cloud servers. Xiangbo Li et al. [7] designed a model to capture the trade-off between the cost and performance of cloud servers to maximize the usage of cloud computing resources. However, the latency caused by content delivery from remote cloud servers affects viewers' quality of experience (QoE), especially with increasing user demands. Utilizing computing resources in edge servers (i.e. edge computing) has gained momentum [10–14]. Authors of [11] [12] combined edge computing with the cloud, and proposed hybrid architectures of cloud-edge computing. By comprehensively considering caching, transcoding, and resource allocation, an edge-assisted live stream processing framework was proposed in [13]. It efficiently improved viewers' throughput, video quality and decreased the playback rebuffering time at the same time. However, it should be noted that renting computing resources for transcoding from cloud servers or edge servers is very expensive, and most live streaming platforms cannot afford it.

Moreover, as the computing and storage capacity of end devices grows, many researchers have turned their attention to studying crowd-based 2-D video transcoding[15–17, 20] in live streaming. Yifei Zhu et al. in [15] proposed an auction-based selection to choose suitable viewers for computational task offloading. In [20], authors considered a two-stage decision approach based on deep reinforcement learning to select transcoders. A large number of crowdsourced computing resources enable support for panoramic video transcoding. For a chunk of 360-degree video, different tiles can be transcoded in parallel on different computational devices. Thus, crowd-based distributed tile transcoding is a promising scheme to decrease the latency of services. However, there are still some problems to be solved, including aspects such as the fairness between crowd, network dynamics, etc.

2.2. Video delivery

Meanwhile, video delivery is also a key process in live streaming system. We have researched several issues about video delivery [18, 23–26]. By designing an augmented graph model, the joint optimization of transcoding and transmission in live streaming systems is converted into a multi-hop routing problem, which is then solved by a Networked Multi-Agent Reinforcement Learning approach in [23]. Considering the ultra-high data rate and ultra-low-delay demands of VR livecast, we proposed a buffer-nadir-based multicast mechanism in [24], as well as designed a multicast-aware transcoding offloading algorithm and a delivery algorithm to achieve cost-efficient virtual reality delivery. Leveraging the virtual queue technology, an augmented queue structure is designed in [18] to jointly capture the dynamic characteristic of transmission and transcoding. Then we introduced an accelerated gradient optimization algorithm to solve the formulated resource optimization problem. In addition to our previous work above, some other scholars have also done related research [27–34]. [27] proposed a multi-MEC cooperative caching architecture and a transmission mechanism for VR video. In [28], a transcoding-enable video delivery and caching scheme was proposed to meet the high-bandwidth and low-latency requirements of VR. [29] introduced transcoding-enabled multicast opportunities, in which the multicast decision is made in the transcoding-enabled situations. Authors of [30] proposed PVRV, a streaming system for panoramic video based on millimeter wave with mobile edge computing. The transcoding in PVRV was performed at the edge servers. VR video streaming in multiple-input multiple-output-orthogonal frequency division multiple access system was researched in [31]. To minimize the transcoding and transmission power, the authors jointly considered video quality selection, beamforming, transmission power, and rate allocation. In article [32], its authors focused on the deadline-sensitive delivery problem for 360-degree video streaming. To handle concurrent requests and dynamic bandwidth, they formulated the content as a long-term integer programming optimization problem, and then they proposed a solution based on Deep Reinforcement Learning and Cooperative Bargaining Game to address it. [33] presented a cell-free multi-group broadcast network solution to deliver VR video from UAVs to VR users in sports stadiums. The authors tackled the problem of dynamic video scheduling and access-point clustering through a series of deep reinforcement learning approaches. Authors in [34] proposed a proactive wireless multicasting framework that clustered users based on FoV overlap and location, and assigned them to small base stations for efficient VR video delivery. However, the deep involvement of crowd in 360-degree video transcoding and delivery has not been studied.

3. System Model and Problem Formulation

This section presents the system model of CA-Live360, and introduces the problem formulation in detail. The mathematical notations for the paper has been summarized in Table 1.

3.1. Overview of the Proposed CA-Live360

Fig.1 illustrates the proposed crowd-assisted live 360-degree video streaming system. Broadcaster is the streaming provider which uploads live 360-degree videos. BS is responsible for managing transcoding tasks and providing services to requesters. Thanks to the boom in FoV predictive technology[35, 36], we adopt the tile-based viewport adaptive streaming scheme[37–39], which can significantly reduce bandwidth consumption in the transmission process of 360-degree video. It

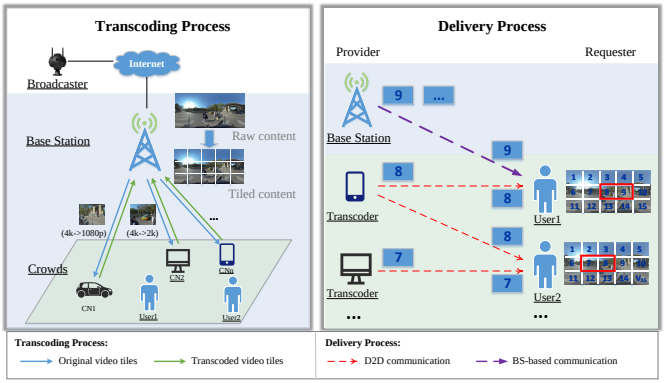

Figure 1: Crowd-assisted live 360-degree video streaming system.

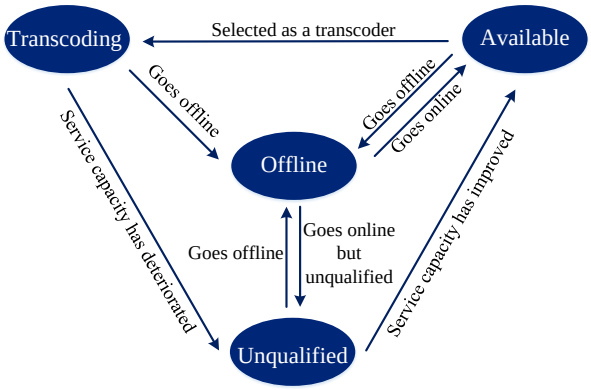

Figure 2: The transition of four states of CNs.

divides panoramic video into tiles, which can be independently codec. In the scheme, BS transmits the tiles in the user's FoV at an appropriate high resolution, while the tiles out of FoV at a low resolution. The transmission of tiles out of FoV is to guarantee that users can watch the video with basic quality when FoV prediction is accidentally inaccurate. Thus, the tiles in FoV have a higher priority than tiles out of FoV. Based on the above observation, we leverage the computing resources of crowd to transcode the tiles in FoV and assign the transcoding tasks of the tiles out of FoV to BS. In this case, we can focus on the strength of crowd to transcode the tiles in FoV, to provide high-quality services with low latency to users. In this work, we focus on the transcoding and delivery of FoV tiles.

We assume that all CNs are willing to participate in the transcoding tasks motivated by some incentive mechanisms such as advertising exemptions or monetary gains [16]. In CA-Live360, BS will assign CNs in crowd to transcode the original tiles to different resolutions; these CNs are called *Transcoders*. For a channel, we assume that there are K CNs who may perform transcoding tasks in the region. We define $\mathcal{K}$ as the set of all CNs, and $i \in \mathcal{K}$ is one of the K CNs. In this crowd-assisted streaming system, each CN in crowd can be in one of the following states in time slot t:

- *Offline:* The CN is offline or out of the region, and could not perform any transcoding task.
- *Unqualified:* The CN cannot participate in transcoding because its service capabilities (i.e., computational or communications) do not meet the minimum standards.
- *Transcoding:* The CN is performing transcoding tasks.
- *Available:* The CN is qualified and has not been assigned a transcoding task.

$\mathcal{O}(t), \mathcal{U}(t), \mathcal{T}(t), \mathcal{A}(t)$ are used to denote the sets of CNs in *Offline*, *Unqualified*, *Transcoding* and *Available* state in time slot t, respectively. Every CN switches between these states over time. The transition of the four states is shown in Fig. 2.

Viewers of live 360-degree video are in the crowd, and we call them *Requesters*. The set of requesters is represented as $\mathcal{R}(t) \subseteq \mathcal{K}$. The requesters in $\mathcal{T}(t)$ can play the received high-resolution videos or their own transcoded tiles directly, but other requesters have to be served with transcoded tiles by a provider. As shown in the transcoding process of Fig.1, tiles with different resolutions will be aggregated to BS after transcoding. Then, both BS and transcoders have the transcoded tiles. Thus, in the delivery process of Fig.1, there are two types of providers for requesters to get live video: transcoders and BS.

- Transcoder: if there is a transcoder that outputs the resolution requested, the requester can get the tile from the transcoder.
- BS: if a requester can not be served by any transcoder, it will stream from BS.

It should be noted that the resolution requested is determined by Adaptive Bitrate Streaming algorithm [40], which is not the focus of this work.

3.2. Transcoding Model

We consider that live 360-degree streaming consists of continuous video chunks and each chunk is a 1-second block. We segment each chunk into $X \times Y$ tiles. For the tiles in FoV, we need to prepare M resolutions to meet the different requirements of heterogeneous requesters, and $\Phi = \{q_1, q_2, ..., q_M\}$ denotes the set of resolutions. Based on FoV prediction, viewer n will request tiles set $\mathcal{G}_n$. The number of FoV tiles that will be requested by all requesters is G. Thus, the total number of transcoding tasks for FOV tiles is $M \cdot G$. We use $\mathbb{G}$ to denote the set of all $M \cdot G$ transcoded tiles with different resolutions. Due to the service quality requirement, we assume the number of CNs that need to work for each task simultaneously is no less than m ($m > 0$). Thus, the total number of transcoders in each time slot is $U = M \cdot G \cdot m$.

In real scenarios, CNs may have other computing tasks, so the available computing resources are independent and identically distributed random processes. We use $C_i(t)$ to represent the available computing resources of CN i, which is within $[C_i^{min}, C_i^{max}]$ with $\mathbb{E}[C_i(t)] = \mu_i^C$. Due to the heterogeneity of CN, we assume μ_i^C follows a normal distribution with $\mathbb{E}\left[\mu_i^C\right] = \lambda^C$.

