Distributed Resource Allocation in Blockchain-Based Video Streaming Systems With Mobile Edge Computing

Mengting Liu, F. Richard Yu, *Fellow, IEEE*, Yinglei Teng, *Member, IEEE*, Victor C. M. Leung, *Fellow, IEEE*, and Mei Song

***Abstract*— Blockchain-based video streaming systems aim to build decentralized peer-to-peer networks with flexible monetization mechanisms for video streaming services. On these blockchain-based platforms, video transcoding, which is computationally intensive and time-consuming, is still a major challenge. Meanwhile, the block size of the underlying blockchain has significant impacts on the system performance. Therefore, this paper proposes a novel blockchain-based framework with an adaptive block size for video streaming with mobile edge computing (MEC). First, we design an incentive mechanism to facilitate collaboration among content creators, video transcoders, and consumers. In addition, we present a block size adaptation scheme for blockchain-based video streaming. Moreover, we consider two offloading modes, i.e., offloading to the nearby MEC nodes or a group of device-to-device (D2D) users, to avoid the overload of MEC nodes. Then, we formulate the issues of resource allocation, scheduling of offloading, and adaptive block size as an optimization problem. We employ a low-complexity alternating direction method of the multipliers-based algorithm to solve the problem in a distributed fashion. Simulation results are presented to show the effectiveness of the proposed scheme.**

***Index Terms*— Blockchain, video transcoding, mobile edge computing.**

I. Introduction

With the skyrocketing growth of the demands for online streaming services, video streaming platforms like Netflix and YouTube have become popular over the past decade [1]. However, these traditional online streaming platforms suffer from several disadvantages: 1) Low profit for content creators. 2) High monthly charge and low privacy for consumers. 3) Less-than-ideal advertising effects for advertisers. Recently, several startups (e.g., Theta, Stream, Viewly, Livepeer, Flixxo, VirtuTV, etc.) are employing *blockchain technology* to solve these pitfalls. Leveraging blockchain technology, these emerging platforms are able to build decentralized peer-to-peer access networks with flexible monetization mechanisms using *smart contract* [2]. In this new era of blockchain-based video streaming, content creators, consumers, and advertisers can fully support each other without the intervention of any third party.

Nevertheless, blockchain-based video streaming systems are also faced with some challenges. One of the main challenges is video *transcoding*. Similar to the traditional video streaming platforms, the original video contents on these emerging platforms are also required to be transcoded/converted into multiple representations in different bitrates, resolutions, qualities, video codec, etc. to cater to heterogeneous users [3]. However, video transcoding is computation-intensive and time-consuming, which sets an extremely high demand of computational resources [4], [5]. In addition, the block size of the blockchain has significant impacts on the performance of blockchain-based video streaming systems. With a larger block size, more transactions can be included on a block, and the throughput of the blockchain can be higher [6]. However, a larger block size will also introduce higher block propagation delays and higher orphaning probability, which will degrade the performance of the blockchain [7], [8]. Therefore, an adaptive block size is considered as a promising solution to improve the performance of blockchain [8], [9].

Recent advances in *mobile edge computing* (MEC) [3], [10]–[12] can provide possible solutions addressing the above challenges in blockchain-based video streaming systems. With MEC, computation-intensive transcoding tasks can be offloaded to the network edge, which is equipped with computing and storage resources to accelerate video streaming services [3], [13]. Although some excellent works [14]–[18] have been done on MEC-enabled video streaming, most of them are carried out in traditional video streaming systems,

Manuscript received June 18, 2018; revised September 27, 2018; accepted November 18, 2018. Date of publication December 12, 2018; date of current version January 8, 2019. This work was supported in part by the National Key R&D Program of China under Grant 2018YFB1201500, in part by the National Natural Science Foundation of China under Grant 61771072 and Grant 61427801, in part by the Beijing Natural Science Foundation under Grant L171011, in part by the Beijing Major Science and Technology Special Projects under Grant Z181100003118012, and in part by the scholarship from the China Scholarship Council under Grant 201706470059. The associate editor coordinating the review of this paper and approving it for publication was L. Le. *(Corresponding author: Yinglei Teng.)*

M. Liu, Y. Teng, and M. Song are with the Beijing Key Laboratory of Space-ground Interconnection and Convergence, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: liumengting@bupt.edu.cn; lilytengtt@gmail.com; songm@bupt.edu.cn).

F. R. Yu is with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada (e-mail: richard.yu@carleton.ca).

V. C. M. Leung is with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: vleung@ece.ubc.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2018.2885266

where the incentive for MEC nodes to assist video transcoding is missing.

To address this issue, blockchain is widely considered as a promising solution since the distributed feature of blockchain is well suitable to employ MEC in video streaming systems through incentive mechanisms [19]–[23]. However, the distinct characteristics of the blockchain introduce non-trivial challenges to the MEC-enabled video streaming systems. To the best of knowledge, blockchain-based video streaming systems with MEC have not been well studied. The key contributions of this paper are summarized as follows:

- We propose a novel MEC-enabled framework for blockchain-based video streaming. With the trust and traceability features of the blockchain, we design an incentive mechanism to facilitate the collaborations among content creators, video transcoders and consumers, without the intervention of any third party.
- We present a block size adaptation scheme for blockchain-based video streaming, where the block size can dynamically changed to accommodate the time-varying nature of video streaming and its low-latency requirements.
- Unlike the previous works only considering one single computation offloading mode, we consider two offloading modes, i.e., offloading to the nearby MEC nodes or a group of device-to-device (D2D) users, to avoid the overload of MEC nodes.
- We formulate the issues of resource allocation, scheduling of offloading, and adaptive block size as an optimization problem. Then, we employ a low-complexity alternating direction method of multipliers (ADMM)-based algorithm to solve the problem in a distributed fashion. Simulation results are presented to show the effectiveness of the proposed scheme.

