# Augmented Queue-Based Transmission and Transcoding Optimization for Livecast Services Based on Cloud-Edge-Crowd Integration

Xingyan Chen, Changqiao Xu, *Senior Member, IEEE*, Mu Wang, Zhonghui Wu, Lujie Zhong, and Luigi Alfredo Grieco, *Senior Member, IEEE*

***Abstract*— Nowadays, amateur broadcasters can massively generate video contents and stream them across the Internet. For this reason, crowdsourced livecast services (CLS) are attracting millions of users around the world. To provide a smooth and high-quality playback experience to viewers with diversified device configurations in dynamic network conditions, CLS providers have to find a way to deploy cost-effective transcoding operations by distributing the computation-intensive workload among Cloud, Edge, and Crowd. In addition, it is necessary to control transcoded streams from million broadcasters to worldwide viewers. To address these challenges, we propose a novel stochastic approach that jointly optimizes the usage of transmission resources (*e.g.*, bandwidth), and transcoding resources (*e.g.*, CPU) in CLS systems that leverage the cooperation of Cloud, Edge, and Crowd technologies. In particular, we first design an augmented queue structure that can jointly capture the dynamic features of data transmission and online transcoding, based on the virtual queue technology. Then, we formulate a joint resource allocation problem, using stochastic optimization arguments, and devise an Accelerated Gradient Optimization (AGO) algorithm to solve the optimization problem in a scalable way. Moreover, we provide four main theoretical results that characterize the algorithm's steady-state queue-length, optimality, and fast-convergence. By conducting both numerical simulations and system-level evaluations based on our prototype, we demonstrate that our solution provides lower system costs and higher QoE performance against state-of-the-art solutions.**

***Index Terms*— Video transcoding, transmission, crowdsourced livecast services, cloud computing, edge computing, crowd.**

Manuscript received August 4, 2020; revised October 25, 2020 and December 15, 2020; accepted December 21, 2020. Date of publication December 28, 2020; date of current version October 28, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61871048, Grant 61872253, and Grant 62001057; in part by the BUPT Excellent Ph.D. Students Foundation CX2019205; and in part by the National Key Research and Development Program of China under Grant 2018YFE0205502. This article was recommended by Associate Editor C. Wu. *(Corresponding author: Changqiao Xu.)*

Xingyan Chen, Changqiao Xu, Mu Wang, and Zhonghui Wu are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: chenxingyan@bupt.edu.cn; cqxu@bupt.edu.cn; wangmu@bupt.edu.cn; zhdacry@bupt.edu.cn).

Lujie Zhong is with the Information Engineering College, Capital Normal University, Beijing 100048, China (e-mail: zhonglj@cnu.edu.cn).

Luigi Alfredo Grieco is with the Department of Electrical and Information Engineering, Politecnico di Bari, 70126 Bari, Italy, and also with the Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), 1-00133 Rome, Italy (e-mail: a.grieco@poliba.it).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TCSVT.2020.3047859.

Digital Object Identifier 10.1109/TCSVT.2020.3047859

## I. Introduction

THANKS to the impressive capacity magnification that wireless communication technologies are experiencing nowadays, we are currently assisting to the explosion of **C**rowdsourced **L**ivecast **S**ervices (CLS) such as Twitch [1] and Youtube Live [2]. According to the latest TwitchTracker [3] stats, Twitch has almost 5.8 million average monthly broadcasters, over 511 billion minutes watched, and near 1.94 million average concurrent viewers in the first half of 2020. Meanwhile, the latest YouTube's Statistics revealed that Coachella's first weekend earned over 82 million live views [4] on Youtube Live. The increasing popularity of CLS also brings new challenges to current CLS systems. CLS platforms need to continuously deliver a massive amount of contents from broadcasters to viewers. In the process of delivery, the stream also needs to be transcoded into multi-quality representations to match different configurations of networks and end devices. As a consequence, the deployment of CLS requires very demanding computing capabilities and bandwidth needs.

Many computing paradigms are considered to address those challenges [5]–[22]. For example, Dong *et al.* in [11] propose a cloud-based architecture that utilizes multiple **C**ontent **D**elivery **N**etworks (CDN) to distribute and transcode video content for CLS cooperatively. However, the multiple versions of transcoded streams overwhelm the backbone network. Furthermore, since the Cloud is geographically remote from the viewers, CLS vulnerable to high delivering latency and congestion fluctuations in wide area networks which affects the smoothness video playback and the interactive experience [15]. To overcome these drawbacks, edge servers are considered as an alternative because they are able to move the workload closer to viewers [15]–[17]. Ma *et al.* in [15] present an edge-assisted mobile personal livecast system which can reduce the broadcast latency and alleviate the backbone network traffic by collaboratively leveraging Cloud and Edge servers. However, renting dedicated Cloud or Edge servers to transcode CLS video online requires huge volumes of concurrent computational resources, which is extremely expensive [19].

The viewers' devices with increased capabilities and intelligence are becoming an alternative resource to facilitate the development of CLS [23], [24]. Many researchers are proposing to leverage the abundant and cheap computing resources available at viewers' devices [19]–[22]. Authors in [19] propose a fog-assisted computing framework in which the global system is divided into multiple regions. In each region, a regional data center is responsible for offloading the transcoding workload to viewers and for recollecting the processed contents. Despite the economic advantages of Crowd compared to Cloud and Edge-based schemes, each regional data center needs to assign and recollect live video content to viewers, which exacerbate the bandwidth consumption and increase the time-shift between different viewers. Furthermore, most of the current solutions are based on deterministic optimization, which may be infeasible in scenarios characterized by time-varying user demand and unpredictable network conditions.

In addition, once contents have been transcoded, they need to be delivered to viewers. This is another challenging task that is being approached in literatures [25]–[29]. For instance, Wang *et al.* in [25] propose a distributed multipath transmission framework to optimally control the data rate of live streaming with stochastic optimization technology. However, pure transmission solutions cannot be directly applied to CLS. As we know, CLS contents may need to be transcoded by one or more nodes during the delivery from broadcasters to viewers. Merely considering the path quality may flood a large amount of data into only a few transcoding nodes experiencing good network conditions and lead to the overload of those nodes or, even worse, to the collapse of the CLS system. To fill the gap, authors in [26] propose an online joint transcoding and delivery approach to achieve high service quality and low resource overhead for adaptive video streaming. However, the authors formulate the problem as a conventional 0/1 knapsack problem, which is NP-hard, and design a greedy-based heuristic algorithm. Consequently, the solution is suboptimal. Besides, this work also lacks the analysis of performance guarantee for the heuristic algorithm. Therefore, an optimal and theory-guaranteed joint resource optimization approach for CLS systems is still needed. For clarity, we summarize two main challenges in the design of CLS systems:

- *Cooperative Transcoding:* The Cloud provides stable and efficient computing resources, but inflates latencies. On the other hand, the edge offers low latency but expensive transcoding service, whereas the Crowd supplies cheap but unstable computing. These three computing paradigms have unique advantages and drawbacks. How to integrate their strengths while remedying their shortcomings is crucial to achieve cost-effective services.
- *Data Transmission*: In a CLS, video contents may be transcoded by one or more nodes during the delivery from broadcasters to viewers. Overlooking the workload of the transcoding nodes may result in the overload of few nodes, while most other nodes are idle in CLS systems. With few nodes operating, the systems will produce transcoding congestion and high latency while further impairing CLS performance seriously. Moreover, since resource provision and network conditions are highly dynamic and unpredictable, designing a rate control algorithm which can timely adapt the variation of network conditions to provide high-quality CLS is non-trivial.

To address these challenges, we propose a novel stochastic mechanism for data transmission and online transcoding that jointly optimizes computing and bandwidth resource utilization with quantifiable delay performance. An augmented queue structure is designed for **C**loud-**E**dge-**C**rowd (CEC) cooperation-based CLS systems to capture the stochastic feature of transmission and transcoding by constructing a model grounded on dual-queues model. The first virtual queue represents the workload on transcoding nodes, and the second one represents the number of packets waiting to be sent. Then, we formulate the resource allocation problem of transmission and transcoding as a stochastic optimization problem. We devise a distributed **A**ccelerated **G**radient **O**ptimization (AGO) algorithm to minimize the long-term cost of both transcoding and transmission with $\mathcal{O}\left(\sqrt{V}\right)$ steady-state queue-length performance. Since the queuing delay for transcoding and delivery is the main latency of the CLS content access, reducing the queue-length can improve the service performance effectively. To the best of our knowledge, we are the first to provide a theoretical analysis for understanding the cost-delay trade-off of dynamic CLS systems. In doing so, we provide analytical answers to assess the workload at each node, the data transmission rate for viewers with different device configurations, and the delay that can be incurred at different nodes and links. The main contributions of this paper are summarized as follows:

- We propose a novel augmented queue-based structure to consider data transmission and online transcoding problems jointly. The augmented queue model mainly consists of two virtual queues. We show how the two virtual queues catch the stochastic features of transcoding and flow control problems. Using this model, we formulate the joint resource allocation problem as a stochastic optimization problem.
- To solve the stochastic optimization problem, we propose Accelerated Gradient Optimization algorithm and investigate the feasibility and the algorithm complexity of its practical implementation. We further provide the theoretical analysis of queue-length bounds, optimality, and convergence performance for our algorithm. To best of our knowledge, our work is the first to provide a theoretical basis for CEC cooperation-based CLS systems in stochastic conditions.
- We evaluate our algorithm through numerical simulations and experiments. Based on real datasets, we carry out a series of track-driven simulations with different parameters. The numerical results are in close agreement with the theoretical predictions. Further, we build a prototype to compare the performance of our approach with state-of-art solutions [15], [22]. The results show that our algorithm outperforms the current solutions in terms of resource cost and latency.

The related works are reviewed in Section II. Section III presents the framework and system model. Section IV formulates the joint resource allocation problem for transcoding and transmission. Section V introduces the design and implementation of AGO. Section VI gives four main theoretical results of our algorithm. We evaluate our design through trace-driven simulations and prototype test in section VII and present the conclusion in Section VIII.

## II. Related Works

So far, many researches consider transcoding or transmission for livecast, but few studies afford joint transcoding and transmission optimization solution that makes the CLS always face the challenge of the balance between improving CLS viewing experience and saving system overhead. Some researchers intend to improve the user viewing experience of CLS by using cloud computing to online transcode live video for bitrate adaptive [6], [8], [11]. Such as, Jiang *et al.* [6] proposed a scalable and accurate prediction system by leveraging CDN based Critical Feature Analytics to provide near real-time QoE predictions for live streaming services, thereby reducing 32% buffering time and improving 12% video bitrate. Aparicio-Pardo *et al.* [8] provide an online video transcoding solution to maximize the QoE with Cloud computing infrastructure. The authors formulate the problem as an integer linear program problem and propose a heuristic algorithm with sub-optimal performance. However, considering only the QoE improving is not enough, without effective allocation of massive online transcoding tasks, the CLS system may suffer a large transcoding load and crash [9]. Therefore, literatures [5], [7], [9], [10], [12]–[14], [17] begin to focus on the transcoding allocation problem in cloud computing. Zhang *et al.* [18] provide a **D**eep **R**einforcement **L**earning (DRL) based scheduling algorithm that by assigning viewers among multiple CDN to optimize resource allocation and improve the QoE. Zheng *et al.* [9] consider minimizing the operational cost of crowdsourced live game video streaming and propose a Lyapunov-based solution to online optimize video transcoding with Cloud computing. Moreover, the authors design an algorithm that can provide good-enough QoE performance for viewers. Besides, Dong *et al.* [12] put forward an online algorithm that can minimize system overhead while ensuring QoE for large CLS providers. A robust dynamic priority-based resource provisioning scheme is proposed in [14] to lower transcoding resource consumption while meeting the delay requirements of CLS with Cloud. Chen *et al.* [5] propose a cost-effective cloud solution that provides a fine-grained resource allocation while optimizing the delivery path from broadcasters, through streaming servers, to viewers to facilitate the geo-distributed crowdsourced live streaming. Besides, Wang *et al.* [7] present a generic framework by leveraging elastic Cloud to accommodate the time-varying resources demand of global viewers. Although these methods can effectively reduce transcoding overhead, passively ensuring service performance can not meet high-quality CLS requirements of viewers.