Table 1: Mathematical Notations

Symbol	Description
$\mathcal{K}$	The set of all CNs
$\mathcal{O}, \mathcal{U}, \mathcal{T}, \mathcal{A}$	The set of CNs in *Offline*, *Unqualified*, *Transcoding*, and *Available* state
$\mathcal{R}, \mathcal{P}$	The set of requesters, the set of providers
$\mathcal{S}$	The set of reassigned CNs
$C_i, \mathcal{B}_i$	Available computing, bandwidth resources of CN i
T_i^{CN}	The transcoding delay of CN i
$T^{BS}(t)$	The transcoding delay of BS
d^{re}	The delay of reassignment
$\mathbb{T}^{tp}$	The delay in transcoding process
$\mathbb{T}_n^{dp}$	The delay in delivery process
$T_{s,n}^{TD}$	The transcoder-based delivery delay
T_n^{BSD}	The BS-based delivery delay
$c_i(T)$	The total online time of CN i by time T
$f_i(T)$	The selection fraction of CN i by time T
x_i	RMSF of CN i
$\kappa_i(t)$	Transcoder cost evaluation
$r_i(t)$	The reward for player i in t
$h_i(t)$	The number of working times for CN i by t
$\bar{\mu}_i(t)$	The average reward for CN i by t
$\mu_i^{UCB}(t)$	The reward evaluation according to UCB policy
Q_i	The fairness queue length of CN i
$\mathbb{R}_j, \mathbb{P}_j$	The set of requester, the set of provider for tile j
Φ, $q*$	A matching, the quota of a matching

We assume that CN i is assigned to transcode the original video tiles j to resolution $q_{i,j}(t)$ with bit-rate $b_{i,j}(t)$, where $q_{i,j}(t) \in \Phi$. We define $\Lambda\left(q_{i,j}(t), b_{i,j}(t)\right)$ as the number of CPU cycles required to transcode an original tile to the target resolution and bit-rate. The measurement of computational resources consumed for transcoding is based on [41]. Consequently the transcoding delay of CN i can be expressed as:

$$T_i^{CN}(t) = \frac{\Lambda\left(q_{i,j}(t), b_{i,j}(t)\right)}{C_i(t)}, \forall i \in \mathcal{T}(t). \tag{1}$$

To ensure a smooth live streaming service, transcoding is done by BS when there are not enough qualified CNs. The delay generated by BS transcoding process is denoted by $T^{BS}(t)$. Apart from the time associated with the transcoding process, stability [16] is also a key point in such a system. This is because a frequent offline status of unstable transcoders determines extra reassignment delays, which affect negatively live streaming services. To consider these two factors jointly, the delay in the transcoding process is computed as follows:

$$\mathbb{T}^{tp}(t) = \begin{cases} T_i^{CN}(t) + I_i(t) \cdot d^{re} & \text{transcoding at CN } i, \\ T^{BS}(t) & \text{BS transcoding.} \end{cases} \tag{2}$$

Here, d^{re} is the reassignment delay. $I_i(t)$ is an indicative function and if CN i is reassigned in time slot t, $I_i(t) = 1$, otherwise, $I_i(t) = 0$.

3.3. Transcoding-aware Delivery Model

After transcoding, video tiles are delivered to requesters. Let $\mathcal{P}(t)$ denote the set of providers in time slot t. It is expressed as:

$$\mathcal{P}(t) = \begin{cases} \{s_0, s_1, s_2, ...s_U\} & \text{crowd transcoding,} \\ s_0 & \text{BS transcoding,} \end{cases} \tag{3}$$

where s_0 is BS and $\{s_1, s_2, ...s_U\}$ are U transcoders. The above equation indicates that if transcoding at a CN, both BS and transcoder have the transcoded video. If transcoding is performed in BS, only BS has the video.

With different types of providers, there are two delivery modes: **Transcoder-based Delivery** and **BS-based Delivery**. Transcoders are geographically closer to requesters, resulting in lower transmission latency for transcoder-based delivery. Thus, requesters are preferentially served by transcoders. When the requester can not get streaming from any transcoder, it will be served by BS.

Transcoder-based Delivery: Tiles delivery from transcoder s to requester n is based on device-to-device (D2D) communication[42]. The transmission rate $R_{s,n}$ is constrained by the weakest device with the poorest bandwidth resource. According to the Shannon–Hartley theorem[43], the communication rate is:

$$R_{s,n}^{TD}(t) = \min_{i \in \{s,n\}} \mathcal{B}_i(t) \log_2\left(1 + \frac{p_i \cdot g_{s,n}}{\sigma^2}\right), \forall s \in \mathcal{P}(t), n \in \mathcal{R}(t), \tag{4}$$

where p_i is the communication power of node i, $g_{s,n}$ is the channel gain between provider s and requester n, and σ^2 is the power of **a**dditive **w**hite **G**aussian **n**oise (AWGN). $\mathcal{B}_i(t)$ denotes the bandwidth resource of CN i in time slot t. Similar to computing resources, $\mathcal{B}_i(t)$ is independent and identically distributed over time and is within $[\mathcal{B}_i^{min}, \mathcal{B}_i^{max}]$ with $\mathbb{E}[\mathcal{B}_i(t)] = \mu_i^{\mathcal{B}}$, and $\mu_i^{\mathcal{B}}$ following a normal distribution with $\mathbb{E}\left[\mu_i^{\mathcal{B}}\right] = \lambda^{\mathcal{B}}$. The time consumed in the transcoder-based delivery process can be expressed as:

$$T_{s,n,j}^{TD}(t) = I_{s,n}^{TD}(t) \cdot \frac{e_j(t)}{R_{s,n}^{TD}(t)}, \forall s \in \mathcal{P}(t), n \in \mathcal{R}(t), \tag{5}$$

where $e_j(t)$ denotes the amount of tile j data that transcoder s transmits to n, and $I_{s,n}^{TD}(t)$ indicates the requester scheduling result. When requester n is served by transcoder s, $I_{s,n}^{TD}(t) = 1$ and otherwise, $I_{s,n}^{TD}(t) = 0$.

BS-based Delivery: If there is no transcoder that can provide a certain tile j for requester n, it will be served by BS. The delay in the BS-based delivery process is computed by the following equation:

$$T_{n,j}^{BSD}(t) = I^m(t) T_j^{up} + T_{n,j}^{down}(t), \tag{6}$$

where $I^m(t)$ indicates the transcoding mode in time slot t. If transcoding is performed at crowd, $I^m(t) = 1$, otherwise, $I^m(t) = 0$. T_j^{up} is the minimum uploading delay of tile j from CNs to BS, and $T_{n,j}^{down}(t)$ is the delivery delay from BS to requester n.

(6) shows that if a CN transcodes, but the requester can not get the video streaming from this transcoder directly, the content acquisition delay will include two parts: T_j^{up} and $T_{n,j}^{down}(t)$. If BS transcoding, the delay is equal to $T_{n,j}^{down}(t)$.

To sum up the above, for tile j that requested by $n, \forall n \in \mathcal{R}$, the delay in delivery process can be expressed as follows:

$$\mathbb{T}_{n,j}^{dp}(t) = \begin{cases} T_{s,n,j}^{TD}(t), & \text{transcoder-based delivery,} \\ T_{n,j}^{BSD}(t), & \text{BS-based delivery,} \\ 0, & \text{requester } n \text{ is a transcoder.} \end{cases} \tag{7}$$

For 360-degree video viewers, they have to receive all tiles in FoV before they can play the video. So we use $\mathbb{T}_n^{dp}(t) = \max\{\mathbb{T}_{n,j}^{dp}(t), \forall j \in \mathcal{G}_n\}$ denote the delivery delay of the last tile received in a chunk for requester n.

3.4. Problem Formulation

Constraints: In a crowd-assisted live streaming system, the transcoding and delivery processes have to comply with the following constraints to ensure a great performance:

- **Fairness Constraint.** As discussed in Section I, the fairness of crowd is extremely important for CA-Live360. In order to evaluate fairness, a new fairness-related metric *selection fraction* is introduced, defined as follows:

$$f_i(T) = \frac{1}{c_i(T)} \sum_{t=0}^{T-1} l_i(t), \forall i \in \mathcal{K}, \tag{8}$$

  where $c_i(T)$ is the total online time of CN i by T. $l_i(t) \in \{0, 1\}$, and $l_i(t) = 1$ indicates that CN i performed transcoding task in time slot t, otherwise, i did not transcode in time slot t.

  The initial value is set to 0: $l_i(0) = 0$. A large selection fraction $f_i(T)$ indicates that CN i was selected much, relative to its online history. To guarantee viewer fairness, we introduce a **R**equired **M**inimum **S**election **F**raction (RMSF) for viewer i as follows:

$$\lim_{T\to\infty} \inf f_i(T) \geq x_i, \forall i \in \mathcal{K}. \tag{9}$$

  In (9), $x_i \in (0, 1)$ is RMSF of CN i, ensuring that no CN is overlooked in the system. In fact, x_i can be adjusted flexibly. For example, premium members in the crowdsourcing platform can be given a larger RMSF than regular members.

- **QoS Constraint.** Different from other task assignment scenarios, live streaming services are delay-sensitive to support real-time interaction. If transcoding and delivery take a long time, the QoS is seriously affected. Therefore, in the proposed scheme, the transcoding and delivery time must be within the specified range to ensure that the latency does not affect the quality of the live-streaming services.

$$\mathbb{T}^{tp}(t) < \mathbb{T}_{Max}^{tp}, \tag{10}$$

$$\mathbb{T}_n^{dp}(t) < \mathbb{T}_{Max}^{dp}, \quad \forall n \in \mathcal{R}(t), \tag{11}$$

  where $\mathbb{T}_{Max}^{tp}$ and $\mathbb{T}_{Max}^{dp}$ are the maximum tolerable transcoding and delivery delay, respectively, determined by the practical requirements in a live streaming system.

Objective: In CA-Live360, the system delay consists of two parts: transcoding delay and delivery delay. The goal of the solution is to minimize the system delay while satisfying the above constraints. Thus, the crowd-assisted transcoding and delivery are formulated as the following problem:

$$\begin{aligned} \min \quad & \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \frac{1}{|\mathcal{R}(t)|} \sum_{n\in\mathcal{R}(t)} \mathbb{T}^{tp}(t) + \mathbb{T}_n^{dp}(t), \\ \text{s. t.} \quad & (9)(10)(11). \end{aligned} \tag{12}$$

Here, $|\mathcal{R}(t)|$ represents the cardinality of the set $\mathcal{R}(t)$. In the problem, we try to reduce the transcoding and delivery delay while considering the fairness and QoS constraints. Intuitively, the problem can be solved in two phases: transcoding and delivery according to (12). In this context, we propose solutions for the two phases in Section 4 and Section 5, respectively. The overview of the two phases is shown in Fig.3.