The rest of this paper is organized as follows. Related works are discussed in Section II. Section III presents the system model. The offloading framework and incentive mechanism are introduced in Section IV. In Section V, we formulate the resource allocation, scheduling of offloading, and adaptive block size into an optimization problem and adopt an ADMM-based algorithm to solve it. Simulation results are discussed in Section VI. Finally, Section VII concludes this paper.

II. Related Works

This section aims to discuss the related works from two aspects, namely traditional video streaming and blockchain with MEC, which are as follows.

A. Traditional Video Streaming

Recent years have witnessed the growing demands for online streaming services. In 2017, around 69% of world's mobile data traffic in 2017 was video, and this number will increase to reach more than 78% by 2021 [1]. To cater to heterogeneous mobile devices, networks, and user preferences, the source video streams are required to be transcoded or converted into multiple representations, which is computational intensive and time consuming [3]. To solve this problem, a lot of works utilize the advances of MEC to provide good video streaming service. In mobile networks, an MEC enhanced adaptive bitrate (ABR) video delivery scheme is presented to enhance the mobile video service, which combines content caching and ABR streaming technology together [14]. Meanwhile, a novel MEC architecture is designed to enhance the performance of dynamic adaptive streaming over HTTP in mobile networks [15]. To enhance the user Quality of Experience (QoE) against the uneven video quality in highly dynamic networks, a context-aware Q-leaning based video streaming approach with MEC is presented in [16]. For delay-sensitive multimedia Internet-of-Things (IoT) tasks, the authors of [17] propose an edge computing framework to enable cooperative processing among resource-abundant mobile devices, where group formation and video-group matching are considered to maximize the human detection accuracy within the video task's deadline. Additionally, video quality adaptation with radio resource allocation for mobile networks is studied in [18], where MEC and in-network caching techniques are used to enhance the video service. Although a number of works [14]–[18] have studied the issues of video streaming, the incentive for MEC nodes to assist video transcoding is missing in the traditional video streaming scenarios.

B. Blockchain With Mobile Edge Computing (MEC)

To address this issue, blockchain is widely considered as a promising solution to employ MEC for video streaming service with incentive mechanisms. There have been several works focusing on deploying MEC to facilitate wider applications of blockchain to mobile networks. An MEC-enabled framework for mobile blockchain networks is proposed in [19], where mobile users can access and utilize resources from an edge computing service provider to support their blockchain applications. To conduct edge resource allocation in mobile blockchain networks, the authors of [20] present an optimal auction based deep learning architecture to ensure the incentive compatibility and individual rationality. Meanwhile, [21] also puts forward an auction-based edge computing resource allocation for mobile blockchain networks, where the social welfare is maximized while ensuring the truthfulness, individual rationality and computational efficiency. Besides, the optimal pricing-based edge computing resource management is investigated in [22] to support mobile blockchain applications where the mining process can be offloaded to MEC nodes. Moreover, a Stackelberg game is used to model the edge computing resource management and pricing problems for mobile blockchain networks in [23].

While these excellent works [19]–[23] have studied the blockchain with MEC, the issue of how to deploy MEC in blockchain-based video streaming scenarios has not been well investigated. This motivates us to design an MEC-enabled framework for blockchain-based video streaming.

III. System Model

In this section, we first describe the system architecture to provide some necessary backgrounds. Following that, we present the related models adopted in this paper, including

TABLE I
NOTATIONS

Symbol	Definition	Symbol	Definition
λ_a	The density of SBSs	λ_u	The density of nodes
M	The number of SBSs	N	The number of nodes
I_m	The size of the original video stream file released by broadcaster BC_m	V_m	The number of targeted versions of the video released by BC_m
L_{v_m}	The length of video segments of the v_{th} version w.r.t. the video released by BC_m	Ω_{v_m}	The profit for TC_{v_m} by accomplishing transcoding task
φ_{v_m}	The input size of each video segment in task Φ_{v_m}	X_{v_m}	The workload/intensity of task Φ_{v_m}
Q_{v_m}	The number of video segments in task Φ_{v_m}	r_{v_m}/τ_{v_m}	The requirements (required bitrate/delay tolerance) of task Φ_{v_m}
P_S^T/P_S^I	The power consumption of SBSs in active/idle state	P_U^T	The transmit power of nodes
α	The pathloss exponent	B_S/B_U	The available bandwidth for cellular/D2D networks
σ_w^2	The noise power	ϖ_e	The unit price of energy cost
S_B	The adaptive block size	$\boldsymbol{\delta}$	Offloading mode selection indicator
$\boldsymbol{b}$	Spectrum allocation vector	$\boldsymbol{c}$	Computational resource allocation profile

video transcoding model, computation offloading model and network model. For the clarity of the following discussion, the key notations are summarized in Table I.

A. System Architecture

As shown in Fig. 1, we consider an MEC-enabled framework for blockchain-based video streaming systems with adaptive block size S_B, where there are one macro base station (MBS), M small cell base stations (SBSs) and N users. The users, also referred as nodes, have to bond an amount of *token* to join the video streaming system. The SBSs and nodes are distributed according to two independent homogeneous Poisson point processes (HPPPs) with density λ_s and λ_u, respectively. We denote the set of SBSs as $\mathcal{M} = \{1, 2, \ldots, M\}$ and use m to refer to the m-th small cell or SBS. Assuming each node accesses to the nearest SBS, there are N_m nodes in small cell m. An MEC server is placed in the MBS, and all the SBSs are connected to the MBS as well as the MEC server. Note that in this MEC-enabled framework, the SBSs and nodes with certain computation capability can assist the transcoding job. In this sense, the transcoders can offload the transcoding task to either the nearby SBS or a group of D2D nodes.