To further improve services performance, literatures [15], [16] propose to employ edge computing for assistance. H. Pang *et al.* in [16] conduct a particular analysis of viewer interaction patterns and propose a novel framework called PIECE to optimizes personalized QoE. Moreover, it provides, together with a novel **D**eep **N**eural **N**etwork (DNN) based algorithm, high-efficiency transcoding, and good video delivery over Cloud-Edge infrastructure. But computing resources of edge servers are expensive and limited, this method is powerless in the face of massive live video data [19]. Many researches [19]–[22] suggest leveraging crowd resources to further save the transcoding cost for the CLS system. Y. Zhu *et al.* in [20] propose a Cloud-Crowd collaborative solution by leveraging end viewers' devices to assist CLS transcoding. Compared with the cloud-based approach, this solution can save 93% of the transcoding overhead. Furthermore, Y. Zhu *et al.* extend the Cloud-Crowd solution to the large-scale CLS system [22]. The authors propose to use ever-grow personal computing devices to cope with the large-scale live video transcoding. Although this Cloud-Crowd solution can save the computation cost of Cloud, the introduction of the Crowd will increase the scale of the CLS system, making live video delivery more complex and challenging.

To provide efficient delivery, live data transmission has been studied extensively in recent years [25]–[29]. For instance, F. Chiariotti *et al.* in [27] indicate that consistently reliable data delivery is essential to provide high-quality live video services. The authors propose a novel latency control protocol for multi-path data delivery, which can provide stable transmission with high throughput and bounded delay. In addition, a novel DRL based approach is proposed in [28] to enhance the live video quality and reduce the end-to-end delay by jointly adjusting the video bitrate and target buffer size. Although these approaches can enhance the quality of the services by improving the transmission performance, they cannot be directly employed in CLS since they ignore the online transcoding. To provide high-quality cost-effective CLS, it is necessary to provide a solution to jointly consider transcoding and transmission. Literature [16] attempts to solve the joint optimization problem by finding the transcoding delivery path for live video flow. Authors in [26] formulate joint online transcoding and delivery problem as an NP-hard knapsack problem and design a heuristic algorithm to solve it. The above approaches have partly improved system overhead reduction and service performance, but formulating the joint problem as non-convex optimization can not guarantee the optimality of their solutions. Compared with [16], [26], our work has several significant differences and advantages. First, we build a novel augmented queue model by converting transcoding task allocation to the virtual queue control problem (The longer the task queue, the more transcoding workloads). Secondly, we formulate the joint problem as a stochastic convex optimization based on the model. Finally, we design an accelerated gradient descent based optimization algorithm which can achieve lower queue-length bound $\mathcal{O}\left(\sqrt{V}\right)$, optimality $\mathcal{O}(1/V)$, and good convergence.

## III. Framework and System Model

In this section, we illustrate the framework of CEC Cooperation-based CLS system. We give the network model present the augmented queue structure with dual virtual queues to model. We use lowercase *italic* symbols to indicate scalars. Vectors and Sets are written in lowercase *italics* **bold** type and uppercase *italics* typo, respectively. Besides, matrices are denoted using uppercase, *italics* and **bold** front. We let $\Delta$

TABLE I
MATHEMATICAL NOTATIONS

| Symbol | Description |
|---|---|
| $N, m$ | number of live channels and video resolution |
| $\mathcal{F}, f$ | Set of the live channel and a specific one |
| $\mathcal{B}, b$ | Set of video resolution and a specific one |
| $\mathcal{G}(\mathcal{V}, \mathcal{L})$ | Undirected graph for the topology for CLS system |
| $\mathcal{V}, \mathcal{L}$ | Set of nodes and links for the CLS system |
| $\mathcal{P}, \mathcal{S}, \mathcal{C}$ | Broadcasters, Cloud and Edge, the viewers |
| $c_u[t]$ | Available transcoding capability of node $u$ at time $t$ |
| $c_l[t]$ | Available bandwidth of link $l$ at time $t$ |
| $\mathcal{O}_n$ | Order set of video resolution for $n$-th channel |
| $\mathcal{G}'(\mathcal{V}', \mathcal{L}')$ | Directed graph of transcoding process |
| $\mathcal{V}', \mathcal{L}'$ | Set of virtual nodes and links for the CLS system |
| $w_l$ | Weight of bandwidth cost per bit at link $l$ |
| $w_h$ | Weight of transcoding cost per bit from $b_0$ to $b_h$ |
| $\mathcal{T}$ | Set of the time-slotted system |
| $\boldsymbol{p}^+[t]$ | Vector of input rate for transcoding flow at $t$ |
| $\boldsymbol{p}^-[t]$ | Vector of output rate for transcoding flow at $t$ |
| $\boldsymbol{i}^+[t]$ | Vector of input rate for transmission flow at $t$ |
| $\boldsymbol{i}^-[t]$ | Vector of output rate for transmission flow at $t$ |
| $\boldsymbol{m}^+[t]$ | Vector of generating rate for transcoding task at $t$ |
| $\boldsymbol{m}^-[t]$ | Vector of processing rate for transcoding task at $t$ |
| $\boldsymbol{s}^+[t]$ | Vector of enqueue rate for transmission flow at $t$ |
| $\boldsymbol{s}^-[t]$ | Vector of send out rate for transmission flow at $t$ |
| $\boldsymbol{q}[t]$ | Vector of transcoding task queue at $t$ |
| $\boldsymbol{q}'[t]$ | Vector of transmission data queue at $t$ |
| $\mathcal{F}_l(u)$ | Set of flows sending by node $u$ through link $l$ |
| $\mathcal{F}_u(u)$ | Set of flows transcoding by node $u$ |
| $w_i$ | Weight of transcoding cost per bit for flow $i$ |
| $w_{i,u}$ | Weight of resource cost per bit for flow $i$ on node $u$ |
| $\beta$ | Weight factor between resource cost and the utility |
| $U(\bar{x}_i), \mathcal{U}(\bar{\boldsymbol{x}})$ | Utility function of the flow $i$ and Objective function |
| $\psi_u[t], \psi_l[t]$ | Nesterov's weight for node $u$ and link $l$ at time $t$ |
| $\boldsymbol{P}, P_v$ | Vector of alternative paths, selected path by node $v$ |
| $\epsilon$ | Distance between the rate $\boldsymbol{x}$ and the system capacity |

denotes the differential operator while $|\cdot|$ and $\|\cdot\|$ denote the 1 and 2-norm, respectively. Table I lists the mathematical notations used in this paper.

## A. CEC Cooperation-Based CLS System

We define that the live streaming library consists of $N$ different channels which are denoted by $\mathcal{F} = \{f_1, f_2, \ldots, f_N\}$ where each channel $f$ contains $m$ versions of different resolutions denoted by $\mathcal{B} = \{b_1, b_2, \ldots, b_m\}$. We define the bitrate of highest resolution video as $b_1$ and the lowest as $b_m$, so we have $b_1 > b_2 > \ldots > b_m$. Besides, we assume that the viewers can leverage their additional or idle resources to assist to the video transcoding and transmission for CLS.

As shown in the Fig. 1, we consider a CEC Cooperation-based CLS system, in which multiple types of nodes are involved, including broadcasters, **C**loud **S**ervers (CSs), **E**dge **S**ervers (ESs) and viewers with diversified terminal devices (TDs *i.e.*, personal computer, laptop). We define three roles for the nodes in this system: *Requester*, the viewer who issues a request for a specific channel $f$; *Transcoders*, an intermediate node (CSs, ESs or viewers) which receives the video from broadcaster or other *Transcoders* and converts them into the resolution asked by the *Requester*. *Provider*, the broadcaster who continuously generates and uploads the original data to the CLS and the intermediate nodes which contain the demand video contents or higher resolution version. As Fig. 1 shows, in CLS, a flow $i$ starting from the

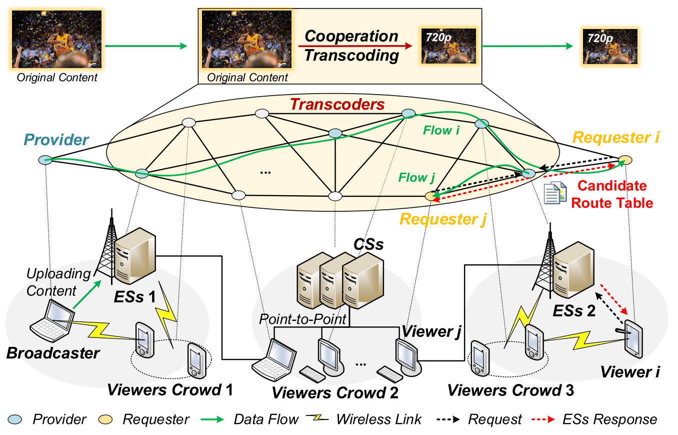


Fig. 1. Diagram of CEC cooperation-based CLS system.

*Provider* may go through multiple *Transcoders* and then reach the *Requester*. How to select the *Providers* and the *Transcoders* for different *Requesters* determines the performance of content delivery and transcoding. In order to achieve a highly efficient selection, the ESs maintain a candidate routing table which contains some alternative route paths to all *Providers* and the workload information of intermediate *Transcoders* on each path. These routes can be obtained by routing algorithms such as [30] or [31]. Since valid routing strategies already exist, this manuscript omits a discussion on routing issues and assumes that Requesters can directly obtain the candidate routing table from nearby ESs. The alternative *Transcoders* can be CSs, its ESs, and viewers which are in possess of the asked video or higher bitrate version. Due to topological proximity, once a *Requester* generates a request for channel $f$, the request will first be captured by the nearby ESs. Then, the *Requester* will select a transmission path based on the candidate routing table provided by the ESs and establish a delivery path with the *Provider*. If the *Provider* only have the higher resolution video, the *Transcoders* on the selected path will collaboratively transcode the video content provided by the *Provider* to deliver the resolution asked by the *Requester*. Otherwise, the *Transcoders* only need to deliver the content to the *Requester*. As Fig. 1 shows, if *Requester* $j$ asks the same channel as *Requester* $i$, then *Requester* $j$ might access the demand content from an intermediate node ***ESs* 2**, which is one of the *Transcoders* of flow $i$. Besides, to face mobile scenarios, the ESs need to update the candidate routing table periodically for the *Requesters*. The unpredictable computing provision and network condition of the intermediate *Transcoders* may further affect the content access of the *Requester*. Thus, the control of the data transmission rate from *Provider* to *Requesters* strongly impacts the quality of service provided by the CLS system. For this purpose, the *Provider* will determine the optimal transmission rate based on the path condition and the workload of all the *Transcoders* in the delivery path.