4. Fairness-guaranteed Transcoding Task Assignment

In order to select high-quality transcoders to achieve the overall optimization goal, we first introduce a criterion to evaluate the transcoders. According to the analysis made in Section 3.2, the transcoders which transcode quickly and stay online for a long time to avoid reassignment delays are preferred. Besides, we also consider the influence of transcoders on the delivery process. If the selected transcoder has great transmission capacity, the delay in the transcoder-based delivery process is small. The maximum transmission capacity of CN is $\mathcal{B}_i(t)\log_2\left(1 + \frac{p_i \cdot g_i}{\sigma^2}\right)$. So for CN i, the minimum transmission delay per unit of data is expressed as:

$$T_i^{min}(t) = \frac{1}{\mathcal{B}_i(t)\log_2\left(1 + \frac{p_i \cdot g_i}{\sigma^2}\right)}. \tag{13}$$

Based on this analysis, we design the transcoder evaluation criterion, cost $\kappa_i(t)$, as follows:

$$\kappa_i(t) = \gamma\mathbb{T}^{tp}(t) + (1 - \gamma)T_i^{min}(t), \tag{14}$$

where $\gamma \in (0, 1)$ is a balance factor. In the transcoding task assignment phase, if the system is focused on transcoding performance, set $\gamma > 0.5$; otherwise, $\gamma < 0.5$. When $\gamma = 0.5$, the transcoding and delivery processes are of equal importance. Overall, a CN with a small cost κ_i indicates stability and has adequate computing and bandwidth resources, so it is preferred in the transcoding task assignment process. However, BS does not have any prior knowledge about viewers. In order to assign transcoding tasks effectively with the time-varying available resources and network conditions, we design a multi-armed bandit-based algorithm and introduce a fairness queue to deal with the fairness constraint next.

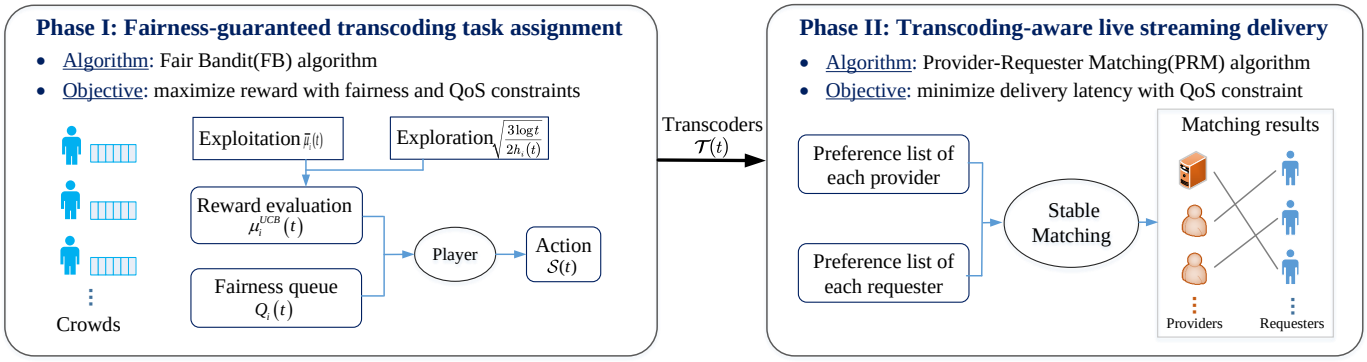

Figure 3: Overview of the two phases in CA-Live360: fairness-guaranteed transcoding task assignment, and transcoding-aware live streaming delivery

4.1. Combinatorial Multi-Armed Bandit Problem

The multi-armed bandit problem refers to the case when a choice must be made between multiple actions, each with an unknown payout (i.e. slot machines are considered "one arm bandits") in order to achieve the most profitable outcome through a series of choices. In the Combinatorial Multi-Armed Bandit (CMAB), each "one arm bandit" is associated with a stochastic reward $r_i \in [0, 1]$ that is independently identically distributed over time with an unknown mean. A player pulls multiple arms and obtains rewards from them. The player's goal is to learn an arm selection policy to maximize the time-average reward following repeated interactions with the stochastic environment.

In this work, $\kappa_i(t)$ is unknown a priori to BS, which is consistent with CMAB. Each CN is regarded as an arm and BS is a player in CMAB. The reward of a CN is independent of other CNs. Furthermore, as the goal is to decrease $\kappa_i(t)$, the reward is defined as:

$$r_i(t) = \begin{cases} \frac{1}{\kappa_i(t)} & if \quad i \in \mathcal{T}(t), \\ 0 & else. \end{cases} \tag{15}$$

In (15), if $i \in \mathcal{T}(t)$, $\kappa_i(t) \neq 0$ for anytime. Accordingly, the expected time-average reward is given by:

$$R(T) = \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{|\mathcal{T}(t)|}\sum_{i\in\mathcal{T}(t)}\frac{1}{\kappa_i(t)}\right]. \tag{16}$$

We define R^* as the maximal time-average reward with the optimal policy. Then the regret is expressed as:

$$\Xi(T) = R^* - R(T). \tag{17}$$

Since R^* is a constant, maximizing the reward $R(T)$ is equal to finding a solution to minimize the regret.

4.2. Upper Confidence Bound Policy

As the reward of each CN is unknown in advance, the player has to strike a balance between *exploration* (i.e. selecting different CNs to learn a more accurate estimation of the mean reward of each CN) and *exploitation* (i.e. maximizing the reward by selecting CNs having shown the best rewards in the past). To evaluate the reward, an upper confidence bound policy (UCB) [44] is employed and described as follows.

$h_i(t)$ is used to represent the total number of times CN i has worked by the current time slot t, i.e.

$$h_i(t) = \sum_{n=0}^{t-1} l_i(n), \tag{18}$$

where the initial value $h_i(0) = 0$.

Let $\bar{\mu}_i(t)$ denote the average reward of CN i before time slot t, which is defined as:

$$\bar{\mu}_i(t) = \begin{cases} \frac{\sum_{n=0}^{t-1} r_i(n) l_i(n)}{h_i(t)} & h_i(t) \neq 0, \\ r_i^0 & h_i(t) = 0, \end{cases} \tag{19}$$

where $r_i^0 \in (0, 1)$ is the initial value.

The reward evaluation according to the UCB policy is expressed as:

$$\mu_i^{UCB}(t) = \begin{cases} \min\left\{\bar{\mu}_i(t) + \sqrt{\frac{3\log t}{2h_i(t)}}, 1\right\}, & h_i(t) \neq 0, \\ \bar{\mu}_i(t), & h_i(t) = 0. \end{cases} \tag{20}$$

In (20), $\bar{\mu}_i(t)$ corresponds to *exploitation* and $\sqrt{\frac{3\log t}{2h_i(t)}}$ represents *exploration*. The player selects the arm with the largest $\mu_i^{UCB}(t)$ in each time slot. This means that the arms which have great historical average rewards and have been played rarely are selected. As the number of iterations increases, the estimated mean reward of each viewer becomes more accurate.

4.3. Fairness Queue

Intuitively, the classic UCB policy performs poorly on the fairness-guaranteed task assignment scheme because it ignores the constraints from (9),(10),(11) in the formulated problem. Especially, the fairness constraint greatly increases the difficulty of solving the problem, because we not only have to seek a balance between exploration and exploitation but also have to achieve a new compromise between reward and fairness. To address this, a virtual queue[45] is designed to handle the fairness constraint.

We assume that each viewer maintains a virtual fairness queue. The current queue length of CN i is $Q_i(t)$. In particular, the fairness queue length dynamic is designed as:

$$Q_i(t) = \begin{cases} Q_i(t-1) & i \in \mathcal{O}(t), \\ [Q_i(t-1) + x_i - l_i(t-1)]^+ & else. \end{cases} \quad (21)$$

Here, $[x]^+ = \max(x, 0)$. As defined in (21), if the state of CN i is *Offline*, the virtual queue length remains the same as in the last time slot. Otherwise, the virtual queue length will increase by x_i, and decrease by one if CN i was transcoding in the last time slot. The initial values $Q_i(0) = 0$ and $Q_i(t)$ are not less than 0 all the time.

Algorithm 1: Fair Bandit (FB) Algorithm

Initialize $h_i(0) = 0, l_i(0) = 0, Q_i(0) = 0, \forall\, i \in \mathcal{K}$, Set $t = 1$;
while CA-Live360 system is up and running **do**
  Check the state of each CN and observe $\mathcal{O}(t), \mathcal{T}(t), \mathcal{A}(t)$;
  for each $i \in \mathcal{K}$ **do**
    Update $\mu_i^{UCB}(t)$ according to (20);
    Update $Q_i(t)$ according to (21);
  end for
  Observe $|\mathcal{S}(t)|$ though (22);
  Select viewers $\mathcal{S}$ according to (23);
  Update $\mathcal{O}(t), \mathcal{T}(t), \mathcal{A}(t)$;
  CNs in $\mathcal{T}(t)$ perform transcoding tasks;
  for each $i \in \mathcal{K}$ **do**
    Update $l_i(t)$,$I_i(t)$;
    Update $r_i(t)$ according to (15);
    Update $h_i(t)$ according to (18);
    Update $\bar{\mu}_i(t)$ according to (19);
  end for
  $t = t + 1$;
end while

4.4. Fair Bandit Algorithm

Different from the classic CMAB model, the number of CNs that needs to be selected in each time slot in our problem is not constant. At the very beginning of each time slot t, BS will check the working state of CNs that performed transcoding tasks in time slot $t-1$. In general, there are two situations that need to be dealt with:

- If there are CNs in $\mathcal{T}(t-1)$ that currently cannot satisfy the QoS constraints indicated in (10) and (11), we consider they are unqualified and their states will be transformed from *Transcoding* to *Unqualified*. The number of such CNs is denoted by $g_1(t)$.
- If there are CNs in $\mathcal{T}(t-1)$ going offline in time slot t, their state will be transformed to *Offline* as well, we use $g_2(t)$ to denote the number of these CNs.