B. Video Transcoding Model

Video segment is considered as the unit of video streams, which is a time-sliced chunk of multiplexed audio and video [24]. We assume that there is only one broadcaster $BC_m, m \in \mathcal{M}$ at each time slot in each small cell m.[1] After publishing the original streams,[2] each broadcaster BC_m would submit a transcoding job transaction TJ_{v_m} onto the blockchain. We use $\langle I_m, V_m, Q_{v_m}, L_{v_m} \rangle$ to denote TJ_{v_m}, where I_m (in *bit*) is the size of the original video stream file, and $v = 1, 2, \ldots, V_m$ represents V_m different targeted versions with Q_{v_m} sliced video segments in length L_{v_m} (in *s*). As soon as these transactions are mined on the blockchain, the next blockhash will be used to pseudo-randomly select a number of transcoders from the nodes to complete these jobs. Assume that for each transaction $TJ_{v_m}, \forall m, v$, there is only one corresponding transcoder TC_{v_m} chosen to perform the transcoding task [25]. Then we adopt $\langle r_{v_m}, \tau_{v_m} \rangle$ to describe the requirements (bitrate r_{v_m} and delay tolerance τ_{v_m}) of the v_m-th targeted version.

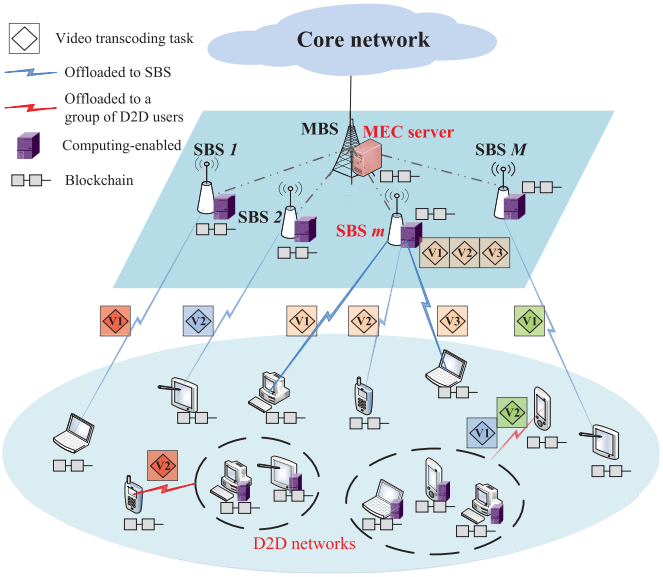


Fig. 1. An illustration of blockchain-based video streaming systems with mobile edge computing (MEC).

C. Computation Offloading Model

For clarity, we use $\Phi_{v_m} = \langle \varphi_{v_m}, X_{v_m}, Q_{v_m}, L_{v_m}, r_{v_m}, \tau_{v_m} \rangle$ to describe the transcoding task w.r.t. TJ_{v_m}, which includes

[1]We assume that there are a number of broadcasters who submit the transcoding request transactions but only one can be answered (i.e., written into the blockchain) every time slot in each small cell, i.e., Time Division Multiple Address (TDMA) mechanism among the broadcasters. It's the block producers in the blockchain system who determine which transaction to be answered (i.e., included in the blocks).

[2]The broadcasters can store the source/transcoded videos in the nearby servers or some blockchain-enabled storage platforms like *Swarm, Storj, Oyster*, etc. For the viewers, the requesting videos can be fetched from their nearby distributed storage nodes.

several main parameters: the input size of video segment φ_{v_m} (in *bit*), workload/intensity X_{v_m} (in *CPU cycles/bit*), the number of video segments Q_{v_m} (in *segment*), the length of video segment L_{v_m} (in *s*), required bitrate r_{v_m} (in *bit/s*) and delay tolerance τ_{v_m} (in *s*). Assuming all the SBSs and nodes have a computation capacity to be a transcoding helper, we consider two offloading modes where the transcoding task is:

1) Offloaded to a Nearby SBS (Mode 0): The transcoder TC_{v_m} offloads the full task Φ_{v_m} to its serving SBS m and the MEC server assigns c_{v_m} CPU cores to SBS m to complete the transcoding job.

2) Offloaded to a Group of D2D Nodes (Mode 1): The transcoder TC_{v_m} divides task Φ_{v_m} into K_{v_m} equal parts $\left\{\Phi_1, \Phi_2, \ldots, \Phi_{K_{v_m}}\right\}$ and separately dispatches them to a group of nearby D2D nodes $U_1, U_2, \ldots, U_{K_{v_m}}$.

We denote $\delta_{v_m} \in \{0,1\}$ as the task offloading decision of $TC_{v_m}, \forall m, v$. Specifically, we have $\delta_{v_m} = 0$ if TC_{v_m} selects *mode 0*, i.e., offloading the whole transcoding task to SBS m. We have $\delta_{v_m} = 1$ if TC_{v_m} chooses *mode 1*, i.e., offloading the task to a group of D2D nodes. So $\boldsymbol{\delta} = \{\delta_{v_m}\}, \forall m, v$ can be considered as the offloading decision profile. Besides, in the case of *mode 0*, we use $\boldsymbol{c} = \{c_{v_m}\}, c_{v_m} \in \{0, 1, \ldots, C\}, \forall m, v$ to denote the computational resource allocation profile, where C is the total number of available CPU cores at the MEC server.