## B. Network Model

We consider the CLS system as an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{L})$ where $\mathcal{V}$ and $\mathcal{L}$ are the sets of nodes and links. Let $\mathcal{P}, \mathcal{S}, \mathcal{C} \subset \mathcal{V}$ denote the broadcasters, the Cloud or Edge servers, and

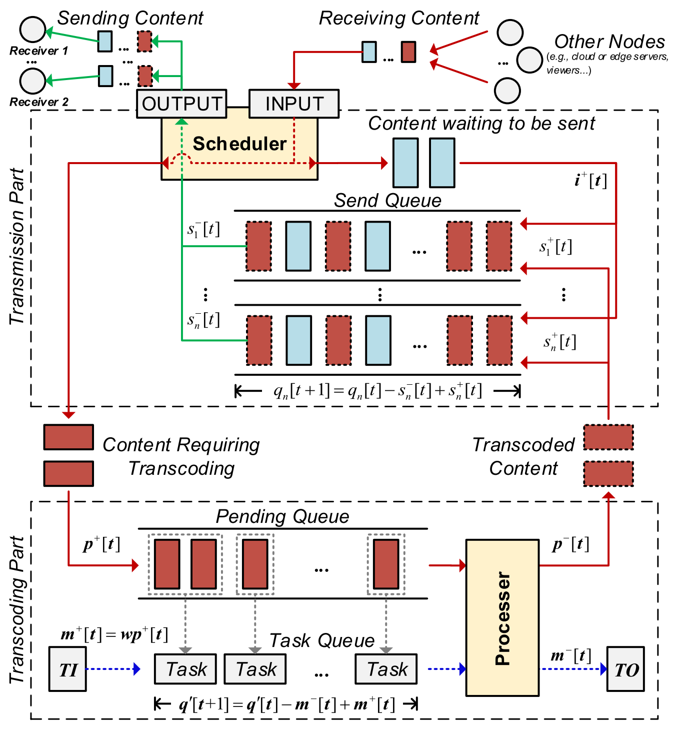


Fig. 2. Diagram of augmented queue structure for node $u$.

the viewers, respectively. We assume that node $u \in \mathcal{V}$ has transcoding capability, and the available computing resource for CLS at time $t$ is denoted by a time-varying variable $c_u[t]$. That is, the actual transcoding times for real-time video like CLS are often stochastically distributed due to the available resource of node $u$ are uncontrolled [32]. Thus, we have $c_u[t] \in \left[0, c_u^{max}\right]$ where $c_u^{max}$ is the maximum available computing resources. We consider $l = (u, v) \in \mathcal{L}$, $u, v \in \mathcal{V}$ as a network link from node $u$ to node $v$. When live video streaming goes through link $l$, it consumes $w_l$ transmission resources (*e.g.*, bandwidth) per bit. Since the live video streaming are subject to various cross-traffic from other applications, we denote the available bandwidth capacity of link $l$ for CLS as random variable $c_l[t] \in [0, c_l^{max}]$ where $c_l^{max}$ is the maximum available bandwidth. According to [33], the higher quality video streaming can be transcoded to lower quality versions, but not vice versa. Therefore, the video transcoding model can be described by a state transition of different versions. As mentioned above, the order set of version for the $n$-th channel is denoted by $\mathcal{O}_n = \{b_0, b_1, \ldots, b_m\}$ which can be represented as a directed graph $\mathcal{G}'(\mathcal{V}', \mathcal{L}')$. Each vertex $v_h' \in \mathcal{V}'$ corresponds to a version of video bitrate $b_h$ and each edge, such as $l' = (0, h) \in \mathcal{L}'$ represents the transcoding process from version $b_0$ to $b_h$. The required computation resources (*e.g.*, CPU usage time) per bit for transcoding from $b_0$ to $b_h$ is denoted by $w_h$ unit where $h \in \{1, 2, \ldots, m\}$.

## C. Augmented Queue Model

We consider a time-slotted system where time is denoted by $\mathcal{T} = \{0, 1, 2, \ldots\}$. In order to comprehensively consider transcoding and transmission effects, we augment each node $u \in \mathcal{V}$ with the dual-queues structure shown as Fig. 2. The augmented queue model mainly includes two parts: *Transmission Part* and *Transcoding Part*. Once node $u$ receives the content from other nodes, the received contents will first enter the *Transmission Part* from the **scheduler**'s INPUT and then be split into two portions. One will enter the *Transcoding Part* for transcoding, and the other will wait to be sent in the *Send Queue*. For ease of understanding, we consider the flow rate vector of contents requiring transcoding at time $t$ as $\boldsymbol{p}^+[t] = \left[p_1^+[t], p_2^+[t], \ldots, p_i^+[t], \ldots\right]$ and the vector of contents waiting to be sent at time $t$ as $\boldsymbol{i}^+[t] = \left[i_1^+[t], i_2^+[t], \ldots, i_i^+[t], \ldots\right]$ where $\{1, 2, \ldots, i, \ldots\}$ represents flow index for different viewers $i$. Once the content enters the *Transcoding Part*, it will wait to be processed in the *Pending Queue*. The TASK INPUT (***TI***) will estimate usage time of CPU processing for different flow $p_i^+[t]$ according to its transcoding requirements (*e.g.*, transcode to 720P). For convenience, we define the processing time unit of per bit for flow $i$ as $w_i$, so we have the task generating rate of flow $i$ is equal to $m_i^+[t] = w_i p_i^+[t]$. To analyze the transcoding process, we construct the *Task Queue* $\boldsymbol{q}'[t]$ to represent the process of content in the *Pending Queue*. We denote $\boldsymbol{m}^-[t]$ as processing rate and have the *Task Queue* evolution as follow:

$$\boldsymbol{q}'[t+1] = \left\{\boldsymbol{q}'[t] - \boldsymbol{m}^-[t] + \boldsymbol{m}^+[t]\right\}^+ \tag{1}$$

When the content is transcoded, the transcoded data leaves the *Transcoding Part* and returns to the *Transmission Part*. We define the leave rate as $\boldsymbol{p}^-[t]$, and the transcoded content will merge into the *Send Queue* $\boldsymbol{q}[t]$ with flows $\boldsymbol{i}^+[t]$. Thus, the enqueue rate of *Send Queue* can be represented as $\boldsymbol{s}^+[t] = \boldsymbol{i}^+[t] + \boldsymbol{p}^-[t]$. We define the send out rate as $\boldsymbol{s}^-[t] = \left[s_1^-[t], s_2^-[t], \ldots, s_j^-[t], \ldots\right]$ where $s_j^-[t]$ denotes the send out rate of one-hop downstream receiver $j$. Since $s_j^-[t]$ may contain multiple viewers' flows, it can be rewritten as $s_j^-[t] = \sum_{i \in \mathcal{V}(j)} s_i^-[t]$ where $\mathcal{V}(j)$ represents the set of flows send to receiver $j$ and $\{1, 2, .., i, \ldots\}$ is flow index. The *Send Queue* update can be expressed as:

$$\boldsymbol{q}[t+1] = \left\{\boldsymbol{q}[t] - \boldsymbol{s}^-[t] + \boldsymbol{s}^+[t]\right\}^+ \tag{2}$$

where $\{\cdot\}^+ \triangleq max\{0, \cdot\}$. In addition, according to [34], [35], we introduce the definitions of mean rate stability and strong stability as follows.

*Definition 1: (Mean Rate Stability) For any queue $q[t]$ over discrete time $t \in \{0, 1, 2, \ldots\}$ if:*

$$\limsup_{t \to \infty} \mathbb{E}\{\|q[t]\|\} < \infty \tag{3}$$

*The queue-length is finite in steady-state and $q[t]$ is mean rate stable.*

*Definition 2: (Strong Stability) For any discrete queue $q[t]$, that satisfies the following equation*

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\|q[\tau]\|\} < \infty \tag{4}$$

*We say that the $q[t]$ is strongly stable.*

## IV. Problem Formulation

Our goal is to find the lowest resource consumption solution of cooperation transcoding and flow control for the CLS system while satisfying the viewers' service demands. Firstly, we consider the transmission cost. As previously mentioned, $s_j^-[t]$ refers to the send rate of flow $j$ at time $t$. We have the average send rate of flow $j$ :

$$\bar{s}_j \triangleq \lim_{t\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in\mathcal{V}(j)} s_i^-[t] = \sum_{i\in\mathcal{V}(j)} \bar{s}_i[t] \tag{5}$$

where $\bar{s}_i[t]$ is the average send rate of flow $i$. We can write the transmission resource cost of link $l$ as $\sum_{j\in\mathcal{F}_l(u)} w_l\bar{s}_j$ where $\mathcal{F}_l(u)$ represents the set of flows sending by node $u$ through link $l$, and $w_l$ is transmission cost per bit of link $l$. Besides, the average transcoding rate for flow $i \in \{1, 2, \dots, n\}$ can be expressed as

$$\bar{m}_i \triangleq \lim_{t\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} m_i^-[t] \tag{6}$$

Similarly, the transcoding overhead for node $u \in \mathcal{V}$ can be expressed as $\sum_{i\in\mathcal{F}_u(u)} w_i\bar{m}_i$, where $\mathcal{F}_u(u)$ represents the set of flows transcoding by node $u$, and $w_i$ is the required transcoding resources per bit for flow $i$. Because $\bar{s}$ and $\bar{m}$ have similar formalization, we use $\bar{x}$ to represent $\{\bar{s}, \bar{m}\}$ for convenience. Therefore, the consumption of node $u \in \mathcal{V}$ can be expressed as $\sum_{i\in\mathcal{F}(u)} w_{i,u}\bar{x}_i$, where $\mathcal{F}(u) = \mathcal{F}_l(u) \cup \mathcal{F}_u(u)$, and $w_{i,u}$ represents the resource overhead (including of transcoding cost $w_i$ and transmission cost $w_l$) per bit for flow $i$ on node $u$. The total resource cost is the sum of overall transcoding and transmission resource consumption. We formulate the joint resource allocation problem of data transmission and online transcoding as the following stochastic optimization problem.

$$\forall t : \mathcal{U}(\bar{\boldsymbol{x}}) = \text{Min}\left(\sum_{u\in\mathcal{V}}\sum_{i\in\mathcal{F}(u)} w_{i,u}\bar{x}_i - \beta\sum_{i\in\mathcal{F}} U(\bar{x}_i)\right) \tag{7a}$$

$$\text{s.t.} \sum_{i\in\mathcal{F}_u(u)} w_{i,u}\bar{x}_i[t] \le c_u[t], \quad \forall u \in \mathcal{V} \tag{7b}$$

$$w_l \sum_{j\in\mathcal{F}_l(u)}\sum_{i\in\mathcal{V}(j)} \bar{x}_i[t] \le c_l[t], \quad \forall l \in \mathcal{L} \tag{7c}$$

$$\bar{x}_i[t] \in [0, x_{max}], \quad \forall i \in \mathcal{C} \tag{7d}$$

where $\beta$ is the weight factor, $U(\bar{x}_i)$ the utility function of flow $i$. $\sum_{u\in\mathcal{V}}\sum_{i\in\mathcal{F}(u)} w_{i,u}\bar{x}_i$ represents the overall consumption of the CLS system. The objective function $\mathcal{U}(\bar{\boldsymbol{x}})$ (7a) is to minimize the sum of all nodes consumption and the negative overall utility. Due to the limited transcoding and link bandwidth resource, the constraint (7b) and (7c) indicate that the transcoding and transmission rate cannot exceed the node and link capacity. The above restrictions also establish the queue stability constraint of node $u$, which means there is a balance between enqueue and dequeue in the long-time term, and the pending time in the queue is bounded. The (7b) and (7c) can be rewritten as $\limsup_{t\to\infty} \mathbb{E}\{\|\boldsymbol{q}_u[t]\|\} < \infty$ and $\limsup_{t\to\infty} \mathbb{E}\left\{\left\|\boldsymbol{q}'_u[t]\right\|\right\} < \infty$, respectively. Therefore, reducing the steady-state queue-length can effectively decrease the access delay of CLS content. Besides, (7d) is the boundary constraints of data transmission rate for different flow $i$, where $x_{max}$ represents the maximum data rate.

## V. Algorithm Design

In this section, we first give a brief introduction of Nesterov's accelerated gradient descent method. Afterwards, to solve the problem (7a), we design an Accelerated Gradient algorithm, which iteratively optimizes the resource allocation of transmission and online transcoding in a distributed manner.