In those cases, the number of transcoders in $\mathcal{T}(t)$ is less than the required U, so BS has to reselect CNs from $\mathcal{A}(t)$. The set of CNs reassigned in time slot t is denoted by $\mathcal{S}(t)$. According to the above analysis, we can derive:

$$|\mathcal{S}(t)| = g_1(t) + g_2(t), \quad (22)$$

where $|\mathcal{S}(t)|$ indicates the number of transcoders that have to be selected in the reassignment. Jointly considering the reward and the fairness queue, BS will select transcoders from $\mathcal{A}(t)$ as follows:

$$\mathcal{S}(t) \in argmax \sum_{i \in \mathcal{A}(t)} \left[\eta \cdot \mu_i^{UCB}(t) + Q_i(t)\right], \quad (23)$$

where η is a weight parameter to control the tradeoff between two factors. For a small value of η, the algorithm tends to guarantee fairness and prefers to select CNs who have a large virtual queue length $Q_i(t)$. On the contrary, if η is large, the CNs with a large estimated reward $\mu_i^{UCB}(t)$ will predominate.

Due to the independence of CNs, we can solve (23) by selecting CNs that are the top-$|\mathcal{S}(t)|$ best CNs. For those CNs, the better ones will be assigned to tough tasks (i.e., transcode complex tiles to lower resolutions), to meet the more substantial computing and communication capacity requirements. The proposed Fair Bandit(FB) algorithm is described in algorithm 1 and is deployed as part of the CA-Live360 solution.

4.5. Theoretical Analysis

This subsection performs a theoretical analysis of the proposed FB algorithm in terms of queue stability and upper bound for regret.

Queue Stability: If there exists a transcoding assignment policy where the fairness constraint is satisfied, the RMSF vector $\mathbf{x} = \{x_1, ..., x_K\}$ is feasible. We define maximal feasibility region $\mathcal{X}$ to denote the set of all RMSF vectors. Then, we give the following theorem.

Theorem 1: *The constraint from (9) is satisfied for any vector* $\boldsymbol{x}$ *strictly inside the maximal feasibility region* $\mathcal{X}$*, i.e., the virtual queue mean rate stable under FB algorithm:*

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[Q_i(t)\right] \leq \frac{D_1}{\epsilon} \leq \infty, \quad (24)$$

where $D_1 = \frac{K}{2} + \eta m$*, and* ϵ *is a positive constant satisfying that* $\boldsymbol{x} - \epsilon \boldsymbol{1}$ *(*$\boldsymbol{1}$ *is a K-dimensional all-ones vector) is strictly inside* $\mathcal{X}$.

Proof: We prove the theorem by adopting Lyapunov drift analysis[46]. The detailed proof is provided in Appendix A.

Based on theorem 1, the virtual queue is not only mean rate stable, but also strong stable whenever the RMSF vector $\mathbf{x}$ is strictly inside $\mathcal{X}$. This implies that the fairness constraint in (9) is satisfied in the FB algorithm.

Upper Bound on Regret: We introduce the regret upper bound of the proposed FB algorithm in the following theorem.

Theorem 2: *With the FB algorithm, the time-average regret is upper bounded as follows:*

$$R(T) \leq \frac{K}{2\eta} + \frac{1}{T}\left[2\sqrt{6mK\log T} + (1 + \frac{5\pi^2}{12})K\right] + \sum_{i \in K} p_i \mu_i, \quad (25)$$

where p_i is the offline or unqualified probability of CN i.

Proof: See Appendix B.

There are three terms in the upper bound of regret. The first term $\frac{K}{2\eta}$ is related to η and represents the control of online learning in (23). When η is large, CNs with higher predictive rewards will be preferred, even though their queue length may be larger and thus less fair. This leads to a smaller regret captured in the first term, but it will result in worse fairness. So in real applications, the η depends on the trade-off demand of the system. The second term is attributed to the cost of the exploration and exploitation process. It is of the order $O(\sqrt{(\log T)/T})$, which is consistent with the standard regret in typical MAB[47]. The third term is a constant, which means that the greater the probability of CNs being offline or unqualified, the larger the regret. Thus, in Theorem 2, the upper bound of the regret depends on T and parameter η. When $T \to \infty$, the regret decreases to $K/2\eta + \sum_{i\in K} p_i\mu_i$.

5. Transcoding-aware Live 360-degree video Streaming Delivery

This section discusses the delivery of live 360-degree video tiles from providers to requesters. Given the set of transcoders $\mathcal{T}(t)$, i.e. the output of FB algorithm, we schedule requesters to providers to minimize the delivery delay with individual QoS constraints, expressed as:

$$\begin{aligned} \min \quad & \sum_{n\in\mathcal{R}(t)} \mathbb{T}_n^{dp}(t), \\ \text{s. t.} \quad & \mathbb{T}_n^{dp}(t) < \mathbb{T}_{Max}^{dp}, \quad \forall n \in \mathcal{R}(t). \end{aligned} \tag{26}$$

Benefiting from the crowd-assisted transcoding scheme, the requesters have the opportunity to get transcoded tiles from the close transcoder rather than BS. Therefore, our system has a natural advantage in reducing delivery latency. In the system, the requester scheduling is a key factor that affects delivery performance. However, on the one hand, the requested video tiles are scattered in various providers (including BS and transcoders). On the other hand, the communication capacity of the transcoder is restricted, so that each transcoder can only transmit video streaming to a limited number of requesters in a time slot. Those situations make the requester scheduling a tricky issue. In this work, we take it as a combination problem between requester and provider. To solve it, we design PRM algorithm in this section.

5.1. Matching Problem Formulation

Originally used in economics, matching theory[48] provides a powerful solution to pair participants in two finite and disjoint sets based on each participant's preference. In this work, we use matching game to denote the cooperation between requesters and providers.

The two sides of the matching are requesters $\mathcal{R}$ and providers $\mathcal{P}$ respectively. For each tile, the problem is obviously a one-to-many matching, because a provider can serve multiple requesters and a requester just gets a video tile from at most one provider. Moreover, due to the limited communication capacity, the maximum number of requesters a transcoder can serve is defined as a positive integer q^*, which is called *quota*. It should be noted that the capacity of BS is sufficient to serve all requesters. So, if a requester cannot be served by any transcoder, it will be matched to BS for getting tiles. Thus, the quota for all providers in $\mathcal{P}$ is:

$$q_s^* = \begin{cases} q^*, & \forall s \in \mathcal{P}\backslash s_0, \\ |\mathcal{R}|, & s = s_0. \end{cases} \tag{27}$$

We use a three-tuple $\Phi : \{\mathcal{P}, \mathcal{R}, q^*\}$ to denote a matching. Then, we formally define matching in the following definition.

***Definition 1*:** *Given two finite and disjoint sets $\mathcal{R}$ and $\mathcal{P}$, a quota q^*. Then, for $\forall n \in \mathcal{R}$ and $\forall s \in \mathcal{P}$, a one-to-many matching function Φ satisfies*

- $\Phi(n) \in \mathcal{P}$ *and* $|\Phi(n)| \in \{0, 1\}$*;*
- $\Phi(s) \subseteq \mathcal{R}$ *and* $|\Phi(s)| \leq q_s^*$*;*
- $\Phi(n) = s \leftrightarrow n \in \Phi(s)$.

where $\Phi(s)$ represents partners of provider s in Φ.

In Definition 1, the first condition indicates that each requester can only be matched to one provider in $\mathcal{P}$. The second condition implies that each provider can be matched to at most q^* requesters in $\mathcal{R}$. The last condition illustrates that the matching is bilateral, i.e., requester n is matched to a given provider s if and only if provider s is matched to requester n.

5.2. Provider-Requester Matching (PRM) Algorithm

In the matching game, each participant has a preference list to guide the matching process. We use $\succ_i$ to represent the preference. For example, given requester n and providers s_1, s_2, the preference relationship $s_1 \succ_n s_2$ denotes that requester prefers provider s_1 more than s_2, and is defined as:

$$s_1 \succ_n s_2 \leftrightarrow U_n(s_1) > U_n(s_2), \tag{28}$$

where $U(\cdot)$ is the utility function.

We use $\mathbb{R}_j \subseteq \mathcal{R}$ to denote the set of requesters that watch tile j, and $\mathbb{P}_j \subseteq \mathcal{P}$ to denote the set of provider for tile j. Next, for each tile $j \in \mathbb{G}$, we establish preference lists for requesters and providers in the following.

Preference list of requesters: The preference list indicates the ranking of preferences for providers in requester's connectable range. Based on our goal in (26), we try to reduce the delivery delay of tiles. Thus, requesters prefer to get tiles from providers with low communication delays. The utility function of requester n for tile j can be designed as $U_{n,j}(s) = 1/T_{s,n,j}^{TD}(t), \forall s \in \mathbb{P}_j, \forall n \in \mathbb{R}_j$, which means that the smaller the delivery delay, the larger the utility. Based on the utility function, the preference list $\mathcal{L}(n, j)$ is built by sorting the providers in descending order. Because transcoder-based delivery produces lower latency than BS-based delivery in general, BS will rank lower than any high-quality transcoders in the preference list.

Preference list for providers: Similarly, a provider prefers to provide services to requesters with good communication quality to reduce the delivery delay. Similar to requester's utility function, the utility of s can be expressed as $U_{s,j}(n) = 1/T^{TD}_{s,n,j}(t), \forall n \in \mathbb{R}_j, \forall s \in \mathbb{P}_j$. So the preference list $L(s, j)$ for provider s is also built according to the utility function, following sorting the requesters in descending utility order.

The PRM algorithm is described in detail in Algorithm 2. In the preparation stage, the preference list of each participant is calculated. $\overline{\mathbb{R}}$ is initialized to denote the set of unmatched requesters. In the matching stage, if transcoding is performed at the crowd, i.e., $\left|\mathbb{P}_j\right| > 1$, the transcoder-requester matching happens. First, every requester makes a proposal to the highest transcoder in its list. Then, each transcoder makes a decision based on the requesters' proposals. If the number of current partners plus new proposals does not exceed the quota q^*, the matching is successful. Otherwise, the transcoder chooses the top q^* favorite requesters in its preference list. If transcoding is performed at BS, or at CNs but there are requesters that cannot be served by any transcoder, i.e., $\left|\overline{\mathbb{R}}\right| \neq 0$, we allocate all these requesters to BS.

5.3. Theoretical Analysis

The algorithm is expected to output a stable matching. To define stable matching, the following definition is introduced.