D. Network Model

The channel radio propagation between nodes and SBSs/nodes is assumed to comprise both path loss and Rayleigh fading. Note that in order to study the performance of transcoding task offloading, we only focus on the channel condition between the transcoders and SBSs or D2D nodes. Specifically, the path loss fading of the channel between TC_{v_m} and SBS m (D2D node $U_{k_{v_m}}$) is $r_{m,v}{}^{-\alpha}$ ($l_{m,v,k}{}^{-\alpha}$) where $r_{m,v}$ ($l_{m,v,k}$) represents the distance between TC_{v_m} and SBS m (D2D node $U_{k_{v_m}}$). Meanwhile, the Rayleigh fading of the link between TC_{v_m} and SBS m (D2D node $U_{k_{v_m}}$) is denoted by $h_{m,v}$ ($g_{m,v,k}$) where $h_{m,v}$ and $g_{m,v,k}$ follow an exponential distribution with mean $1/\varsigma$, i.e., $h_{m,v} \sim \exp(\varsigma)$ and $g_{m,v,k} \sim \exp(\varsigma)$. Note that $h_{m,v}, g_{m,v,k}, \forall m, v, k$ are mutually independent with each other. The transmit power density of the SBSs and nodes with are P_S^T and P_U^T, respectively.

After the trancoding job is done, the SBS and D2D nodes need to send the transcoded version of the video segment back to the transcoder through the downlinks. In downlinks, we assume that two separate channels with bandwidth B_S and B_U are assigned for the cellular networks and D2D networks while an orthogonal frequency-division multiplexing access (OFDMA) transmission mechanism is adopted within each small cell. Thus, we use $b_{v_m}^{(0)} \in [0,1]$ and $b_{k_{v_m}}^{(1)} \in [0,1]$ to describe the spectrum allocation in the downlink cellular network and D2D network, respectively. Hence, we have $\boldsymbol{b} = \left\{b_{v_m}^{(0)}, b_{k_{v_m}}^{(1)}\right\}, \forall m, v, k$ as the spectrum allocation profile.

IV. Offloading Framework and Incentive Mechanism

In this section, we first present the offloading framework and the performance analysis for each offloading mode, and then introduce the incentive mechanism for the blockchain based video streaming systems.

A. Offloading Framework

In order to release the burden of transcoders, we utilize the recent advances in MEC by allowing the video transcoding tasks to be offloaded to the nearby SBS (*mode 0*) or a group of D2D users (*mode 1*). To facilitate the offloading framework, we first derive some important performance metrics including output size, delay and energy consumption as follows.

1) Offloaded to a Nearby SBS (Mode 0):

a) Output size: The output size refers to the size of transcoded video segments, which is the key indicator for streaming overhead and can be calculated by the product of bitrates and video segment length as $O_{v_m}^{(0)} = r_{v_m} L_{v_m}$.

b) Delay: The total delay is defined as the time from the arrival of a video stream segment to the completion of transcoding task, which consists of queuing time, transcoding time and the time cost for sending the transcoded version back[3] [1]. For transcoder TC_{v_m} in *mode 0*, the system delay can be expressed by

$$D_{v_m}^{(0)} = D_{v_m}^{(0,t)} + D_{v_m}^{(0,q)} + D_{v_m}^{(0,d)}, \tag{1}$$

where the transcoding delay $D_{v_m}^{(0,t)}$, queuing delay $D_{v_m}^{(0,q)}$ and the delay for sending the transcoded version back $D_{v_m}^{(0,d)}$ are derived as follows.

Before diving into the derivation, we first introduce the queuing model. In this blockchain-based system, we assume the video segments generated from the video stream source are maintained in a queue. Then we model the video transcoding process at SBS m with c_{v_m} CPU cores working in parallel as an M/G/F queue.[4] The arrival of video segments follows a Poisson distribution with rate $\lambda^{(0)}$ while the service time follows a general distribution with mean value $\mu^{(0)}$. Specifically, the queue length ℓ_{v_m} increases by one when a video segment arrives and decreases one after the completion of one video segment's transcoding.

i) Transcoding delay: For transcoder TC_{v_m} who selects *mode 0*, the transcoding time for SBS m allocated with c_{v_m} CPU cores to covert one video segment w.r.t. task Φ_{v_m} is $D_{v_m}^{(0,t)} = \frac{1}{\mu^{(0)} c_{v_m}}$.

ii) Queueing delay: Average queueing delay is considered in this paper, which comprises two parts, i.e., the remaining processing time of the current transcoding tasks at the SBS and the sum of the transcoding time of all the video segments in the queue [4], which can be written as

$$D_{v_m}^{(0,q)} = D_{v_m}^{(0,r)} + \frac{\mathbb{E}(\ell_{v_m})}{\mu^{(0)} \dot{c}_{v_m}}, \tag{2}$$

[3] In this paper, we only consider the time cost for completing the transcoding job after the edge computing nodes (e.g., SBS or D2D nodes) get the original video segments. More general cases can be considered in future works.

[4] Although there is only one broadcaster at each time slot in each small cell according to our assumption, one transcoder can handle several transcoding tasks at one time.

where $D_{v_m}^{(0,r)}$ represents the remaining processing time at SBS m, $\mathbb{E}\left(\ell_{v_m}\right)$ is the average queue length, and $\dot{c}_{v_m}$ is the current CPU resources at SBS m.