### *A. The Nesterov's AGD Method*

The Nesterov's AGD method [36] is a powerful tool which aims at solving unconstrained minimization problem $\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$, where $f(\cdot)$ is a convex function and twice continuously differentiable. However, our problem (7) is constrained and the convexity of objective function is depend on $U(\cdot)$. To apply the powerful method, we first need to reformulate the problem to unconstrained form, and then set a appropriate function $U(\cdot)$ since the overall network consumption is linearly dependent on $\bar{x}$. Fortunately, several forms such as utility function in [37] can make the objective function satisfy the convex and twice continuously differentiable. We defined the Hessian matrix of $f(\cdot)$ about $\boldsymbol{x}$ as $\boldsymbol{H}(\boldsymbol{x})$ and the Lipschitz constant of the $f'(\cdot)$ as $b$ where $f'(\cdot)$ is the gradient of $f(\cdot)$. Besides, we define two constants $a$ and $b$ that satisfy $a\boldsymbol{I} \leqslant \boldsymbol{H}(\boldsymbol{x}) \leqslant b\boldsymbol{I}$, $\forall\boldsymbol{x}$ where $0 \leqslant a \leqslant b$ and $\boldsymbol{I}$ is the identity matrix. From [38] Chapter 2.2.9 (p.80), the author gives the constant step scheme II of the Nesterov's AGD Method:

*Initialization:* Choose an $n$-dimensional vector $\boldsymbol{\alpha}[0] \in \mathcal{R}^n$ and $\gamma[0] \in (0, 1)$. Define an auxiliary variable $\boldsymbol{\theta}[0]$ and set $\boldsymbol{\theta}[0] = \boldsymbol{\alpha}[0]$, $\xi = \frac{a}{b}$. Step of the $t$-th Iteration ($t \geqslant 0$):

1) Compute $f(\boldsymbol{\theta}[t])$ and $f'(\boldsymbol{\theta}[t])$, and set:

$$\boldsymbol{\alpha}[t+1] = \boldsymbol{\theta}[t] - \frac{f'(\boldsymbol{\theta}[t])}{b}. \tag{8}$$

2) Based on the quadratic equation $\gamma^2[t+1] = \gamma^2[t] - \gamma[t+1]\gamma^2[t] + \xi\gamma[t+1]$, get $\gamma[t+1] \in (0, 1)$, and calculate the value of $\eta[t]$:

$$\eta[t] = \frac{\gamma[t](1-\gamma[t])}{\gamma^2[t] + \gamma[t+1]} \tag{9}$$

3) Compute $\boldsymbol{\theta}[t+1]$ by:

$$\boldsymbol{\theta}[t+1] = \boldsymbol{\alpha}[t+1] + \eta[t]\left(\boldsymbol{\alpha}[t+1] - \boldsymbol{\alpha}[t]\right) \tag{10}$$

According to [38], it shows that for the strongly convex objective function, the Nesterov's AGD method achieves $\left[\mathcal{O}(1/V), \mathcal{O}\left(\sqrt{V}\right)\right]$ utility-delay trade-off which is better than $[\mathcal{O}(1/V), \mathcal{O}(V)]$ performance of traditional queue-based solution [34], Based on that, we consider the stochastic optimization problem (7) which aims at minimizing the overall resource consumption under the queue stability constraints and capacity limitations of node and link. The objective function (7a) is convex and twice continuously differentiable. However, the Nesterov's AGD method is only suitable for optimization of unconstrained problems. Therefore, we need to reshape the form of (7). We defined $\boldsymbol{\psi}$ as the auxiliary weight and build the *min-drift-minus-penalty* expression of (7) shown as follow:

$$\min\left\{V\mathcal{U}(\bar{\boldsymbol{x}}) - \psi_u\left(\sum_{i\in\mathcal{F}_u(u)} w_i\bar{x}_i - c_u[t]\right) - \psi_l\left(w_l\sum_{j\in\mathcal{F}_l(u)}\bar{x}_j - c_l[t]\right)\right\} \tag{11}$$

where $V$ is a nonnegative penalty parameter. It's noted that the *min-drift-minus-penalty* is associate with $\bar{\boldsymbol{x}}$ and $\boldsymbol{\psi}$. Then we can represent the function (11) as $\min g(\bar{\boldsymbol{x}}, \boldsymbol{\psi})$. Thus, according to the equation (8), we can get the first iteration step of the Nesterov's AGD method for our problem $g(\bar{\boldsymbol{x}}, \boldsymbol{\psi})$ as follow:

$$\boldsymbol{\alpha}[t+1] = \boldsymbol{\psi}[t] - \frac{g'(\boldsymbol{\psi}[t])}{b} \tag{12}$$

where $\boldsymbol{\alpha}[t+1]$ is auxiliary variable, $g'(\boldsymbol{\psi})$ is the first-order derivative of $g$ with respect to $\boldsymbol{\psi}$ and $b$ is the Lipschitz constant of $g'(\boldsymbol{\psi})$. According to (11), we have $g'(\boldsymbol{\psi}) = \boldsymbol{w}\bar{\boldsymbol{x}} - \boldsymbol{c}$. For $j \in \mathcal{F}_l(u)$, we get $g'(\psi_l[t]) = w_l\sum_{j\in\mathcal{F}_l(u)}\bar{x}_j[t] - c_l[t]$ where $\bar{x}_j[t]$ is the average send out rate of *Send Queue* for receiver $j$ and $c_l[t]$ is the maximum transmission capacity of link $l$. Because $\bar{x}_j[t]$ and $\frac{c_l[t]}{w_l}$ can be represented by $q[t]$, we have $g'(\psi_l[t]) = -\sum_{j\in\mathcal{F}_l(u)}\{q_j[t+1] - q_j[t]\}^+/w_l = -\sum_{j\in\mathcal{F}_l(u)}\{\Delta q_j[t]\}^+/w_l$. Similarly, we have $g'(\psi_u[t]) = \sum_{i\in\mathcal{F}_u(u)} w_i\bar{x}_i[t] - c_u[t]$. Since $m_i^+[t] = w_i\bar{x}_i[t]$ and $c_u[t]$ is equal to the process rate $m_i^-[t]$, we have $g'(\psi_u[t]) = -\sum_{i\in\mathcal{F}_u(u)}\left\{q_i'[t+1] - q_i'[t]\right\}^+ = -\sum_{i\in\mathcal{F}_u(u)}\left\{\Delta q_i'[t]\right\}^+$. We use the queue variation $\frac{\Delta\boldsymbol{q}[t]}{\boldsymbol{w}}$ to represent the gradient $-\frac{g'(\boldsymbol{\psi}[t])}{b}$ where $\boldsymbol{w}$ is the vector of weight parameters for different nodes or links. Thus, (12) can be rewritten as follow:

$$\boldsymbol{\alpha}[t+1] = \boldsymbol{\psi}[t] + \frac{\Delta\boldsymbol{q}[t]}{\boldsymbol{w}} \tag{13}$$

For convenience, we denote $\boldsymbol{q}_\dagger[t]$ as $\boldsymbol{q}[t]/\boldsymbol{w}$. According to (10) and (13), we can get $\boldsymbol{\psi}[t+1]$ as:

$$\boldsymbol{\psi}[t+1] = \boldsymbol{\psi}[t] + \Delta\boldsymbol{q}_\dagger[t] + \eta\left[\left(\boldsymbol{\psi}[t] + \Delta\boldsymbol{q}_\dagger[t]\right) - \left(\boldsymbol{\psi}[t-1] + \Delta\boldsymbol{q}_\dagger[t-1]\right)\right] \tag{14}$$

where the $\eta$ is the condition number.

**Algorithm 1** Accelerated Gradient Optimization Algorithm

```
Input:
Choose the nonnegative parameters $V$ and $\eta \in [0,1)$;
For each link $l \in \mathcal{L}\cup\mathcal{L}'$, set the $q^\dagger[-1] = q^\dagger[0] = 0$;
Set the initial Nesterov's weights $\psi[-1] = \psi[0] = 0$;
1  foreach time slot $\Delta\mathcal{T}$ do
2      foreach Requesters $v \in \mathcal{C}$ do
3          Obtain the alternative paths $\boldsymbol{P}$ from nearby ESs;
4          Select the transmission path $P_v$:
               $P_v = \arg\min_{P\in\boldsymbol{P}}\left\{\sum_P \psi[t]\right\}$   (15)
5          Establish a delivery path $P_v$ from Provider to Requester;
6      end
7  end
8  while $t \in \mathcal{T}$ do
9      foreach Provider $b \in \mathcal{P}$ do
10         Determine the data rate $\boldsymbol{x}[t]$ for all Requesters;
               $\boldsymbol{x}[t] = \min\left\{\mathcal{U}'^{-1}\left(\sum_P \frac{\boldsymbol{\psi}[t]}{V}\right), \boldsymbol{x}_{max}\right\}$   (16)
11     end
12     foreach node $u \in \mathcal{V}$ do
13         Transcoding rate $\boldsymbol{p}^+[t]$ Decision:
               $\boldsymbol{p}^+[t] = \min\left\{\mathcal{U}'^{-1}\left(\frac{\boldsymbol{\psi}[t]}{V}\right), \frac{c_u[t]}{\boldsymbol{w}}\right\}$   (17)
14         Processing and sending-out rate $x_u^-[t]$ decision:
               $x_u^-[t] = \min\left\{\operatorname{argmax}_{i\in\mathcal{F}(u)}(\psi_i[t]), \frac{c}{w}\right\}$   (18)
15         Update queue $\Delta q_\dagger[t] = q_\dagger[t+1] - q_\dagger[t]$;
16         Update Nesterov's weight $\psi[t+1]$ by (14);
17     end
18 end
```

## B. The AGO Algorithm

We consider CEC cooperation-based CLS system which contains broadcaster, CSs, ESs and viewers with transcoding capability. ESs in this system act as coordinators to maintain the candidate routing table for the *Requesters*. We propose a practical Accelerated Gradient Optimization algorithm to solve the joint resource allocation problem of data transmission and online transcoding. We give the pseudo-code (**Algorithm 1**) of AGO and describe it from the *Requesters* $v \in \mathcal{C}$ perspective.

As the algorithm shows, we set the initial value of the queue-length $\boldsymbol{q}_\dagger[-1] = \boldsymbol{q}_\dagger[0] = 0$ and the Nesterov's weight $\boldsymbol{\psi}[-1] = \boldsymbol{\psi}[0] = 0$ for all node $u \in \mathcal{V}$. Since the nonnegative parameters $V$ and the condition numbers $\eta$ can influence the queue backlog, algorithm optimization and the convergence rate, we omit the discussion of them there and give the details in the next two sections. In every time slot, $\Delta\mathcal{T}$, each *Requester* will get the candidate routing table from the nearest ESs. The table provides alternative routing paths information with the weight vector $\boldsymbol{\psi}[t]$ of nodes and links on each path. The *Requester* will select the path with the minimum sum of weight $\sum_P \psi[t]$ and establish a delivery path with the *Provider*. After the path established, in each iteration, the *Provider* will decide the data rate for the *Requester* according to (16) where $\mathcal{U}'^{-1}(\cdot)$ is an inverse derivative of objective function (11). Besides, each node $u \in \mathcal{V}$ calculates the transcoding portion $\boldsymbol{p}^+[t] = \left\{p_1^+[t], p_2^+[t], \ldots, p_i^+[t], \ldots\right\}$ based on equation (17) where $\boldsymbol{w} = \{w_1, w_2, \ldots, w_i, \ldots\}$ and $i$ is the flow index. Meanwhile, the node will select the flow with maximizing $\psi[t]$ to process and determine the rate by equation (18). In addition, the links and nodes need to update the $\psi[t+1]$ according to the queue variation $\Delta \boldsymbol{q}_{\dagger}[t]$.

### C. Implementation and Complexity Analysis

According to the algorithm description, AGO mainly includes three parts: (1) delivery path selection at *Requester*; (2) the determination of data rate at *Provider*; and (3) the enqueue/dequeue rate decisions of video processing at intermediate nodes. The delivery path selection only needs the Nesterov's weight of nodes and links on the delivery path. Since the weight is maintained by ESs, the *Requesters* can obtain the information directly. Besides, the *Provider* also needs the Nesterov's weight information which can be only easily updated by exchanging information with nodes and links on the communication path. This weight update can be done during data transmission. For example, the *Transcoders* can periodically add the Nesterov's weight to the data packet header and then deliver it to the *Provider*. In addition, for each intermediate node $u$, all updating processes require only local information. Neither global communication between all servers (CSs and ESs) nor interaction with the Crowd is required to perform our algorithm. Hence, AGO algorithm can be deployed in a distributed manner with only on-path information exchange.

The complexity of the first part depends on the number of candidate paths $|\boldsymbol{P}|$, and the number of intermediate nodes on the path that is no more than $|\mathcal{V}|$. Because the delivery path selection requires the *Requester* to calculate the sum of Nesterov's weight of all nodes on each delivery path, the space complexity and time complexity at *Requester* side are both $\mathcal{O}(|\boldsymbol{P}| \cdot |\mathcal{V}|)$. Compare with the number of nodes $|\mathcal{V}|$, the number of hops on the delivery path is relatively small and this processing is triggered every $\Delta\mathcal{T}$ slot. Hence, the process of delivery path selection is lightweight. The complexity of AGO algorithm at *Provider* side is mainly determined by the number of *Requesters* who access the *Provider*. In each iteration, the *Provider* calculates the inverse gradient of the function (11) for each *Requester* with the sum of on-path Nesterov's weight. Since the weight can be obtained by accumulating the weight of nodes on the delivery path, as well as the complexity of equation (15) is $\mathcal{O}(1)$, the processing at the *Provider* side run in $\mathcal{O}(|\mathcal{C}|)$ time and space. In the third part, the intermediate nodes need to calculate enqueue rate $p^{+}[t]$ and schedule sending-out data for each flow, as well as update the virtual queue and Nesterov's weight locally. According to equations (13), (16), and (17), the time complexity at intermediate nodes is $\mathcal{O}(|\mathcal{F}|)$ where $|\mathcal{F}|$ is the total number of flows processed in the nodes. Besides, the intermediate nodes only need to store two time-slot information of queue-length and Nesterov's weight for each flows. The space complexity is also $\mathcal{O}(|\mathcal{F}|)$. To prove the feasibility of our algorithm, we further evaluate the complexity and the bandwidth cost of signalling compare with two mainstream methods [16], [22] in Section VI.