Definition 2: *A requester-provider pair (n, s) in a given matching Φ is a blocking pair if any of the following conditions are met:*

- *$s \succ_n \Phi(n)$, and matching n to s will not go against the quota q^*;*
- *$s \succ_n \Phi(n)$, $n \succ_s \Phi(s)$ and the quota of provider s has been reached.*

Then we have the following definition and theorem:

Definition 3: *The matching Φ is defined as stable if there is no blocking pair.*

Theorem 3: *The output Φ of PRM is a stable matching.*

Proof: We prove Theorem 3 by contradiction. The matching Φ is assumed a non-stable matching. According to Definition 2, at least one blocking pair (n, s) exists. There are two cases: one is that requester n never makes a proposal to transcoder s, and the other is that n has made a proposal but is rejected by s. For the first case, due to the fact that the requester n proposes to a transcoder based on the preference list, there exists $(n, s') \in \Phi$, and the requester n has a higher preference for the transcoder s' than s. So (n, s) is not a blocking pair according to Definition 2, and this case does not hold true. For the second case, it indicates that there exists $(n', s) \in \Phi$. The transcoder s has a higher preference for the requester n' than n, so n is rejected. So (n, s) is not a blocking pair and this case is also invalid. The above analysis of the two cases proves that the matching Φ is stable.

6. Performance Evaluation

This section evaluates the performance of the proposed CA-Live360 solution. First, we introduce the experimental setup.

Algorithm 2: Provider-Requester Matching (PRM) Algorithm

Input: The set of requesters $\mathcal{R}$, the set of transcoders $\mathcal{P}$ and the quota q^*.
Output: matching result Φ.
for each $j \in \mathbb{G}$ **do**
  /*Preparation stage */
  Establish the preference list for each requester $n \in \mathbb{R}_j$ and provider $s \in \mathbb{P}_j$;
  Initialize the set of unmatched requesters $\overline{\mathbb{R}} = \mathbb{R}_j$.
  /*Matching stage */
  if $\left|\mathbb{P}_j\right| > 1$ **then**
    while The preference list of any requester $n \in \overline{\mathbb{R}}$ is not empty **do**
      Every requester $n \in \overline{\mathbb{R}}$ makes a proposal to its favorite transcoder based on the preference list;
      Update $\aleph(s)$, which denote the set of new proposals to transcoder s.
      for each $s \in \mathbb{P}_j$ **do**
        if $|\Phi(s)| + |\aleph(s)| \leq q^*$ **then**
          Matching is successful, update the matching result Φ and $\overline{\mathbb{R}}$;
        else if the ranking of the any proposing n in $\aleph(s)$ is higher than any current partner n' in $\Phi(s)$ **then**
          s accepts the proposal of n and reject n'.
          s and n' remove each other from their preference list.
          Update matching results Φ and $\overline{\mathbb{R}}$.
        else
          s rejects n and holds the current partner n'.
          n removes s from its preference list.
        end if
      end for
    end while
  end if
  if $\left|\overline{\mathbb{R}}\right| \neq 0$ **then**
    Allocate all requesters in $\overline{\mathbb{R}}$ to BS.
  end if
end for

Then we analyze the results in turn for the fairness-guaranteed transcoding task assignment scheme, the transcoding-aware delivery scheme, and the proposed CA-Live360 solution.

6.1. Simulation Setup

We consider a scenario that focuses on one region with an area of 1000×1000 m^2 as illustrated in Fig. 1. There are one BS and 100 CNs, i.e. $\mathcal{K} = 100$. CNs are randomly distributed in the region. The transmission power p_i is set to 100mW for all CNs. Channel gain models in the simulation are based on 3GPP standardization. Noise power σ^2 is -100dBm [49]. The available computing resources $C_i(t)$ obey normal distribution with λ^C = 2.5GHz, and the range of $C_i(t)$ is [0, 5GHz]. The avail-

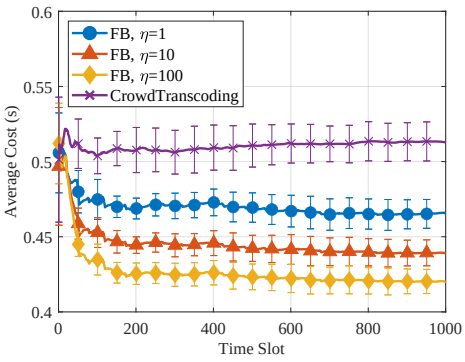

Figure 4: Average cost of FB and CrowdTranscoding.

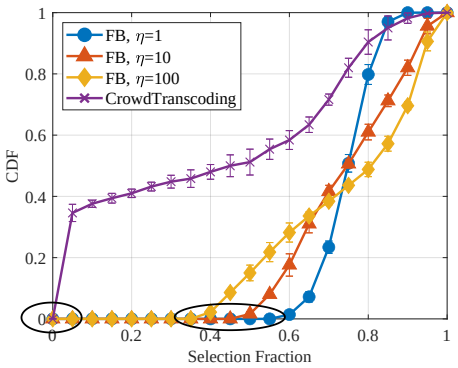

Figure 5: CDF of CNs' selection fraction (RMSF=0.3).

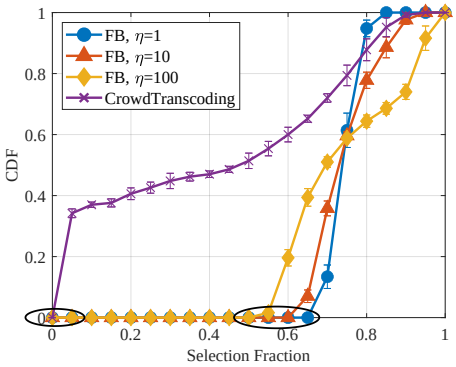

Figure 6: CDF of CNs' selection fraction (RMSF=0.5).

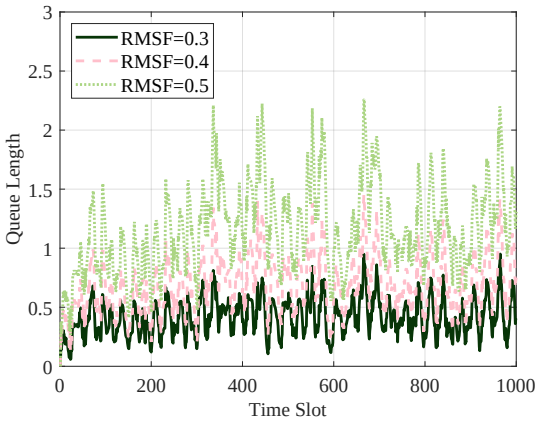

Figure 7: Queue length with different RMSFs.

able communication resources $\mathcal{B}_i(t)$ obey normal distribution with $\lambda^{\mathcal{B}} = 5$MHz, and the range of $\mathcal{B}_i(t)$ is [0, 10MHz][49].

In the transcoding process, the number of CNs that perform transcoding tasks for each resolution is $m = 5$. The reassignment time is set to $d^{re} = 0.3s$, γ is set to 0.5. The offline or unqualified probability of each CN obeys uniform distribution, and the range is $[0.1, 0.3]$. To ensure good QoS, the maximum tolerable transcoding time T^{tp}_{Max} is set to 0.5s. In the delivery process, the number of total requesters for each channel is 30. The maximum tolerable delivery time T^{dp}_{Max} is set to 0.3s.

The 360-degree video in the simulation is based on dataset [50]. We use the 4K (3840×1920 pixels) video "Female Basketball Match" in the dataset as the original video. Then we randomly select 30 users' traces when watching this video in dataset [50] to simulate viewers' FoV. The 360-degree video chunks are segmented into 4×4 tiles, so the resolution of each original tile is 960×480. We use two most popular VR resolutions $\{2560 \times 1440, 1920 \times 1080\}$ as target resolutions, and the resolution of tile is 640×360, and 480×270 respectively. Selected CNs need to transcode original tiles to target resolutions. The corresponding computing resources consumption are based on [41]. Based on above settings, a series of experiments were carried out. We conduct five independent executions of each simulation to obtain more robust data and mean values of these results with confidence intervals (confidence level is set to 95%) are shown in the next subsection.

6.2. *Performance of the Transcoding Task Assignment*

In the experiment for the transcoding task assignment assessment, we focus on four aspects: system cost, fairness of CNs, queue stability and scalability. To evaluate the performance of our proposed scheme, we compare it with the CrowdTranscoding proposed in [16]. We set $\eta \in \{1, 10, 100\}$ in FB, simulation time $T = 1 \times 10^3$ and the RMSF $x_i = 0.3$ for every CN.

Cost: It is defined as the average of all CNs' cost $\kappa_i(t)$. Fig.4 demonstrates the cost comparison of our proposed transcoding task assignment scheme and the baseline method. In all cases, the cost reaches a relatively stable value finally. We observe that the cost of CrowdTranscoding is the highest among all the curves. This is because CrowdTranscoding focuses on the stability of CNs while not considering the dynamics of computing and communication resources. The proposed method, when $\eta = 1$, focuses too much on fairness and results in a higher cost than when $\eta = 10$ or $\eta = 100$. The impact of η is consistent with what was expected. In practice, η can be adapted to application requirements.

Fairness: We use the **C**umulative **D**istribution **F**unction (CDF) of CNs' selection fraction to demonstrate the fairness of the system. As shown in Fig.5, the three curves of the proposed method with different η all satisfy the RMSF (0.3), while CrowdTranscoding cannot guarantee fairness. For CrowdTranscoding, the selection fractions of about 38% CNs are only 0.05, which means that those CNs are rarely chosen. For FB, with different η, the curves vary. It is found that the actual minimum selection fraction is small when η is large. The reason is that the FB with large η gives preference to the CNs that have lower costs.

Then, we set the RMSF to 0.5 and the results are shown in Fig.6. The assignment in CrowdTranscoding is again not fair. The proposed FB with different η ensures selection fractions of all CNs larger than 0.5. These results show that our FB algorithm works well for different RMSFs.

The experimental results presented demonstrate the proposed transcoding task assignment scheme's superior performance to that of an alternative solution [16] and confirm that our method with appropriate weight parameters can guarantee CNs' fairness while achieving lower costs.

Queue Stability: Fig.7 shows the average queue length with different RMSFs. We observe that all three curves oscillate

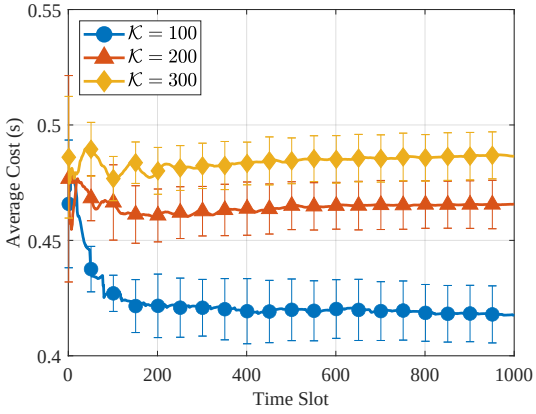

Figure 8: Average cost with different number of CNs.