According to the Little's formula, we have $\mathbb{E}\left(\ell_{v_m}\right) = \lambda^{(0)}\mathbb{E}\left[D_{v_m}^{(0,q)}\right]$ and $D_{v_m}^{(0,q)} = \frac{D_{v_m}^{(0,r)}}{1-\lambda/\left[\mu^{(0)}\dot{c}_{v_m}\right]}$ with the remaining processing time $D_{v_m}^{(0,r)} = \frac{1}{2}\lambda^{(0)}\left[{\sigma_q}^2 + \frac{1}{\left(\mu^{(0)}\dot{c}_{v_m}\right)^2}\right]$ and the variance of task commences ${\sigma_q}^2$ (Details and proof can be found in [4]).

iii) Delay for sending the transcoded version back: The time cost for SBS m to send the transcoded video segment back to TC_{v_m} is calculated by

$$D_{v_m}^{(0,d)} = \frac{O_{v_m}^{(0)}}{b_{v_m}^{(0)} B_S e_{v_m}^{(0)}}, \tag{3}$$

where $e_{v_m}^{(0)}$ is the spectrum efficiency (SE) of the downlink from SBS m to TC_{v_m}, which can be obtained using stochastic geometry methods [26] as

$$\begin{aligned} & e_{v_m}^{(0)} \\ &= \ln\left(1+\frac{P_S h_{m,v} {r_{m,v}}^{-\alpha}}{{\sigma_w}^2 + I_{v_m}^{(0)}}\right) \\ &= \int_0^{\infty} \exp\left(\begin{array}{l} -\varsigma\beta {r_{m,v}}^{\alpha}{\sigma_w}^2 \\ -2\pi\lambda_S \int_{r_{m,v}}^{\infty}\left(1-\frac{1}{1+\varsigma\beta\left(\frac{r_{m,v}}{z}\right)^{\alpha}}\right) z dz \end{array}\right) d\beta, \end{aligned} \tag{4}$$

where $I_{v_m}^{(0)} = \sum_{x=1,x\neq m}^{M} P_S^T h_{x,v} r_{x,v}{}^{-\alpha}$ is the interference from other SBSs, and ${\sigma_w}^2$ is the power spectrum density of additive white Gaussian noise.

c) Energy consumption: When the transcoder TC_{v_m} selects *mode 0*, i.e., offloading the task to its serving SBS, the total energy consumption to complete the transcoding task is [27]

$$\begin{aligned} E_{v_m}^{(0)} &= \kappa(f_{SBS})^3 D_{v_m}^{(0,t)} + P_S^I D_{v_m}^{(0,q)} + P_S^T D_{v_m}^{(0,d)} \\ &= \kappa(f_{SBS})^3 \frac{1}{\mu^{(0)} c_{v_m}} + P_S^I D_{v_m}^{(0,q)} + P_S^T D_{v_m}^{(0,d)}, \end{aligned} \tag{5}$$

where κ is the computation energy efficiency coefficient, f_{SBS} denotes the CPU-cycle frequency of SBSs, and P_S^I is the power consumption in idle state for SBSs.

2) Offloaded to a Group of D2D Users (Mode 1): In this part, we conduct the performance analysis for *mode 1*, i.e., the transcoding task is offloaded to a group of D2D nodes.

a) Output size: The size of the transcoded version w.r.t. one single part $\Phi_{k_{v_m}}$ can be calculated by $O_{k_{v_m}}^{(1)} = \frac{r_{v_m} L_{v_m}}{K_{v_m}}$.

b) Delay: Differ from *mode 0*, only the transcoding delay and the delay for sending the transcoded version back are considered in *mode 1*, and the queueing delay is left out due to the small computational capacity of each D2D node. For TC_{v_m} in *mode 1*, the total delay is

$$D_{k_{v_m}}^{(1)} = D_{k_{v_m}}^{(1,t)} + D_{k_{v_m}}^{(1,d)}, \tag{6}$$

where the transcoding delay $D_{k_{v_m}}^{(1,t)}$ and the delay for sending the transcoded version back $D_{k_{v_m}}^{(1,d)}$ are specified as follows.

i) Transcoding delay: For the part of video segment $\Phi_{k_{v_m}}$ offloaded to $U_{k_{v_m}}$ by TC_{v_m}, the transcoding delay is $D_{k_{v_m}}^{(1,t)} = \frac{\varphi_{v_m} X_{v_m}}{K_{v_m} f_{k_{v_m}}}$ with the CPU-cycle frequency $f_{k_{v_m}}$ for $U_{k_{v_m}}$.

ii) Delay for sending the transcoded version back: The time cost for $U_{k_{v_m}}$ to send the transcoded video segments back to TC_{v_m} is calculated as

$$D_{k_{v_m}}^{(1,d)} = \frac{O_{k_{v_m}}^{(1)}}{b_{k_{v_m}}^{(1)} B_U e_{k_{v_m}}^{(1)}}, \tag{7}$$

where $e_{k_{v_m}}^{(1)}$ is the SE of the downlink from $U_{k_{v_m}}$ to TC_{v_m}, which can also be obtained with stochastic geometry methods [26] as:

$$\begin{aligned} e_{k_{v_m}}^{(1)} &= \frac{O_{k_{v_m}}^{(1)}}{b_{k_{v_m}}^{(1)} B_U \ln\left(1+\frac{P_U^T g_{m,v,k} l_{m,v,k}{}^{-\alpha}}{{\sigma_w}^2 + I_{k_{v_m}}^{(1)}}\right)} \\ &= \int_0^{\infty}\int_{l_{m,v,k}}^{\infty} \\ &\quad \times \exp\left(\begin{array}{l} -\varsigma\beta {l_{m,v,k}}^{\alpha}{\sigma_w}^2 \\ -2\pi\frac{\lambda_U}{K_{v_m}}\int_{\phi}^{\infty}\left(1-\frac{1}{1+\varsigma\beta\left(\frac{l_{m,v,k}}{z}\right)^{\alpha}}\right) z dz \end{array}\right) \\ &\quad \times f(\phi)\, d\phi d\beta, \end{aligned} \tag{8}$$

where $I_{k_{v_m}}^{(1)} = \sum_{x=1,x\neq m}^{M}\sum_{v=1}^{V_x}\sum_{k=1}^{K_{v_x}} P_U^T g_{x,v,k} l_{x,v,k}{}^{-\alpha}$ is the interference from the nodes in other small cells, and ϕ is the second dimension linear contact distribution [28] whose probability density function (PDF) is calculated as