## VI. Main Theoretical Results

In this section, we will reveal four theoretical results of AGO. Firstly, we introduce the queue-length reduction performance, which is associated with the non-negative parameters $V$ and the condition numbers $\eta$.

*Theorem 1: Given $V$ and $\eta \in [0,1)$, the steady queue-length over node $u$ generated by AGO algorithm satisfied the following equation:*

$$\limsup_{t\to\infty} \mathbb{E}\{\|\boldsymbol{q}[t]\|_1\} = \mathcal{O}\left((1+\eta)\sqrt{V}\right) + \mathcal{O}\left((1-\eta)V\right)$$

*Thus, if we choose an $\eta$ which increases speed faster then $1-\mathcal{O}\left(1/\sqrt{V}\right)$, the average steady-state queue-length can be rewritten as:*

$$\limsup_{t\to\infty} \mathbb{E}\{\|\boldsymbol{q}[t]\|_1\} = \mathcal{O}\left(\sqrt{V}\right)$$

**Theorem 1** reveals several essential points. When $\eta$ is a constant value and $V \to \infty$, the queue-length is dominated by $\mathcal{O}((1-\eta)V)$. Hence, the queue backlog is approximately $1-\eta$ fraction of the traditional queue-based algorithm $\mathcal{O}(V)$. Moreover, when $\eta$ variation within the asymptotic regime $1-\mathcal{O}\left(1/\sqrt{V}\right)$, the scale of the queue backlog reaches $\mathcal{O}\left(\sqrt{V}\right)$. The physical implication of the queue length for all nodes and links on the delivery path represents the accumulation of transcoding tasks and the data waiting to be sent, respectively. When the network processing capacity is determined, longer queues may result in more processing time, which also means a large delay for viewers to access the CLS content [25], [39]. Therefore, the reduction of queue length is one way to reduce the queue time of transmission and transcoding. The accelerated gradient approach can effectively reduce the mean of queue length $\mathcal{O}\left(\sqrt{V}\right)$, so it is very suitable for CLS.

Next, we discuss the optimality of our algorithm. According to equation (11), we define the optimal data transmission rate of our problem as $\boldsymbol{x}^* = [x_1^*, x_2^*, \ldots]$ and the AGO algorithm solution as $\boldsymbol{x}^\infty = [x_1^\infty, x_2^\infty, \ldots]$. Besides, we define the gap between AGO and optimal solution in terms of $\boldsymbol{x}$ and $\mathcal{U}(\cdot)$ as $G_x = \|\boldsymbol{x}^\infty - \boldsymbol{x}^*\|$ and $G_u = U(\boldsymbol{x}^\infty) - U(\boldsymbol{x}^*)$ respectively.

*Theorem 2: Under the iteration of* (14)(16)(17)(18)*, the rate gap $G_x$ is bounded by $\mathcal{O}\left(1/\sqrt{V}\right)$. Meanwhile, the utility gap $G_u$ is bounded by $\mathcal{O}(1/V)$. These imply that $\boldsymbol{x}^\infty \to \boldsymbol{x}^*$ asymptotically as $V$ increases.*

**Theorem 2** shows that the optimality of AGO is independent of $\eta$. However, $V$ affects both the optimality and queue length of AGO. Choosing a larger $V$ can obtain a better result but will increase the queue length. Since our algorithm can provide an $\mathcal{O}\left(1/\sqrt{V}\right)$ near optimal service rate, for our dual-queue model, it means that our algorithm can make full use of the system resource and quickly process the backlog data of the transmission or transcoding queues in the CLS system. The last theoretical result of our algorithm is convergence. We denote the convergence rate by the required number of time-slots to get the $\mathcal{O}\left(1/\sqrt{V}\right)$ near optimal solution.

*Theorem 3: Let $\eta \in [0,1)$ and $V \in (\Omega, \infty)$. The Hession matrix of the objective function* (11) *for $\boldsymbol{x}^*$ is denoted as $\mathcal{H}^* \in \mathbb{R}^{N\times N}$. We denote $\boldsymbol{\varepsilon}_{\mathcal{H}^*} = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N\}$ as the eigenvalues vector of $\mathcal{H}^*$. Then, $E\{\boldsymbol{x}[t]|\boldsymbol{w}[t]\}$ satisfies linear convergence* *with a coefficient $\mathcal{O}_R(V,\eta)$:*

$$\mathcal{O}_R(V,\eta) \le \frac{1}{2}\max_{\varepsilon_i^*,\forall i}\left[\left|(1+\eta)\left(1-\frac{\varepsilon^*}{V}\right) \pm \sqrt{(1+\eta)^2\left(1-\frac{\varepsilon_i^*}{V}\right)^2 - 4\eta\left(1-\frac{\varepsilon_i^*}{V}\right)}\right|\right] < 1$$

The optimization of convergence is $\mathcal{O}_R(V,\eta) = \frac{(\sqrt{\xi}-1)}{\sqrt{\xi}}$, where $\xi = a/b$ is the condition number of Hession matrix $\mathcal{H}$. According to [40], the optimal convergence rate $\mathcal{O}'_R$ of the queue-based algorithm is $\sqrt{\xi-1}/\sqrt{\xi+1}$. This implies AGO algorithm is always superior to the traditional queue-based algorithm in the rate of convergence. Faster convergence rate can improve the adaptability of the algorithm to the dynamic environment, thus improving the quality of CLS. **Theorem 1** only guarantees the performance of long-term average queue-length, and do not consider the variation and stability of queue backlog. We give **Theorem 4** that shows the AGO algorithm can provide finite average queue backlog:

*Theorem 4: Suppose there exist a positive value $\epsilon$ for the AGO algorithm in our problem* (7) *that makes the steady queue-length satisfying the following inequality:*

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\left\{\|\boldsymbol{q}[\tau]\|_1\right\} \le \frac{K+V\mathcal{U}_{max}}{\epsilon}$$

*where $K = \sum_i \mathbb{E}\left\{\frac{(x_i^+[t])^2+(x_i^-[t])^2}{2}\right\}$ and $\mathcal{U}_{max}$ is the upper bound of utility function (7a).*

According to the proof of **Theorem 4**, there always exist a positive $\epsilon$ in the feasible region that makes AGO algorithm satisfies $\mathbb{E}\left\{\sum_i\left[x_i^-[t]-x_i^+[t]\right]\right\} \ge \epsilon$. Based on equations (16) (17) (18), $x_i^-[t]$ is related to the capacity of links and nodes in CLS system and $x_i^+[t]$ depends on the optimality of the AGO algorithm. Since **Theorem 2** shows that the rate gap $\|\boldsymbol{x}^\infty - \boldsymbol{x}^*\|$ of AGO algorithm is bounded by $\mathcal{O}\left(1/\sqrt{V}\right)$, and thereby the rate $\boldsymbol{x}[t]$ of our solution is $\mathcal{O}\left(1/\sqrt{V}\right)$ near optimal. Thus, the rate gap and constraints (7b) (7c) confirm that the $q[t]$ generated by the AGO algorithm is strong stability and the upper bound of the average queue-length is $\frac{K+V\mathcal{U}_{max}}{\epsilon}$.

Due to space limitations, we omit the proofs of Theorems 1-4, which can be found in the supporting document.

## VII. Performance Evaluation

In this section, we introduce the datasets we used, experimental scenario and parameter settings. Then we analyse the numerical results of our method compared with Content Delivery Network-based (CDN-based) [16] and Crowd-based [22] solutions. Finally, we evaluate the performance of our solution in a prototype system.

### A. Datasets and Experiment Setup

We conduct a series of simulations to demonstrate the validity of the theoretical result and realise a prototype implementation based on an open-source framework **srs** [41] for further verification. To better evaluate the effectiveness of our approach, we use a real-world dataset [42] that crawled from the official APIs [43] to reconstruct a more realistic evaluation environment. The dataset contains trace records of more than 1.5 million broadcasters and 9 million streams from Feb. 1st to 28th, 2015. Each trace records every 5min including stream ID, source resolution, stream start/end time, the number of viewers, and so on. Since the channel streams traces with too few viewers often has small variation on the number of audiences, we selected the traces with around 1000 peak viewers as driving data to generate requests for our simulation.

To evaluate the numerical performance of our AGO algorithm, we consider a simulation scenario similar to Fig. 1 which includes one **C**loud **S**erver (CS), eight **E**dge **S**ervers (ESs), and 1000 viewers. We divide the scenario into nine regions. One is equipped with a CS and the other with one ESs in each. The ESs are connected to the CS via 300Mbps wire link and communicate with viewers within its coverage through 100Mbps shared wireless link. In addition, we set up 800 mobile nodes and 200 fixed network nodes as broadcasters and viewers. Mobile nodes can communicate with ESs through wireless links and directly connect to other mobile nodes by **D**evice-to-**D**evice (D2D) within their coverage. We set the maximum D2D bandwidth to 10Mbps. Besides, all fixed nodes are connected to CS or ESs with 10Mbps wire link. We set the number of CLS channel to 40, and each channel has six video resolution. According to [16], we set up the cost of bandwidth and transcoding at different resolution as shown in Table II, which are measured by Twitch's official video statistic tool and AWS c4.8xlarge instance, respectively. Besides, we list some important simulation parameters in Table III.

TABLE II
BANDWIDTH REQUIREMENT AND TRANSCODING COST

| resolution | 1080p 60fps | 1080p | 720p 60fps | 720p | 480p | 360p |
|---|---|---|---|---|---|---|
| bandwidth (Mbps) | 5.86 | 4.45 | 2.75 | 1.93 | 1.10 | 0.52 |
| transcoding (vCPU use) | 454% | 333% | 210% | 142% | 81.6% | 50.5% |

TABLE III
PARAMETER SETTING FOR NUMERICAL SIMULATION

| Parameters | Values |
|---|---|
| Number of CS, ESs and Viewers | 1,8,1000 |
| Number of Fixed Viewers and Mobile Viewers | 200,800 |
| Bandwidth of Wire Link Between CS and ESs | 300Mbps |
| Bandwidth of Wire Link Between Servers and Viewers | 10Mbps |
| Maximum Bandwidth of ESs' Sharing Link | 100Mbps |
| Maximum Bandwidth of D2D Link | 10Mbps |
| Number of CLS channel | 40 |
| Number of resolution for each channel | 6 |
| Maximum Transmission Unit | 1500B |
| Total Time-Slot of Simulation | 1000, 2000 |
| Weight of Transcoding Cost at ESs, CS and Viewers | 5,3,1 |

### B. Methodology and Numerical Result

We use the dataset to drive our simulation and evaluate the performance of the steady-state queue-length, convergence rate

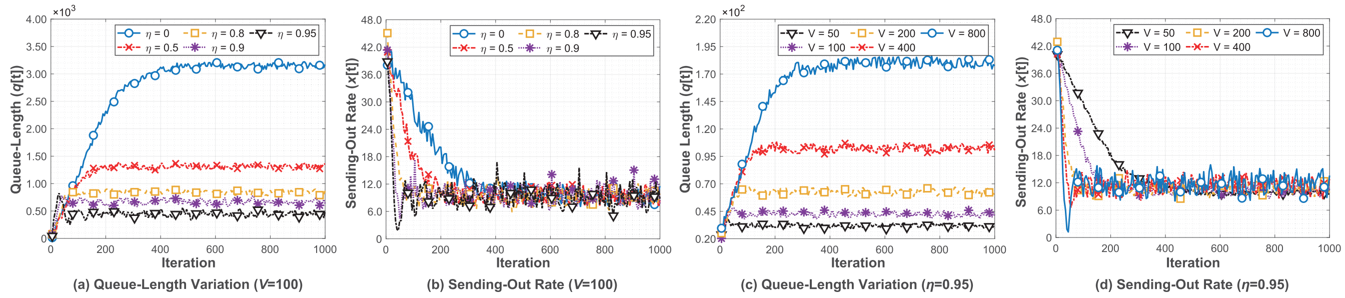


Fig. 3. Comparison of main theoretical results. (a) The queue $q[t]$ variation under different $\eta$ (0, 0.5, 0.8, 0.9, 0.95) with $V = 100$; (b) The convergence of average sending-out rate $x[t]$ of different $\eta$ with $V = 100$; (c) The queue $q[t]$ variation under different $V$ (50, 100, 200, 400, 800) with $\eta = 0.95$; (d) The convergence of average sending-out rate $x[t]$ of different $V$ with $\eta = 100$;.