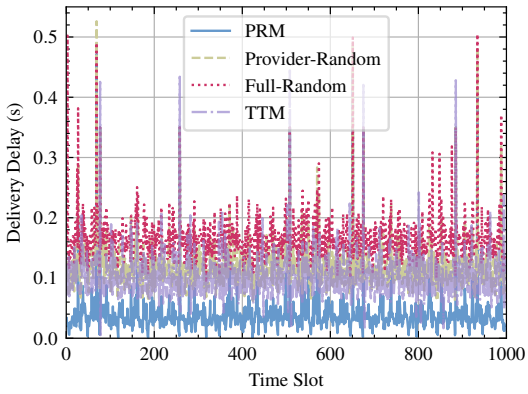

Figure 9: Delivery delay of different schemes.

within fixed ranges, which demonstrates the stability of the designed virtual queue. But the stable ranges of different RMSFs vary. This is because CNs with large RMSFs (x_i) will generate longer queue according to (21) and CNs with large RMSFs have more chances to be selected according to (23). However, as already mentioned, there is flexibility to set different RMSFs for different CNs, so that the benefit would be as desired (e.g. premium CNs get higher gains).

Scalability: To evaluate the scalability of the proposed scheme, we measure the cost with a variable number of CNs and set $\eta = 10$, as shown in Fig.8. In the figure, the three curves tend to become stable over time, which demonstrates that our solution will not be influenced negatively when the number of CNs changes dynamically.

6.3. Performance of Transcoding-aware Live Stream Delivery

To evaluate the performance of the proposed PRM algorithm, we compare it to other two methods:

- **Full-Random**: In this scheme, requesters, and providers are matched randomly;
- **Provider-Random**: Requesters send proposals to providers according to their preferences. Each provider just accepts q^* requesters randomly.
- **TTM** [28]: This is a Two-Tier Matching (TTM) algorithm. In the first tier of TTM, matching occurs between the base station and users. In the second tier, the transcoders are matched with users.

To show specific details regarding the data, the results of one simulation execution are presented in Fig.9. As expected, the Full-Random scheme performed the worst. It produces the largest delay, and the delay fluctuates more. The delivery delay in Provider-Random scheme is better than Full-Random but worse than the proposed PRM algorithm. The is because in the Provider-Random scheme, the requesters send proposals to providers based on their preferences, but the providers just accept requesters randomly. When the proposed PRM algorithm is employed, the provider selects requesters based on its preferences, so it achieves the best performance result. TTM algorithm matches BS and users first, so that some users who would have been able to access video content with lower latency through transcoders are served by base stations. Therefore the delay of TTM is higher than PRM. In addition, note that there are some time slots (such as 259, 509, 652), when the delivery delay is high. This occurs as, during these slots, the system performs base station transcoding, introducing high delivery delay for all requesters.

Furthermore, we selected arbitrarily three viewers to analyze their delivery delay in each time slot. Fig.10 shows the delivery delay within 200-time slots. When the delay equals zero, the viewer is offline or transcoding state because there is no delivery process for them. If the delay is relatively large, BS-based delivery occurs in that time slot. In contrast, if the delay is relatively small, the requester gets video streaming by transcoder-based delivery.

6.4. Performance of CA-Live360

The proposed CA-Live360 includes transcoding and delivery process. Considering the fact that the goal of the CA-Live360 is to provide low-delay services, we use system delay, which is the sum of transcoding delay and delivery delay, to measure the performance. In order to show its effectiveness, we compare CA-Live360 with two baseline schemes:

- **Cloud**: In this scheme, transcoding is done at a cloud server, then the transcoded stream is transmitted to BS and delivered to viewers;
- **DRL-TS**[20]: This is a deep reinforcement learning-based transcoder selection scheme. After transcoding, the streaming is transmitted to BS for further delivery.

Fig.11 presents the comparative results of one simulation execution. When the Cloud-based method is used, the transcoding task is done at the cloud server. Although the cloud server has abundant computing resources, it has to transmit the transcoded stream from the remote server to BS. The transmitting process is time-consuming, which leads to a high system delay as shown in Fig.11. When employing DRL-TS, the transcoding is executed at the CNs and therefore the transcoding delay is low. The delivery process in DRL-TS is the same as in the Cloud-based

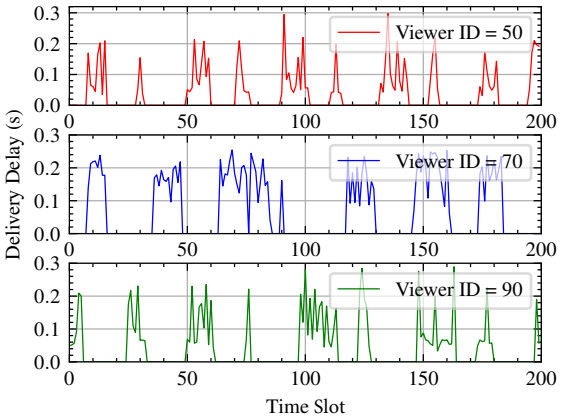

Figure 10: Delivery delay of different viewers.

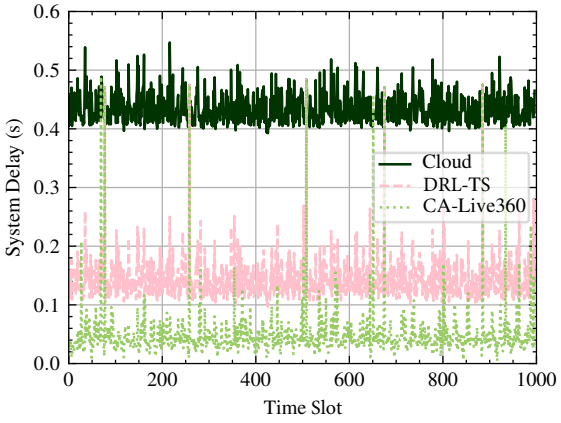

Figure 11: System delay of different schemes.

method: viewers get streaming from BS. Therefore, in terms of system delay, DRL-TS is better than the Cloud-based method. The difference between DRL-TS and CA-Live360 is the delivery stage. As in CA-Live360 there is a transcoding-aware delivery, it takes advantage of the proximity of the transcoder and requesters in crowd-assisted transcoding, and thus reduces the delivery delay. The results also show how our solution outperforms the other solutions and these results are consistent with our previous analysis.

7. Conclusion and Future Work

This paper proposes CA-Live360, a novel crowd-assisted live streaming solution to provide high-quality live 360-degree video streaming services. To achieve low-latency transcoding while maintaining fairness, we propose an innovative fairness-guaranteed transcoding task assignment scheme, using a novel Fair Bandit algorithm based on the combinatorial multi-armed bandit approach. Additionally, to minimize delivery latency, a transcoding-aware delivery scheme is proposed, which includes a provider-requester matching algorithm for efficient scheduling. The key advantage of CA-Live360 over similar schemes resides in its ability to simultaneously enables fair transcoding task assignment and low-latency streaming services. Our experiments demonstrate CA-Live360's superiority to state-of-the-art schemes in terms of latency and fairness, validating its effectiveness. Despite its strengths, CA-Live360 has some limitations: primarily focusing on the downlink from BS to the user, and lacking deployment in real-world scenarios. These limitations suggest the need for further research and practical testing. Future work will delve into optimizing the transmission from a broadcaster's perspective, focusing on upload strategies to BS. Additionally, implementing CA-Live360 in a real-world system for practical testing will help identify and address challenges that might arise in deployments.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (NSFC) under grant No. 62225105, 62394323, 62301070, and by the Postdoctoral Science Foundation of China under grant No. 2022M720518.

References

[1] TwitchTracker, https://twitchtracker.com/statistics.
[2] C. Tian et al.,"VSOIQE: A Novel Viewport-Based Stitched 360° Omnidirectional Image Quality Evaluator," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 32, no. 10, pp. 6557-6572, Oct. 2022.
[3] VR - statistics and facts. https://www.statista.com/topics/2532/virtual-reality-vr/.
[4] J. Yi et al., "An Analysis of Delay in Live 360° Video Streaming Systems," ACM MM 2020 - Proceedings of the 28th ACM International Conference on Multimedia. 2020, pp. 1-6.
[5] Low Latency Streaming: What is it and How can it be solved? https://bitmovin.com/cmaf-low-latency-streaming/
[6] C. Dong, Y. Jia, H. Peng, X. Yang and W. Wen, "A Novel Distribution Service Policy for Crowdsourced Live Streaming in Cloud Platform," in IEEE Transactions on Network and Service Management, vol. 15, no. 2, pp. 679-692, June 2018.
[7] X. Li, M. A. Salehi, Y. Joshi, M. K. Darwich, B. Landreneau and M. Bayoumi, "Performance Analysis and Modeling of Video Transcoding Using Heterogeneous Cloud Services," in IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 4, pp. 910-922, 1 April 2019.
[8] Y. Zheng, D. Wu, Y. Ke, C. Yang, M. Chen and G. Zhang, "Online Cloud Transcoding and Distribution for Crowdsourced Live Game Video Streaming," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 27, no. 8, pp. 1777-1789, Aug. 2017.
[9] Z. Zhang, R. Wang, F. R. Yu, F. Fu and Q. Yan, "QoS Aware Transcoding for Live Streaming in Edge-Clouds Aided HetNets: An Enhanced Actor-Critic Approach," in IEEE Transactions on Vehicular Technology, vol. 68, no. 11, pp. 11295-11308, Nov. 2019.
[10] S. Wang, S. Bi and Y. -J. A. Zhang, "Adaptive Wireless Video Streaming: Joint Transcoding and Transmission Resource Allocation," in IEEE Transactions on Wireless Communications, vol. 21, no. 5, pp. 3208-3221, May 2022.
[11] H. Yuan and M. Zhou, "Profit-Maximized Collaborative Computation Offloading and Resource Allocation in Distributed Cloud and Edge Computing Systems," in IEEE Transactions on Automation Science and Engineering, vol. 18, no. 3, pp. 1277-1287, July 2021.
[12] L. Ren, Y. Liu, X. Wang, J. Lü and M. J. Deen, "Cloud–Edge-Based Lightweight Temporal Convolutional Networks for Remaining Useful Life Prediction in IIoT," in IEEE Internet of Things Journal, vol. 8, no. 16, pp. 12578-12587, 15 Aug.15, 2021.
[13] X. Huang, L. He, L. Wang and F. Li, "Towards 5G: Joint Optimization of Video Segment Caching, Transcoding and Resource Allocation for Adaptive Video Streaming in a Multi-Access Edge Computing Network," in IEEE Transactions on Vehicular Technology, vol. 70, no. 10, pp. 10909-10924, Oct. 2021.