$$\begin{aligned} & f(\phi) \\ &= 1 - \frac{1}{2}\exp\left(-4\pi\lambda_U\phi^2\right) + \frac{1}{2}\exp\left(-6\pi\lambda_U\phi^2\right) \\ &\quad -2\pi\lambda_U\int_{\phi}^{\infty} y \exp\left\{-\lambda_U\left[\begin{array}{l}\left(4\phi^2+2y^2\right)\left(\pi-\arccos\frac{\phi}{y}\right) \\ +6\phi\sqrt{y^2-\phi^2}\end{array}\right]\right\} dy. \end{aligned} \tag{9}$$

c) Energy consumption: When the transcoder TC_{v_m} chooses *mode 1*, i.e., offloading the transcoding task to a group of D2D users $\left\{U_1, U_2, \ldots, U_{K_{v_m}}\right\}$, the total energy consumption to transcode the whole video segment is [4]

$$E_{v_m}^{(1)} = \sum_{k=1}^{K_{v_m}}\left[\kappa {f_{k_{v_m}}}^3 D_{k_{v_m}}^{(1,t)} + P_U^T D_{k_{v_m}}^{(1,d)}\right]. \tag{10}$$

B. Incentive Mechanism

In the video streaming systems without any kind of intermediaries, the nodes may be reluctant to become transcoders since the transcoding jobs are computational intensive and time-consuming, or they may submit incorrect results or arbitrarily postpone the transcoding process. In other words, there is a lack of incentive to encourage the nodes to complete the transcoding jobs correctly and on time. To address this issue, we introduce an incentive mechanism for the proposed framework, which is based on the virtual currency circulated in the blockchain-based video streaming systems, i.e., token.

1) The Revenue of Transcoding Service: The revenue TC_{v_m} can receive by finishing the transcoding job is denoted by $R_{v_m}^{SC}$, whose details are pre-determinded in the smart contract and coded on the blockchain [25]. If TC_{v_m} successfully completed the transcoding job, it would receive a percentage ($\eta_{v_m}^b$) of block reward $R_{v_m}^B$ shared by the bonded nodes who stake on it. Other rewards and punishment are denoted by $R_{v_m}^O$ which comprises the transcoding reward, fee share, slashed reward, etc. In conclude, we can quantify the transcoding revenue $R_{v_m}^{SC}$ of TC_{v_m} as

$$R_{v_m}^{SC} = \eta_{v_m}^b \left(1 - \mathbb{P}_{orphan}\right) R_{v_m}^B + R_{v_m}^O, \tag{11}$$

where two key factors are outlined as follows:

• **Orphaning probability:** As is shown in (11), the expected block reward $R_{v_m}^B$ is discounted by the chances that the block is orphaned, $\mathbb{P}_{orphan}$, which largely depends on the block size S_B and block generation time T_G. Using the fact that block times follow a Poisson distribution, the orphaning probability can be approximated as [7], [9], [23]:

$$\mathbb{P}_{orphan} = 1 - e^{-\frac{t_P}{T_G}} = 1 - e^{-\frac{\xi S_B}{T_G}}, \tag{12}$$

where t_P is the block propagation time that is assumed to be linear with block size, i.e., $t_P = \xi S_B$ where ξ (*second/kB*) is the marginal time needed to reach consensus [29].

• **Block reward:** Following similar assumptions of block reward in [23] and [29], block reward includes a fixed reward $\dot{R}$ (fixed in a certain long period) and a variable reward (determined by the number of transactions included in one block). Accordingly, we can model the available block reward after the transactions are successfully included in the blockchain as

$$R_{v_m}^B = \dot{R} + \varepsilon \frac{S_B}{\chi_{v_m}}, \tag{13}$$

where ε is a given variable reward factor and χ_{v_m} denotes the average size of transactions w.r.t. task Φ_{v_m}.

Observing (12) and (13), we can find that the block size issue introduces a tradeoff between *orphaning probability* and *block reward*. Specifically, the larger the block is, it would take more time to propagate the mined block to the whole network, thus increases the chance of "orphaning". However, with larger block size, the block reward becomes higher due to more transactions included on one block at a time.

2) The Cost of Transcoding Service: In this paper, we assume the transcoders need to pay for the energy cost for the transcoding task. By introducing the unit price of energy ϖ_e (*token/J*), the payment for the energy cost of transcoding per video segment can be quantified as

$$Z_{v_m}^e = \varpi_e \left[\left(1 - \delta_{v_m}\right) E_{v_m}^{(0)} + \delta_{v_m} E_{v_m}^{(1)}\right]. \tag{14}$$

Taking the transcoding service revenue as well as cost into consideration, the profit TC_{v_m} can receive by transcoding the whole video file is given by (15) according to (11)-(14).

$$\begin{aligned}
&\Omega_{v_m} \\
&= R_{v_m}^{SC} - Z_{v_m}^e Q_{v_m} \\
&\overset{(a)}{=} \eta_{v_m}^b e^{-\frac{\xi}{T_G} S_B} \left(\dot{R} + \frac{\varepsilon}{\chi_{v_m}} S_B\right) - \varpi_e Q_{v_m} \\
&\times \left[\begin{array}{l} (1 - \delta_{v_m}) \left(\dfrac{\kappa (f_{SBS})^3}{\mu^{(0)}} \dfrac{1}{c_{v_m}} + \dfrac{\Lambda_{v_m}^{(0)}}{b_{v_m}^{(0)}} + P_S^I D_{v_m}^{(0,q)}\right) \\ \quad + \delta_{v_m} \displaystyle\sum_{k=1}^{K_{v_m}} \left(\dfrac{\Lambda_{k_{v_m}}^{(1)}}{b_{k_{v_m}}^{(1)}} + \kappa f_{k_{v_m}}{}^3 D_{k_{v_m}}^{(1,t)}\right) \end{array}\right],
\end{aligned} \tag{15}$$

where the part reward $R_{v_m}^O$ not relevant to the optimization variables is eliminated in (a), and these denotations are defined for ease of description: $\Lambda_{v_m}^{(0)} = \frac{P_S^T O_{v_m}^{(0)}}{B_S e_{v_m}^{(0)}}$ and $\Lambda_{k_{v_m}}^{(1)} = \frac{P_U^T O_{k_{v_m}}^{(1)}}{B_U e_{k_{v_m}}^{(1)}}$.