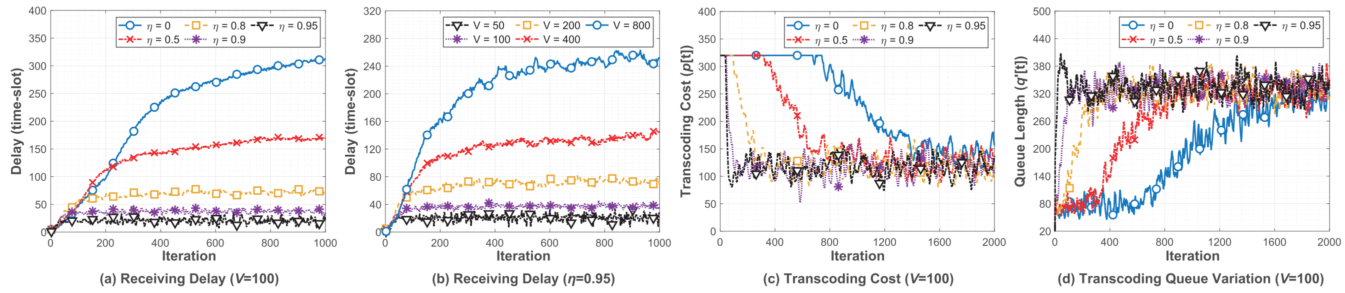


Fig. 4. Comparison of main theoretical results. (a) The receiving delay under different $\eta$ (0, 0.5, 0.8, 0.9, 0.95) with $V = 100$; (b) The receiving delay of different $V$ (50, 100, 200, 400, 800) with $\eta = 0.95$; (c) The transcoding cost $p[t]$ under different $\eta$ (0, 0.5, 0.8, 0.9, 0.95) with $V = 100$; (d) The queue variation $q'[t]$ under different $\eta$ with $V = 100$;.

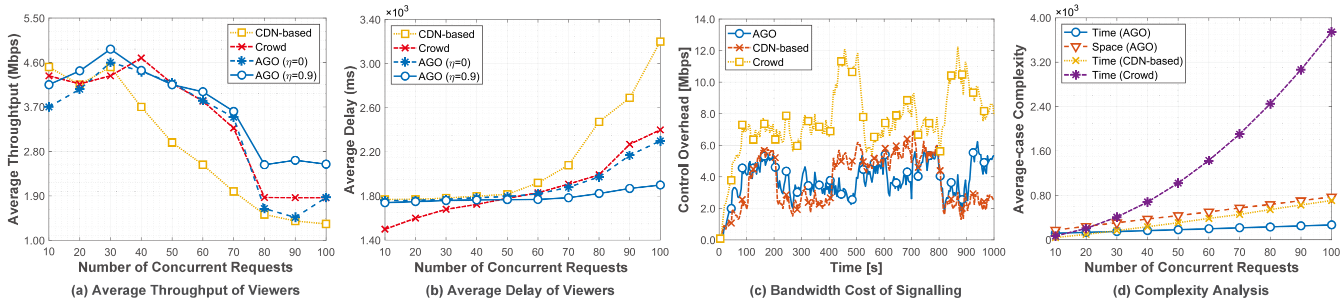


Fig. 5. Comparison of main theoretical results. (a) The variation of average throughput with different number of concurrent requests; (b) The average delay of viewers in different numbers of concurrent requests; (c) Bandwidth cost of signalling with different number of concurrent requests; (d) The complexity analysis of the three solutions.

and optimality with different $\eta$ and $V$. Fig. 3 (a) presents the queue length for one of transmission link in the edge servers with various $\eta$. We let $V = 100$ and increase $\eta$ from 0 to 0.95. We can see that when $\eta$ is equal to 0, 0.5, 0.8, 0.9 and 0.95, the steady-state queue-length is about 312, 130, 81, 62 and 45. According to **Theorem 1**, the stable theoretical value of queue length satisfies the $1-\eta$ reduction, which confirms the $\mathcal{O}\left((1-\eta)\,V\right)+\mathcal{O}\left((1+\eta)\sqrt{V}\right)$ result. Furthermore, Fig. 3 (b) shows that the receive rate initially decreases and then stabilizes to the same value as the iteration. This proves that the optimality of our algorithm is independent of $\eta$. Besides, the convergence rate initially increases and then decreases as $\eta$ varies from 0 to 0.95. Fig. 3 (c) shows the relationship between the queue-lengths and $V$ with $\eta = 0.95$. As $V$ increases from 50 to 800, the queue-length increases proportionately and the convergence speed decreases, which satisfies **Theorem 2**. From Fig. 3 (d), the receive rate of different $V$ converges to the optimal value, and the larger $V$ corresponds to smaller fluctuations. We also test the numerical result of receiving delay for viewers under different $\eta$ and $V$ which are shown in Fig. 4 (a) and Fig. 4 (b). Fig. 4 (a) shows that as the $\eta$ increases, the receiving delay decreases accordingly. This was following the expectation since larger $\eta$ correspond to a lower queue-length and faster convergence (Converge to the optimal data sending rate). This result also reveals that our algorithm with large $\eta$ can quickly converge to the optimal transmission rate, thereby reducing the transmission queue and providing lower delay. By the same token, a lower receiving delay can be achieved with smaller $V$ since the queue-length increases linearly with $V$. Although the converge rate decreases with $V$, the rate is bound by $\frac{\sqrt{\xi}-1}{\sqrt{\xi}}$, see **Theorem 3**. Thus, the receiving delay is dominated by queue-length variation under different $V$. Moreover, we investigate the variation of transcoding queue and the sum of overhead and of the ESs. We can see from Fig. 4 (c) and Fig. 4 (d) that, as the $\eta$ increases, the convergence rate increases and the cost decreases. Different from the transmission, the transcoding process has more violent jitter of the stable-state queue-length. Besides, as shown in Fig. 4 (c), the larger $\eta$, the less transcoding overhead. This is because faster convergence allows the CLS system to reach optimality quicker. Different from the transmission, the length of transcoding queues with different $\eta$ (0, 0.5, 0.8, 0.9, 0.95) converge to the same value. The reason for this phenomenon is that when the nodes on the delivery path are in a stable state, the optimal transcoding tasks allocation is determined accordingly. Similarly, when the algorithm has a faster convergence rate, the transcoding queue of the system can reach a steady state faster.

We compared our solution with two mainstream solutions: 1) Content Delivery Network based architecture [16] (CDN-based): for this architecture, the video transcoding is mainly performed in the CS and ESs. Viewers can access the CLS from the CS and ESs. 2) Crowd cooperation-based architecture [22] (Crowd): this architecture mainly schedules the transcoding task to CS, ESs and stable viewers. When the viewer watches the video long enough, it will be selected as stable viewers. We set the threshold to 5 minutes. This strategy mainly uses the resources of ESs and Crowd, and the CS act as a backup resource. We analyze the **A**verage **T**hroughput (AT), the **A**verage **D**elay (AD) of all viewers and the resource overhead of transmission and transcoding under different number of concurrent viewers in the steady-state network. Fig. 5 (a) illustrates the trend of AT for four strategies (CDN-based, Crowd, $\eta = 0$ and $\eta = 0.9$) as the number of concurrent requests increasing. The CDN-based solutions start at a plateau, then decreases as the number of concurrent requests reaches 30. In Crowd and our approach ($\eta = 0$ and $\eta = 0.9$), as the requests increasing, AT first experiences a brief increase, then quickly reaches a peak, then begins to decline, and finally gradually reaches a plateau. The decreasing trend is due to the computing resources, which is caused by the increasing number of arrival requests. Because the backhaul bandwidth is the bottleneck in our experiment settings, the CDN-based solution has the fastest descent speed and the worst performance under a large number of concurrent viewers. The reason why the Crowd and our approach has a temporary increase at the beginning is that the use of edge resources offloads backhaul network traffic. So some backhaul bandwidth resources are idle when the number of concurrent viewers is less than 30. When the number reaches 80, our solution increases AT by 50% and 22.7% compared with the CDN-based and Crowd solution, respectively. Besides, compared with the CDN-based and Crowd solutions, AGO reduces the AD by 38.5% and 16.7%, as shown in Fig. 5 (b). Besides, we count the average bandwidth cost per second of signalling used to optimize transmission and transcoding in CLS system as the **C**ontrol **O**verhead (CO). As Fig. 5 (c) shows, the CO of all three solutions experiences a sharp growth and then reaches a phase with stochastic fluctuation. In AGO, the CO is mainly the traffic of the candidate routing table and Nesterov's weights information of nodes and links. In the CDN-based solution, The signalling overhead is mainly caused by the communication between different servers (CS and ESs). In the Crowd solution, the regional Data Center needs to maintain the viewing state of the Crowd and allocate the transcoding task, which requires to monitor viewer status continuously. Our method is superior to the Crowd solution. Although the CDN-based solution has better CO performance than AGO, this comes at a price in terms of higher complexity and video transmission and transcoding cost. We also estimate the average-case complexity of three solutions. According to Fig. 5 (d), the time complexity is the best among the three solutions. Since the CDN-based solution needs to traverse the viewer-server pairs to realize the cloud-edge selection for all viewers, the complexity is $\mathcal{O}(|\mathcal{C}||\mathcal{S}|)$ where $|\mathcal{C}|$ and $|\mathcal{S}|$ are the number of viewers and cloud-edge servers, respectively. In our simulation, the number of cloud-edge servers is fixed (nine) and the time complexity increases linearly with the number of concurrent requests. The crowd solution selects stable viewers as transcoders to offloading transcoding tasks for Cloud. The transcoder selection needs to traverse all viewers to elect a certain number of candidates. This processing is similar to a sorting algorithm and the worst-case time complexity is $\mathcal{O}(|\mathcal{C}|log(|\mathcal{C}|))$. Besides, the complexity of AGO is linearly increasing with the number of concurrent requests. Note that our algorithm requires more units of storage resource than time since it needs to store two-slot queues and Nesterov's weight information. Fig. 6 shows the comparison of our approach ($\eta = 0.9$) with CDN-based and Crowd in term of **A**verage **T**ransmission **C**ost (ATC) and **A**verage trans**C**oding **C**ost (ACC) for each node in CLS system under different arrival rate of viewers. In the beginning, compared with the CDN-based solution, our approach has a higher transcoding cost. However, as the arrival rate grows, our approach has an increasing advantage in transmission consumption. Compared with the Crowd solution, our approach has dominance in both transmission and transcoding overhead.

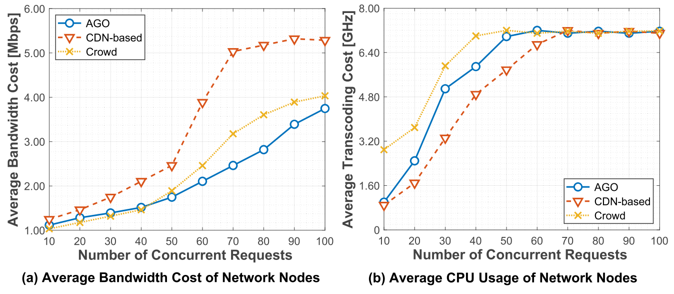


Fig. 6. Resource cost of three solutions ($\eta = 0.9$, $V = 100$) (a) Average bandwidth cost of the system nodes (b) Average CPU usage of the nodes.

## C. Evaluation Over System Prototype

We build a prototype system based on an open-source framework **srs** [41] and evaluate the performance of CDN-based, Crowd and AGO. The topology of our system is shown in Fig. 7, including one broadcaster, one CS, two ESs, two fix viewers and two clusters of mobile viewers.