[14] H. Xiao et al., "Transcoding-Enabled Cloud–Edge–Terminal Collaborative Video Caching in Heterogeneous IoT Networks: An Online Learning Approach With Time-Varying Information," in IEEE Internet of Things Journal, vol. 11, no. 1, pp. 296-310, 1 Jan.1, 2024.
[15] Y. Zhu, Q. He, J. Liu, B. Li and Y. Hu, "When Crowd Meets Big Video Data: Cloud-Edge Collaborative Transcoding for Personal Livecast," in IEEE Transactions on Network Science and Engineering, vol. 7, no. 1, pp. 42-53, 1 Jan.-March 2020.
[16] Q. He, C. Zhang and J. Liu, "CrowdTranscoding: Online Video Transcoding With Massive Viewers," in IEEE Transactions on Multimedia, vol. 19, no. 6, pp. 1365-1375, June 2017.
[17] X. Liu, M. Derakhshani and S. Lambotharan, "Joint Transcoding Task Assignment and Association Control for Fog-Assisted Crowdsourced Live Streaming," in IEEE Communications Letters, vol. 23, no. 11, pp. 2036-2040, Nov. 2019.
[18] X. Chen, C. Xu, M. Wang, Z. Wu, L. Zhong and L. A. Grieco, "Augmented Queue-Based Transmission and Transcoding Optimization for Livecast Services Based on Cloud-Edge-Crowd Integration," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 11, pp. 4470-4484, Nov. 2021.
[19] Sanjay Segu Nagesh, Niroshinie Fernando, Seng W. Loke, AzadehGhari Neiat, and Pubudu N. Pathirana. 2022. Opportunistic mobile crowd computing: task-dependency based work-stealing. In Proceedings of the 28th Annual International Conference on Mobile Computing And Networking (MobiCom '22). Association for Computing Machinery, New York, NY, USA, 775–777. https://doi.org/10.1145/3495243.3558751
[20] M. Liu, Y. Teng, F. R. Yu, V. C. M. Leung and M. Song, "A Deep Reinforcement Learning-Based Transcoder Selection Framework for Blockchain-Enabled Wireless D2D Transcoding," in IEEE Transactions on Communications, vol. 68, no. 6, pp. 3426-3439, June 2020.
[21] Y. Ma, C. Xu, X. Chen, H. Xiao, L. Zhong and G. -M. Muntean, "Fairness-Guaranteed Transcoding Task Assignment for Viewer-Assisted Crowdsourced Livecast Services," ICC 2021 - IEEE International Conference on Communications, 2021, pp. 1-6.
[22] FFmpeg: https://ffmpeg.org
[23] X. Chen et al., "A Universal Transcoding and Transmission Method for Livecast with Networked Multi-Agent Reinforcement Learning," IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, 2021, pp. 1-10.
[24] L. Zhong et al., "A Multi-user Cost-efficient viewer-assisted VR Content Delivery Solution in 5G-and-beyond Heterogeneous Networks," in IEEE Transactions on Mobile Computing. Early Access, 2022. DOI: 10.1109/TMC.2022.3162147.
[25] Q. Zeng, Z. Yin, Q. Pan, et al., "TVSR-OR: Tile-based 360-degree video streaming over real time streaming protocol with optimized read," in Transactions on Emerging Telecommunications Technologies, vol. 34, no. 5, e4759, May. 2023.
[26] S. Yang, J. Hu, K. Jiang, et al., "Hybrid-360: An adaptive bitrate algorithm for tile-based 360 video streaming," in Transactions on Emerging Telecommunications Technologies, vol. 33, no. 4, e4430, Apr. 2023.
[27] L. Chen, Y. Tang, J. Xia, et al., "Multi-MEC collaboration for VR video transmission: Architecture and cache algorithm design," in Computer Networks, vol. 234. Oct. 2023.
[28] H. Xiao et al., "A Transcoding-Enabled 360° VR Video Caching and Delivery Framework for Edge-Enhanced Next-Generation Wireless Networks," in IEEE Journal on Selected Areas in Communications, vol. 40, no. 5, pp. 1615-1631, May 2022.
[29] K. Long, Y. Cui, C. Ye and Z. Liu, "Optimal Wireless Streaming of Multi-Quality 360 VR Video By Exploiting Natural, Relative Smoothness-Enabled, and Transcoding-Enabled Multicast Opportunities," in IEEE Transactions on Multimedia, vol. 23, pp. 3670-3683, 2021.
[30] Y. Liu, J. Liu, A. Argyriou and S. Ci, "MEC-Assisted Panoramic VR Video Streaming Over Millimeter Wave Mobile Networks," in IEEE Transactions on Multimedia, vol. 21, no. 5, pp. 1302-1316, May 2019.
[31] C. Guo, L. Zhao, Y. Cui, Z. Liu and D. W. K. Ng, "Power-Efficient Wireless Streaming of Multi-Quality Tiled 360 VR Video in MIMO-OFDMA Systems," in IEEE Transactions on Wireless Communications, vol. 20, no. 8, pp. 5408-5422, Aug. 2021.
[32] X. Tan, S. Wang, X. Xu, Q. Zheng, J. Yang and S. Chen, "DACOD360: Deadline-Aware Content Delivery for 360-Degree Video Streaming Over MEC Networks," in IEEE Transactions on Multimedia, vol. 26, pp. 4168-4182, 2024.
[33] F. Hu, Y. Deng and A. H. Aghvami, "Cooperative Multigroup Broadcast 360° Video Delivery Network: A Hierarchical Federated Deep Reinforcement Learning Approach," in IEEE Transactions on Wireless Communications, vol. 21, no. 6, pp. 4009-4024, June 2022.
[34] C. Perfecto, M. S. Elbamby, J. D. Ser and M. Bennis, "Taming the Latency in Multi-User VR 360°: A QoE-Aware Deep Learning-Aided Multicast Framework," in IEEE Transactions on Communications, vol. 68, no. 4, pp. 2491-2508, April 2020.
[35] R. Zhang et al., "Buffer-Aware Virtual Reality Video Streaming With Personalized and Private Viewport Prediction," in IEEE Journal on Selected Areas in Communications, vol. 40, no. 2, pp. 694-709, Feb. 2022.
[36] Y. Zhu, G. Zhai, Y. Yang, H. Duan, X. Min and X. Yang, "Viewing Behavior Supported Visual Saliency Predictor for 360 Degree Videos," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 32, no. 7, pp. 4188-4201, July 2022.
[37] C. Zheng, J. Yin, F. Wei, Y. Guan, Z. Guo and X. Zhang, "STC: FoV Tracking Enabled High-Quality 16K VR Video Streaming on Mobile Platforms," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 32, no. 4, pp. 2396-2410, April 2022.
[38] N. Kan, J. Zou, C. Li, W. Dai and H. Xiong, "RAPT360: Reinforcement Learning-Based Rate Adaptation for 360-Degree Video Streaming With Adaptive Prediction and Tiling," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 32, no. 3, pp. 1607-1623, March 2022.
[39] H. Xiao, C. Xu, C. Fang, et al., " VAAC-IM: Viewing Area Adaptive Control in Immersive Media Transmission," Proceedings of the 34th edition of the Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '24). Association for Computing Machinery, New York, NY, USA, 8–14.
[40] N. Dao, D. Vu, W. Na, et al., "Adaptive bitrate streaming in multi-user downlink NOMA edge caching systems with imperfect SIC, " in Computer Networks, vol. 212, July 2022.
[41] J. Shi, L. Pu and J. Xu, "Allies: Tile-Based Joint Transcoding, Delivery and Caching of 360° Videos in Edge Cloud Networks," 2020 IEEE 13th International Conference on Cloud Computing (CLOUD), Beijing, China, 2020, pp. 337-344.
[42] M. S. M. Gismalla et al., "Survey on Device to Device (D2D) Communication for 5GB/6G Networks: Concept, Applications, Challenges, and Future Directions," in IEEE Access, vol. 10, pp. 30792-30821, 2022.
[43] E. Price and D. P. Woodruff, "Applications of the Shannon-Hartley theorem to data streams and sparse recovery," 2012 IEEE International Symposium on Information Theory Proceedings, 2012, pp. 2446-2450.
[44] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," Machine Learning, vol. 47, no. 2, pp. 235-256, May 2002.
[45] F. Li, J. Liu and B. Ji, "Combinatorial Sleeping Bandits with Fairness Constraints," IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, Paris, France, 2019, pp. 1702-1710.
[46] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," Synthesis Lectures on Communication Networks, vol. 3. San Rafael, CA, USA: Morgan Claypool, 2010, pp. 1–211.
[47] Tor Lattimore and Csaba Szepesv ari. Bandit algorithms. Cambridge University Press, 2020.
[48] Roth, A.E. Deferred acceptance algorithms: history, theory, practice, and open questions. Int J Game Theory 36, 537–569, 2008.
[49] Z. Zhang, R. Wang, F. R. Yu, F. Fu and Q. Yan, "QoS Aware Transcoding for Live Streaming in Edge-Clouds Aided HetNets: An Enhanced Actor-Critic Approach," in IEEE Transactions on Vehicular Technology, vol. 68, no. 11, pp. 11295-11308, Nov. 2019.
[50] C. Wu, Z. Tan, Z. Wang, and S. Yang, "A Dataset for Exploring User Behaviors in VR Spherical Video Streaming." Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17). Association for Computing Machinery, New York, NY, USA, 193–198, 2017.