V. Problem Formulation and Solution

For blockchain-based ideo streaming systems, it's important to attract more users to become transcoders to perform transcoding for broadcasters, which can make the blockchain ecosystem more vibrant. To this end, we formulate the maximization of average transcoding service profit (i.e., average reward for the transcoders) as an optimization problem by jointly considering the issues of adaptive block size, scheduling of offloading, computational resource allocation and spectrum allocation $(S_B, \boldsymbol{\delta}, \boldsymbol{c}, \boldsymbol{b})$. To obtain the solution, we first relax the discrete variables and use the reformulation-linearization-technique (RLT) to reformulate the problem, and then adopt an ADMM-based algorithm to solve it in a distributed fashion.

A. Problem Formulation

In this paper, we aim to maximize the average reward for the transcoders by seeking the optimal offloading scheduling and resource allocation scheme for video transcoding with adaptive block size. Therefore, we formulate the optimization problem as $\mathcal{P}1$, where the optimization variables $(S_B, \boldsymbol{\delta}, \boldsymbol{c}, \boldsymbol{b})$ describe the adaptive block size S_B, the offloading decision indicator vector $\{\delta_{v_m}, \forall m, v\}$, the computational resource allocation profile $\{c_{v_m}, \forall m, v\}$, and the spectrum allocation profile $\left\{b_{v_m}^{(0)}, b_{k_{v_m}}^{(1)}, \forall m, v, k\right\}$.

$$\begin{aligned}
\mathcal{P}1: \quad & \max_{S_B, \boldsymbol{\delta}, \boldsymbol{c}, \boldsymbol{b}} \quad \frac{1}{M} \sum_{m=1}^{M} \sum_{v=1}^{V_m} \Omega_{v_m} \\
& s.t.\ C_1: \delta_{v_m} \in \{0, 1\}, \quad \forall m, v \\
& \qquad C_2: \sum_{m=1}^{M} \sum_{v=1}^{V_m} c_{v_m} \leq C, \\
& \qquad\quad c_{v_m} \in \{0, 1, \ldots, (1 - \delta_{v_m}) C\}, \quad \forall m, v \\
& \qquad C_3: \sum_{v=1}^{V_m} b_{v_m}^{(0)} B_S \leq B_S, 0 \leq b_{v_m}^{(0)} \leq 1 - \delta_{v_m}, \quad \forall m, v \\
& \qquad C_4: \sum_{v=1}^{V_m} \sum_{k=1}^{K_{v_m}} b_{k_{v_m}}^{(1)} B_U \leq B_U, 0 \leq b_{k_{v_m}}^{(1)} \leq \delta_{v_m}, \quad \forall m, v, k \\
& \qquad C_5: (1 - \delta_{v_m}) \left(\frac{1 / \mu^{(0)}}{c_{v_m}} + \frac{\Lambda_{v_m}^{(0)} / P_S^T}{b_{v_m}^{(0)}} + D_{v_m}^{(0,q)}\right)
\end{aligned}$$

$$+\ \delta_{v_m}\left(\frac{\Lambda_{k_{v_m}}^{(1)}\ P_U^T}{b_{k_{v_m}}^{(1)}}+D_{k_{v_m}}^{(1,t)}\right)\le\tau_{v_m},\quad\forall m,v,k$$

$$C_6: S_B\le\dot{S} \tag{16}$$

where constraints C_1 guarantee that the transcoding task offloading mode selection is valid. Constraints C_2 ensure the validity of computational resource allocation. Constraints C_3 and C_4 guarantee that the sum of spectrum allocated to all the downlinks between SBS/D2D nodes and the transcoders cannot exceed the total available spectrum bandwidth B_S/B_U. In order to meet the delay requirement, C_5 are put forward. Finally, constraint C_6 makes sure that the block size does not excess the block size limit $\dot{S}$.

Observing $\mathcal{P}1$, we can find this problem is extended from the Knapsack problem, which is faced with two major challenges: **1)** It's a mixed discrete and second order (in form of x/y in the objective and C_5, where x and y are optimization variables) optimization problem, which is notoriously difficult to solve [30]. **2)** $\mathcal{P}1$ has a quite large size. In order to solve $\mathcal{P}1$ in a centralized way, the MEC server needs to access the channel state information (CSI) of all the nodes, which is very challenging, especially for dense networks.

B. Problem Reformulation

To address **the first difficulty** (i.e., discrete and second order property), we can use discrete variable relaxation and RLT to reformulate the problem. First, we introduce two microscales, θ_c and θ_b, to avoid *divide-by-zero* error induced by $\boldsymbol{c}$ and $\boldsymbol{b}$, thus $\mathcal{P}1$ is converted into