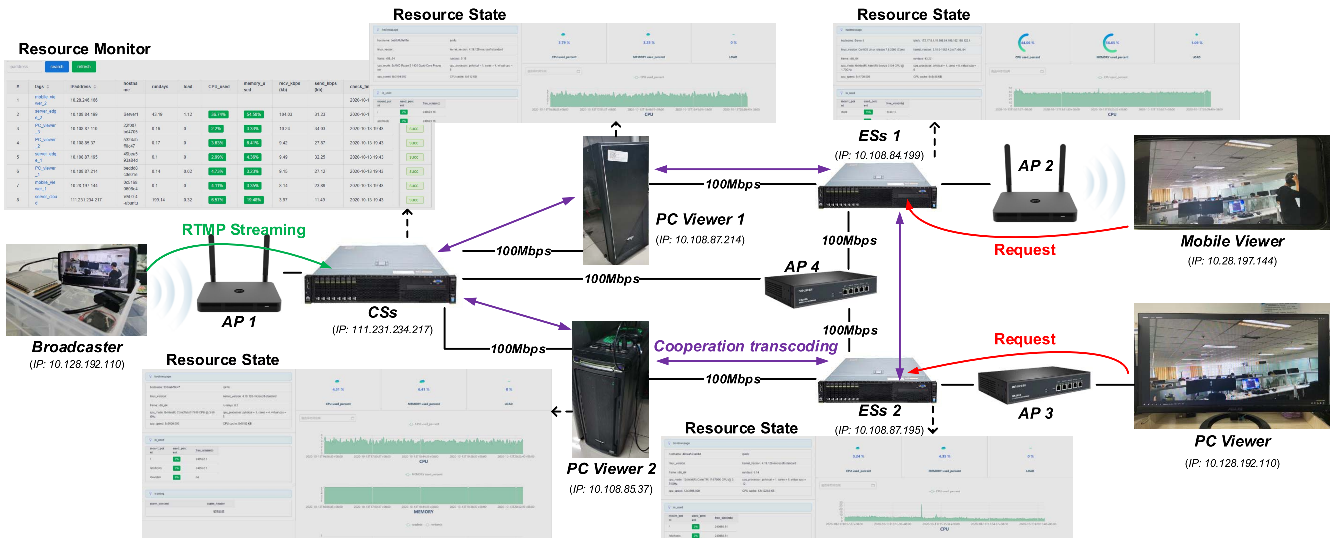


Fig. 7. The topology diagram of the prototype system.

We rent ecs.g5.2xlarge (Intel Xeon Platinum 8163, Eight-Core 2.5Ghz/32GB) of aliyun as CS. Further, we used two servers (Xeon Bronze 3104, Six-Core 1.7Ghz/32GB) to deploy ESs and set up two virtual containers (Intel i7-7700k, Quad-Core 4.2Ghz/16GB and AMD Ryzen 5 Quad-Core 3.2GHz/16GB) as fix viewers through Docker [44]. All of our experiments are conducted on the operating system, which is Centos 7. We use **yasea** [45] (an **R**eal-**T**ime **M**essaging **P**rotocol (**RTMP**) live streaming client for Android) to push the streaming from the broadcaster (HUAWEI Mate 20 pro) to CS through the wireless network. When the viewers request a livecast service, they need to request the ESs first and decide the transcoding path. After that, the broadcasters will upload the content to CS through **RTMP**. The ESs and CS will use **FFmpeg** [46] to transcode the **RTMP** streaming into required resolution. For CDN-based solution, the CS complete most of the transcoding and then push the contents to ESs. Viewers can get CLS from all the servers. Besides, the transcoding is mainly processed by the ESs and fixed viewers in Crowd solution. When Crowd resources are insufficient for the transcoding, CS will be used.

Fig. 8 illustrates the architecture of our prototype system consists of four main modules included a resource monitoring module, a streaming module, a scheduling module, and a web-based video player. The resource monitoring module is responsible for monitoring the resource status, such as CPU usage, memory, network I/O, etc. of each host. The collected resource status will be periodically fetched by the scheduling module with a submodule called ResMonitor. Further, the ResMonitor translates the status into virtual queues and normalizes them as Nesterov's weight input to the srsController, another submodule of the scheduling module. With the srsController, the scheduling module can control transcoding task assignment among different hosts and user streaming access by using **FFmpeg** in the streaming module. We modified the source codes of **srs** and built the streaming module that supports the control of the scheduled module by local interface files. To achieve fast and flexible deployment at different devices,

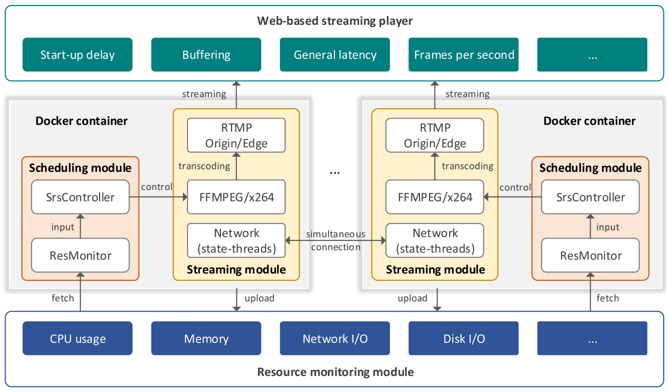


Fig. 8. illustration of the prototype system architecture.

we use virtualization technology to package the streaming module and the scheduling module into a **Docker** container [44] as well as open source on a public **Docker** hub [47]. To monitor the video playback status, we also deployed a Web-based streaming player similar to srsPlayer [48].

We used the prototype to compare the performance of three methods on buffering time, average download delay, resource efficiency and Quality of Experience. We set the start-up delay of the livecast service to 5 seconds and **G**roup **o**f **P**icture (GoP) to 2 seconds. More details of experimental parameters are shown in Table IV. In prototype tests, we first evaluate the download delay of the three solutions. As shown in Fig. 9, we test the average/maximum/minimum GoP download time of accessing the content in about a one-hour monitoring and the delay variation under a different number of concurrent viewers. The transparent blocks (blue/red/yellow) represent the maximum-minimum delay interval of the three solutions, respectively. With the increasing of concurrent viewers, the average download time of the three methods shows an increasing trend. We can see that our solution has the slowest increase in latency, which indicates our solution has better performance

TABLE IV
PARAMETER SETTING FOR SYSTEM PROTOTYPE

| Parameters | Values |
|---|---|
| $\eta$, $V$ | 0.95, 100 |
| Group of Picture | 2s |
| Start-up Delay for Buffering | 5s |
| Playback Buffer Length | 5s |
| Duration of a Time-Slot | 60s |
| Bandwidth of Wire Link | 100Mbps |
| Maximum Bandwidth of Wireless Link | 400Mbps |
| Operating Frequency of Access Point | 5GHz |
| Maximum Interface Queue Length | 100 packets |
| Frame Duration | 16ms |
| Application-Layer Protocol | RTMP |
| Codec Library/Transcoder | FFMPEG/x264 |

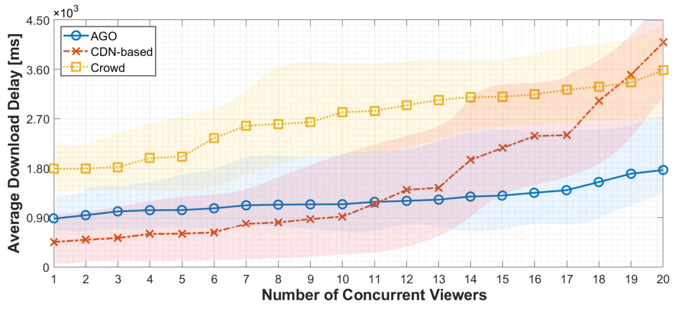


Fig. 9. Average download delay of GoP under the various number of concurrent viewers in three solutions.

in terms of system capacity. Although the CDN-based solution has the lowest delay when the number of concurrent viewers is less than 11, as the number of viewers grows continuously, its access delay increases sharply. In addition, our approach has better delay performance compare with the Crowd and is the only solution that can provide download delay less than 2s with 20 concurrent viewers. According to [49]–[51], we compare QoE of three solutions with three main indicators including latency [s], stalling time [s] and buffering time [s] and the QoE score transform function are shown as following:

$$QoE_k = 1 - a_k \times \frac{r_O^k}{r_M^k}, \quad k \in K \tag{19}$$

where $K = \{1, 2, 3\}$ represent the index set of latency, stalling and buffering time. $a_k$ is the weight of different indicators. $r_O^k$ and $r_M^k$ are the observed value and the historical worst value of indicator $k$, respectively. We set $a_1 = 10\%$, $a_2 = 60\%$ and $a_3 = 30\%$ [51]. In Fig. 10 (a) we examine average throughput, latency, buffering time and stalling time on our prototype system with 20 concurrent requests. Fig. 10 (b) shows the variation of integrated QoE and it can see that the QoE score of all three methods goes through a process of decreasing, increasing, and finally stabilizing. Our approach is the first to reach a stable stage and has better performance than the other two methods. Also, we use the python package **psutil** [52] to get the resource usage of Linux server, including CPU utilization [%] of each nodes and import/export bandwidth [kbps] of CS and ESs. As Fig. 11 (a) shows, compared with CDN-based solutions, our approach reduces the network traffic by about 35.9%. Our method can allocate transcoding tasks

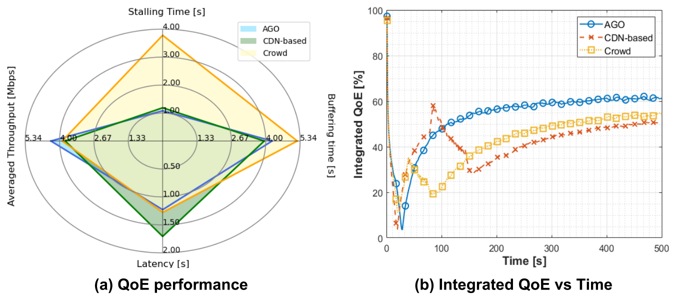


Fig. 10. QoE performance evaluation (a) different QoE performance indicators (b) the variation of integrated QoE over time.

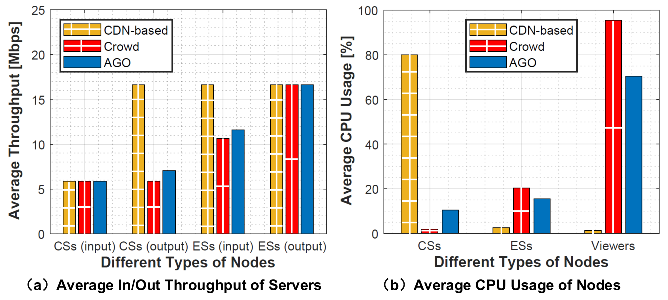


Fig. 11. Resource usage of the network nodes (a) average in/out throughput of Cloud and Edge servers (b) average CPU utilization of different nodes.

effectively, which saving CPU utilization by about 15% and 25% compared to the other two solutions respectively.

## VIII. CONCLUSION

In this paper, we propose a novel augmented queue-based model for CEC cooperation-based CLS systems, which cleverly combines transcoding and transmission by two virtual queues to achieve low-delay and resource-efficient CLS. We further analyze the queue variation of the augmented queue and formalize the resource allocation problem of transcoding and transmission. We design a distributed algorithm called AGO based on accelerating gradient for the resource allocation problem and present four theoretical results of the AGO algorithm. Theoretical results show that, compared with the traditional queue-based solution, AGO has better performance in queue length and convergence. We use real-world data set to design simulation and verify the theoretical results. Besides, compared with two current mainstream solutions through simulation and prototype test, experimental results show that our approach provides a significant improvement in resource efficiency and service quality.

## REFERENCES

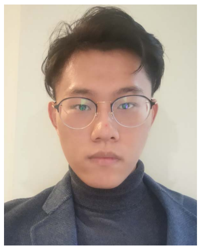
[1] *Twitch*. Accessed: Jun. 2011. [Online]. Available: https://www.twitch.tv/
[2] *Youtube-Live*. Accessed: Feb. 2017. [Online]. Available: https://tv.youtube.com/
[3] *Twitchtracker*. Accessed: Jan. 2019. [Online]. Available: https://twitchtracker.com/statistics
[4] *Youtube Stat*. Accessed: Oct. 2020. [Online]. Available: https://www.omnicoreagency.com/youtube-statistics/
[5] F. Chen, C. Zhang, F. Wang, and J. Liu, "Crowdsourced live streaming over the cloud," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 2524–2532.