Appendix A. Proof of Theorem 1

We prove theorem 1 by the Lyapunov-drift analysis. $\mathbf{Q(t)} = \{Q_0(t), Q_1(t), ..., Q_K(t)\}$ is defined as a vector of virtual queue in

time slot t. Then the Lyapunov function is expressed as

$$L(\mathbf{Q(t)}) \triangleq \frac{1}{2}\sum_{i\in\mathcal{K}}(Q_i(t))^2\,, \tag{A.1}$$

Then the Lyapunov drift is given as

$$\begin{aligned}
&\triangle L(\mathbf{Q(t)}) \qquad\qquad (A.2)\\
&\triangleq L(\mathbf{Q(t+1)}) - L(\mathbf{Q(t)})\\
&= \frac{1}{2}\sum_{i\in\mathcal{K}} Q_i(t+1)^2 - \frac{1}{2}\sum_{i\in\mathcal{K}} Q_i(t)^2\\
&= \frac{1}{2}\sum_{i\in\mathcal{R}(t+1)} Q_i(t+1)^2 - \frac{1}{2}\sum_{i\in\mathcal{R}(t+1)} Q_i(t)^2\\
&\quad + \frac{1}{2}\sum_{i\in\mathcal{O}(t+1)} Q_i(t+1)^2 - \frac{1}{2}\sum_{i\in\mathcal{O}(t+1)} Q_i(t)^2\\
&\overset{(a)}{\leq} \frac{1}{2}\sum_{i\in\mathcal{R}(t+1)} [Q_i(t) + x_i - l_i(t)]^2 - \frac{1}{2}\sum_{i\in\mathcal{R}(t+1)} Q_i(t)^2\\
&= \frac{1}{2}\sum_{i\in\mathcal{R}(t+1)} [x_i - l_i(t)]^2 + \sum_{i\in\mathcal{R}(t+1)} Q_i(t)\,[x_i - l_i(t)]\\
&\overset{(b)}{\leq} \frac{K}{2} + \sum_{i\in\mathcal{R}(t+1)} x_i Q_i(t) - \sum_{i\in\mathcal{R}(t+1)} l_i(t) Q_i(t). \qquad (A.3)
\end{aligned}$$

where (a) is based on the defined queue length (21). And (b) is because that $[x_i - l_i(t)]^2 \leq 1$, and $|\mathcal{R}(t+1)| \leq K$.

Then, take the conditional expectation of two sides in inequality (A.3):

$$\begin{aligned}
&\mathbb{E}\left[L(\mathbf{Q(t+1)}) - L(\mathbf{Q(t)})|\mathbf{Q(t)}\right]\\
&\leq \frac{K}{2} + \sum_{i\in\mathcal{R}(t+1)} x_i Q_i(t) - \mathbb{E}\left[\sum_{i\in\mathcal{R}(t+1)} l_i(t) Q_i(t)|\mathbf{Q(t)}\right]\\
&= \frac{K}{2} + \sum_{i\in\mathcal{R}(t+1)} x_i Q_i(t) + \mathbb{E}\left[\sum_{i\in\mathcal{R}(t+1):i\in\mathcal{T}(t)} \eta\mu_i^{UCB}|\mathbf{Q(t)}\right]\\
&\quad - \mathbb{E}\left[\sum_{i\in\mathcal{R}(t+1):i\in\mathcal{T}(t)} Q_i(t) + \eta\mu_i^{UCB}\mathbf{Q(t)}\right]\\
&\leq \frac{K}{2} + \sum_{i\in\mathcal{R}(t+1)} x_i Q_i(t) + \eta m\\
&\quad - \mathbb{E}\left[\sum_{i\in\mathcal{R}(t+1):i\in\mathcal{T}(t)} \left(Q_i(t) + \eta\mu_i^{UCB}\right)|\mathbf{Q(t)}\right] \qquad (A.4)
\end{aligned}$$

Based on Lemma 1 in [45] and Theorem 4.5 in [46], we can get the following inequalities for any $\epsilon > 0$

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i\in\mathcal{R}(t+1):i\in\mathcal{T}(t)} Q_i(t) + \eta\mu_i^{UCB}|\mathbf{Q(t)}\right]\\
&\geq \sum_{i\in\mathcal{R}(t+1):i\in\mathcal{T}(t)} Q_i(t)(x_i+\epsilon)\\
&\geq \sum_{i\in\mathcal{R}(t+1)} Q_i(t)(x_i+\epsilon) \qquad (A.5)
\end{aligned}$$

Then, we get

$$\begin{aligned}
&\mathbb{E}\left[L(\mathbf{Q(t+1)}) - L(\mathbf{Q(t)})|\mathbf{Q(t)}\right]\\
&\leq \frac{K}{2} + \sum_{i\in\mathcal{R}(t+1)} x_i Q_i(t) + \eta m\\
&\quad - \sum_{i\in\mathcal{R}(t+1)} Q_i(t)(x_i+\epsilon)\\
&= D_1 - \epsilon\sum_{i\in\mathcal{R}(t+1)} Q_i(t) \qquad (A.6)
\end{aligned}$$

where $D_1 = \frac{K}{2} + \eta m$. Finally, invoking the Lyapunov Drift Theorem (Theorem 4.5) in [46], (A.6) suggests that virtual queues are not only mean rate stable, but also strongly stable, i.e.,

$$\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[Q_i(t)\right] \leq \frac{D_1}{\epsilon} \leq \infty \tag{A.7}$$

So, the proof is completed.

Appendix B. Proof of Theorem 2

We consider the optimal A-only policy π^*. Then $l_i^*(t)$ is the indicator and $\mathcal{T}^*(t)$ is the set of transcoder under policy π^*. The system reward with policy pi is defined as

$$R^* = \mathbb{E}\left[\sum_{i\in\mathcal{T}^*(t)}\mu_i\right]. \tag{B.1}$$

Then, the regret can be expressed as

$$\begin{aligned}
\Xi(T) &= R^* - R\\
&= \mathbb{E}\left[\sum_{i\in\mathcal{T}^*(t)}\mu_i\right] - \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T}\sum_{i\in\mathcal{T}^*(t)} r_i(t)\right]\\
&= \frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\left[\sum_{i\in\mathcal{T}^*(t)}\mu_i - \sum_{i\in\mathcal{T}(t)}\mu_i\right] \qquad (B.2)
\end{aligned}$$

We defined $\triangle\ R(t) = \sum_{i\in\mathcal{K}}\left[l_i^*(t) - l_i(t)\right]\mu_i$. Then, the upper bound of drift-plus-regret is

$$\begin{aligned}
&L(\mathbf{Q(t+1)}) - L(\mathbf{Q(t)}) + \eta\ \triangle\ R(t)\\
&\leq \frac{K}{2} + \sum_{i\in\mathcal{R}(t+1)} x_i Q_i(t) - \sum_{i\in\mathcal{R}(t+1)} l_i(t) Q_i(t)\\
&\quad + \eta\sum_{i\in\mathcal{K}}\left[l_i^*(t) - l_i(t)\right]\mu_i\\
&\quad + \eta\sum_{i\in\mathcal{R}(t+1)}\left[l_i^*(t) - l_i(t)\right]\mu_i + \eta\sum_{i\in\mathcal{O}(t+1)}\left[l_i^*(t) - l_i(t)\right]\mu_i\\
&= \frac{K}{2} + \sum_{i\in\mathcal{R}(t+1)}\left(Q_i(t) + \eta\mu_i\right)\left(l_i^*(t) - l_i(t)\right)\\
&\quad + \sum_{i\in\mathcal{R}(t+1)} Q_i(t)\left(x_i - l_i^*(t)\right) + \sum_{i\in\mathcal{O}(t+1)}\eta\left[l_i^*(t)\mu_i - l_i(t)\mu_i\right]\\
&\overset{(a)}{\leq} \frac{K}{2} + \sum_{i\in\mathcal{R}(t+1)}\eta\mu_i + \sum_{i\in\mathcal{R}(t+1)}\left(Q_i(t) + \eta\mu_i\right)\left(l_i^*(t) - l_i(t)\right)\\
&\quad + \sum_{i\in\mathcal{R}(t+1)} Q_i(t)\left(x_i - l_i^*(t)\right) \qquad (B.3)
\end{aligned}$$

where (a) is because that $l_i^*(t) \in \{0,1\}$ and $l_i(t) \in \{0,1\}$. Then, the expected drift-plus-regret is bounded by

$$\begin{aligned}
&\mathbb{E}\left[L(\mathbf{Q}(\mathbf{t}+\mathbf{1})) - L(\mathbf{Q}(\mathbf{t})) + \eta \triangle R(t)\right] \\
&\leq \frac{K}{2} + \sum_{i\in\mathcal{R}(t+1)} \eta\mu_i \\
&\quad + \sum_{i\in\mathcal{R}(t+1)} \mathbb{E}\left[(Q_i(t) + \eta\mu_i)\left(l_i^*(t) - l_i(t)\right)\right] \\
&\quad + \sum_{i\in\mathcal{R}(t+1)} \mathbb{E}\left[Q_i(t)\left(x_i - l_i^*(t)\right)\right] \\
&\overset{(a)}{\leq} \frac{K}{2} + D_2 \\
&\quad + \mathbb{E}\left[\sum_{i\in\mathcal{R}(t+1)} (Q_i(t) + \eta\mu_i)(l_i^*(t) - l_i(t))\right] \qquad \text{(B.4)}
\end{aligned}$$

where (a) follows from $\mathbb{E}\left[l_i^*(t)\right] \geq x_i$, because that policy π is stationary and feasible. $D_2 = \sum_{i\in K} \eta p_i \mu_i$ and p_i is the offline probability of viewer i. We defined $J(t) = \sum_{i\in\mathcal{R}(t+1)}(Q_i(t) + \eta\mu_i)(l_i^*(t) - l_i(t))$. By summing (B.4), for all $t \in \{1,2,...,T\}$, we obtain

$$\begin{aligned}
&\mathbb{E}\left[L(\mathbf{Q}(\mathbf{T})) - L(\mathbf{Q}(\mathbf{0}))\right] + \eta \sum_{t=0}^{T-1} \left[\triangle R(t)\right] \\
&\leq \frac{KT}{2} + TD_2 + \sum_{t=0}^{T-1} \mathbb{E}\left[J(t)\right] \qquad \text{(B.5)}
\end{aligned}$$

Because $L(\mathbf{Q}(\mathbf{T})) \geq= $ and $L(\mathbf{Q}(\mathbf{0})) = 0$, we have $\eta \sum_{t=0}^{T-1} \left[\triangle R(t)\right] \leq \frac{KT}{2} + TD_2 + \sum_{t=0}^{T-1} \mathbb{E}\left[J(t)\right]$. By dividing both sides by $T\eta$, the following inequality holds

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\triangle R(t)\right] \leq \frac{K + 2D_2}{2\eta} + \frac{1}{T\eta}\sum_{t=0}^{T-1} \mathbb{E}\left[J(t)\right] \qquad \text{(B.6)}$$

Here we give the following bound directly

$$\frac{1}{T\eta}\sum_{t=0}^{T-1} \mathbb{E}\left[J(t)\right] \leq \frac{1}{T}\left[2\sqrt{6mK\log T} + (1 + \frac{5\pi^2}{12})K\right] \qquad \text{(B.7)}$$

The detailed analysis can be referred to [45]. Finally, by plugging (B.7) into (B.6), the proof of Theorem 2 is completed.