[6] J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, and H. Zhang, "CFA: A practical prediction system for video qoe optimization," in *Proc. 13th Usenix Conf. Netw. Syst. Design Implement.*, 2016, pp. 137–150.
[7] F. Wang, J. Liu, M. Chen, and H. Wang, "Migration towards cloud-assisted live media streaming," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 272–282, Feb. 2016.
[8] R. Aparicio-Pardo, K. Pires, A. Blanc, and G. Simon, "Transcoding live adaptive video streams at a massive scale in the cloud," in *Proc. 6th ACM Multimedia Syst. Conf.*, Mar. 2015, pp. 49–60.
[9] Y. Zheng, D. Wu, Y. Ke, C. Yang, M. Chen, and G. Zhang, "Online cloud transcoding and distribution for crowdsourced live game video streaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 8, pp. 1777–1789, Aug. 2017.
[10] B. Yan *et al.*, "LiveJack: Integrating CDNs and edge clouds for live content broadcasting," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 73–81.
[11] C. Dong, Y. Jia, H. Peng, X. Yang, and W. Wen, "A novel distribution service policy for crowdsourced live streaming in cloud platform," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 2, pp. 679–692, Jun. 2018.
[12] C. Dong, W. Wen, T. Xu, and X. Yang, "Joint optimization of data-center selection and video-streaming distribution for crowdsourced live streaming in a GEO-distributed cloud platform," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 2, pp. 729–742, Jun. 2019.
[13] L. Wei, J. Cai, C. H. Foh, and B. He, "QoS-aware resource allocation for video transcoding in clouds," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 1, pp. 49–61, Jan. 2017.
[14] G. Gao, Y. Wen, and C. Westphal, "Dynamic priority-based resource provisioning for video transcoding with heterogeneous QoS," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 5, pp. 1515–1529, May 2019.
[15] M. Ma *et al.*, "Characterizing user behaviors in mobile personal livecast: Towards an edge computing-assisted paradigm," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 14, no. 3s, pp. 1–24, Aug. 2018.
[16] H. Pang *et al.*, "Optimizing personalized interaction experience in crowd-interactive livecast: A cloud-edge approach," in *Proc. 26th ACM Int. Conf. Multimedia*, Oct. 2018, pp. 1217–1225.
[17] R.-X. Zhang *et al.*, "Livesmart: A QoS-guaranteed cost-minimum framework of viewer scheduling for crowdsourced live streaming," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 420–428.
[18] R.-X. Zhang *et al.*, "A practical learning-based approach for viewer scheduling in the crowdsourced live streaming," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 16, no. 2s, pp. 1–22, Jul. 2020.
[19] Q. He, C. Zhang, X. Ma, and J. Liu, "Fog-based transcoding for crowdsourced video livecast," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 28–33, Apr. 2017.
[20] Y. Zhu, J. Liu, Z. Wang, and C. Zhang, "When cloud meets uncertain crowd: An auction approach for crowdsourced livecast transcoding," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 1372–1380.
[21] Q. He, C. Zhang, and J. Liu, "CrowdTranscoding: Online video transcoding with massive viewers," *IEEE Trans. Multimedia*, vol. 19, no. 6, pp. 1365–1375, Jun. 2017.
[22] Y. Zhu, Q. He, J. Liu, B. Li, and Y. Hu, "When crowd meets big video data: Cloud-edge collaborative transcoding for personal livecast," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 42–53, Jan. 2020.
[23] (2020). *Cisco Annual Internet Report (2018–2023) White Paper*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectiv%es/annual-internet-report/white-paper-c11-741490.html
[24] J. Ren, Y. Zhang, K. Zhang, and X. Shen, "Exploiting mobile crowdsourcing for pervasive cloud services: Challenges and solutions," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 98–105, Mar. 2015.
[25] M. Wang, C. Xu, X. Chen, H. Hao, L. Zhong, and D. O. Wu, "Design of multipath transmission control for information-centric Internet of Things: A distributed stochastic optimization framework," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9475–9488, Dec. 2019.
[26] Z. Wang, L. Sun, C. Wu, W. Zhu, Q. Zhuang, and S. Yang, "A joint online transcoding and delivery approach for dynamic adaptive streaming," *IEEE Trans. Multimedia*, vol. 17, no. 6, pp. 867–879, Jun. 2015.
[27] F. Chiariotti, S. Kucera, A. Zanella, and H. Claussen, "Analysis and design of a latency control protocol for multi-path data delivery with pre-defined QoS guarantees," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1165–1178, Jun. 2019.
[28] L. Cui, D. Su, S. Yang, Z. Wang, and Z. Ming, "TCLiVi: Transmission control in live video streaming based on deep reinforcement learning," *IEEE Trans. Multimedia*, early access, Apr. 6, 2020, doi: 10.1109/TMM.2020.2985631.
[29] H. Pang *et al.*, "Content harvest network: Optimizing first mile for crowdsourced live streaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 7, pp. 2112–2125, Jul. 2019.
[30] S.-G. Chen and Y.-K. Lin, "Search for all minimal paths in a general large flow network," *IEEE Trans. Rel.*, vol. 61, no. 4, pp. 949–956, Dec. 2012.
[31] H.-W. Lee, E. Modiano, and K. Lee, "Diverse routing in networks with probabilistic failures," *IEEE/ACM Trans. Netw.*, vol. 18, no. 6, pp. 1895–1907, Dec. 2010.
[32] Z. Dong *et al.*, "A general analysis framework for soft real-time tasks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 6, pp. 1222–1237, Jun. 2019.
[33] T. X. Tran and D. Pompili, "Adaptive bitrate video caching and processing in mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 9, pp. 1965–1978, Sep. 2019.
[34] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 396–409, Apr. 2008.
[35] Z. Jiang, C. Xu, J. Guan, Y. Liu, and G.-M. Muntean, "Stochastic analysis of DASH-based video service in high-speed railway networks," *IEEE Trans. Multimedia*, vol. 21, no. 6, pp. 1577–1592, Jun. 2019.
[36] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $o(1/k^2)$," *Soviet Math. Doklady*, vol. 27, no. 2, pp. 543–547, 1983.
[37] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Apr. 1998.
[38] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87. Berlin, Germany: Springer, 2013.
[39] G. Carofiglio, M. Gallo, L. Muscariello, M. Papalini, and S. Wang, "Optimal multipath congestion control and request forwarding in information-centric networks," in *Proc. 21st IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2013, pp. 1–10.
[40] A. Eryilmaz and R. Srikant, "Joint congestion control, routing, and MAC for stability and fairness in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1514–1524, Aug. 2006.
[41] *srs*. Accessed: Jan. 2020. [Online]. Available: https://github.com/ossrs/srs
[42] *Clivecast*. Accessed: Feb. 2017. [Online]. Available: https://clivecast.github.io/
[43] *Developer Apis*. Accessed: Jan. 2020. [Online]. Available: http://dev.twitch.tv/
[44] *Docker*. Accessed: Jul. 2016. [Online]. Available: https://github.com/docker
[45] *Yasea*. Accessed: Feb. 2020. [Online]. Available: https://github.com/begeekmyfriend/yasea
[46] *Ffmpeg*. Accessed: Oct. 2019. [Online]. Available: https://ffmpeg.org/
[47] *Prototype*. Accessed: Dec. 2020. [Online]. Available: https://hub.docker.com/repository/docker/buptfzc/prototype
[48] *Srsplayer*. Accessed: Jan. 2013. [Online]. Available: http://ossrs.net/players/srs_player.html
[49] Z. Xu, X. Zhang, and Z. Guo, "QoE-driven adaptive K-Push for HTTP/2 live streaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 6, pp. 1781–1794, Jun. 2019.
[50] C. Xu, M. Wang, X. Chen, L. Zhong, and L. A. Grieco, "Optimal information centric caching in 5G Device-to-Device communications," *IEEE Trans. Mobile Comput.*, vol. 17, no. 9, pp. 2114–2126, Sep. 2018.
[51] H. Wang, K. Wu, J. Wang, and G. Tang, "Rldish: Edge-assisted QoE optimization of HTTP live streaming with reinforcement learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2020, pp. 706–715.
[52] *Psutil*. Accessed: Dec. 2020. [Online]. Available: https://pypi.org/project/psutil/

**Xingyan Chen** received the B.E. degree in applied physics from the College of Science, Beijing University of Posts and Telecommunications (BUPT), in 2016. He is currently pursuing the Ph.D. degree with the Network Architecture Research Center, School of Computing, BUPT, under the supervision of Prof. Changqiao Xu. His research interests include multimedia communications, networked distributed computing, and stochastic optimization.

**Changqiao Xu** (Senior Member, IEEE) received the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences (ISCAS), in January 2009. He was an Assistant Research Fellow and the Research and Development Project Manager with ISCAS from 2002 to 2007. He was a Researcher with the Athlone Institute of Technology and a joint Ph.D. Researcher with Dublin City University, Ireland, from 2007 to 2009. He joined the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in December 2009. He is currently a Full Professor with the State Key Laboratory of Networking and Switching Technology and the Director of the Network Architecture Research Center, BUPT. He has published over 160 technical papers in prestigious international journals and conferences, including IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, IEEE WIRELESS COMMUNICATIONS, *IEEE Communications Magazine*, IEEE/ACM TRANSACTIONS ON NETWORKING, and IEEE TRANSACTIONS ON MOBILE COMPUTING. His research interests include future Internet technology, mobile networking, multimedia communications, and network security. He has served a number of international conferences and workshops as the co-chair and a technical program committee member. He is also serving as the Editor-in-Chief of *Transactions on Emerging Telecommunications Technologies* (Wiley).

**Mu Wang** received the M.S. degree in computer technology from the Beijing University of Posts and Telecommunications (BUPT) in 2015. He is currently pursuing the Ph.D. degree with the Institute of Network Technology, BUPT. His research interests include information centric networking, wireless communications, and multimedia sharing over wireless networks.



**Zhonghui Wu** received the B.E. degree in applied physics (basic science of communication) from the College of Science, Beijing University of Posts and Telecommunications, in 2019. He is currently pursuing the Ph.D. degree with the Network Architecture Research Center under the supervision of Prof. Changqiao Xu. His research interests include multimedia networking and blockchain.

**Lujie Zhong** received the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2013. She is currently an Associate Professor with the Information Engineering College, Capital Normal University, Beijing. Her research interests include communication networks, computer system and architecture, and mobile Internet technology.

**Luigi Alfredo Grieco** (Senior Member, IEEE) received the Dr.Eng. degree (Hons.) in electronic engineering from the Politecnico di Bari, Bari, Italy, in October 1999, and the Ph.D. degree in information engineering from the Universita di Lecce, Lecce, Italy, in December 2003. From January 2005 to October 2014, he held an assistant professor position at the Department of Electrical and Information Engineering (DEI), Politecnico di Bari. From March to June 2009, he has been a Visiting Researcher with INRIA, Sophia Antipolis, France, working on the topic of Internet measurements. From October to November 2013, he has been a Visiting Researcher with LAAS-CNRS, Toulouse, France, working on information-centric networking design of M2M systems. From November 2014 to December 2018, he has been an Associate Professor in telecommunications with DEI, Politecnico di Bari, where he has been a Full Professor in telecommunications since December 2018. He authored around 200 scientific articles published in venues of great renown that gained more than 8000 citations. His current research interests include the Industrial Internet of Things, information-centric networking, and nano-communications. He has been constantly involved as a technical program committee member for many prestigious conferences. Within the Internet Engineering Task Force (Internet Research Task Force), he contributed (as an author of RFC 7554) new standard protocols for industrial IoT applications (new standard architectures for tomorrow ICN-IoT systems). He served as the Editor-in-Chief for the *Transactions on Emerging Telecommunications Technologies* (Wiley) from 2016 to 2019. He also serves as an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, for which he has been awarded as top Associate Editor in 2012 and 2017. Since January 2019, he has been a Founding Member and the Subarea-Chair of the IEEE SIG on Intelligent Internet Edge. In 2020, he has been nominated as a Scientific Coordinator of the IoT4.0 Lab. He is the Founding Editor-in-Chief of the *Internet Technology Letters Journal* (Wiley).