$$\begin{aligned}
&\mathcal{P}_1{}':\\
&\max_{S_B,\boldsymbol{\delta},\boldsymbol{c},\boldsymbol{b}}\frac{1}{M}\sum_{m=1}^{M}\sum_{v=1}^{V_m}{\Omega_{v_m}}'\\
&s.t.\ C_1-C_4,C_6\\
&\quad C_5{}': (1-\delta_{v_m})\left(\frac{1/\mu^{(0)}}{c_{v_m}+\theta_c}+\frac{\Lambda_{v_m}^{(0)}/P_S^T}{b_{v_m}^{(0)}+\theta_b}+D_{v_m}^{(0,q)}\right)\\
&\qquad+\ \delta_{v_m}\left(\frac{\Lambda_{k_{v_m}}^{(1)}/P_U^T}{b_{k_{v_m}}^{(1)}+\theta_b}+D_{k_{v_m}}^{(1,t)}\right)\le\tau_{v_m},\quad\forall m,v,k
\end{aligned} \tag{17}$$

where Ω_{v_m} in the objective function turns into

$${\Omega_{v_m}}'=\eta_{v_m}^b e^{-\frac{\xi}{T_G}S_B}\left(\dot{R}+\frac{\varepsilon}{\chi_{v_m}}S_B\right)-\varpi_e Q_{v_m}\times\left[\begin{array}{l}(1-\delta_{v_m})\\ \left(\dfrac{\kappa(f_{SBS})^3}{\mu^{(0)}}\dfrac{1}{c_{v_m}+\theta_c}+\dfrac{\Lambda_{v_m}^{(0)}}{b_{v_m}^{(0)}+\theta_b}+P_S^I D_{v_m}^{(0,q)}\right)\\ +\delta_{v_m}\displaystyle\sum_{k=1}^{K_{v_m}}\left(\frac{\Lambda_{k_{v_m}}^{(1)}}{b_{k_{v_m}}^{(1)}+\theta_b}+\kappa {f_{k_{v_m}}}^3 D_{k_{v_m}}^{(1,t)}\right)\end{array}\right]. \tag{18}$$

Lemma 1: Problem ${\mathcal{P}_1}'$ *is a lower bound of problem* $\mathcal{P}_1$.

Proof: For better illustration, we define the feasible set of $\mathcal{P}_1$ and ${\mathcal{P}_1}'$ as $\mathcal{F}_1$ and ${\mathcal{F}_1}'$, respectively. Compared with $\mathcal{P}_1$, ${\mathcal{P}_1}'$ only shows the difference in the objective and ${C_5}'$. Due to the fact that $\frac{1}{x}>\frac{1}{x+\theta},\theta>0,\forall x$, the objective function of $\mathcal{P}_1$ is larger than that of ${\mathcal{P}_1}'$, i.e., $\Omega_{v_m}>{\Omega_{v_m}}',\forall m,v$. For the same reason, we can infer that C_5 is stricter than ${C_5}'$, then we have $\mathcal{F}_1\subset{\mathcal{F}_1}'$. In conclude, the optimal result of ${\mathcal{P}_1}'$ is smaller than that obtained from $\mathcal{P}_1$. In other words, ${\mathcal{P}_1}'$ provides a lower bound solution for $\mathcal{P}_1$. ■

Next, we define $\hat{c}_{v_m}:=\frac{1}{c_{v_m}+\theta_c}$, $\hat{b}_{v_m}^{(0)}:=\frac{1}{b_{v_m}^{(0)}+\theta_b}$ and $\hat{b}_{k_{v_m}}^{(1)}:=\frac{1}{b_{k_{v_m}}^{(1)}+\theta_b}$ as the auxiliary variables, and reformulate ${\Omega_{v_m}}'$, C_2, C_3, C_4 and ${C_5}'$ as follows:

$${\Omega_{v_m}}''\approx\eta_{v_m}^b e^{-\frac{\xi}{T_G}S_B}\left(\dot{R}+\frac{\varepsilon}{\chi_{v_m}}S_B\right)-\varpi_e Q_{v_m}\times\left[\begin{array}{l}(1-\delta_{v_m})\left(\dfrac{\kappa(f_{SBS})^3}{\mu^{(0)}}\hat{c}_{v_m}+\hat{b}_{v_m}^{(0)}\Lambda_{v_m}^{(0)}+P_S^I D_{v_m}^{(0,q)}\right)\\ +\delta_{v_m}\displaystyle\sum_{k=1}^{K_{v_m}}\left(\hat{b}_{k_{v_m}}^{(1)}\Lambda_{k_{v_m}}^{(1)}+\kappa {f_{k_{v_m}}}^3 D_{k_{v_m}}^{(1,t)}\right)\end{array}\right], \tag{19}$$

and

$$\begin{aligned}
C_2{}':&\ \sum_{m=1}^{M}\sum_{v=1}^{V_m}\frac{1}{\hat{c}_{v_m}}\le C+N\theta_c,\\
&\ \frac{1}{(1-\delta_{v_m})C+\theta_c}\le\hat{c}_{v_m}\le\frac{1}{\theta_c},\quad\forall m,v\\
C_3{}':&\ \sum_{v=1}^{V_m}\frac{1}{\hat{b}_{v_m}^{(0)}}\le 1+V_m\theta_b,\\
&\ \frac{1}{(1-\delta_{v_m})+\theta_b}\le\hat{b}_{v_m}^{(0)}\le\frac{1}{\theta_b},\quad\forall m,v\\
C_4{}':&\ \sum_{v=1}^{V_m}\sum_{k=1}^{K_{v_m}}\frac{1}{\hat{b}_{k_{v_m}}^{(1)}}\le 1+N\theta_b,\\
&\ \frac{1}{(1-\delta_{v_m})+\theta_b}\le\hat{b}_{k_{v_m}}^{(1)}\le\frac{1}{\theta_b},\quad\forall m,v,k\\
C_5{}'':&\ (1-\delta_{v_m})\left(\frac{\hat{c}_{v_m}}{\mu^{(0)}}+\frac{\hat{b}_{v_m}^{(0)}\Lambda_{v_m}^{(0)}}{P_S^T}+D_{v_m}^{(0,q)}\right)\\
&\ +\delta_{v_m}\left(\frac{\hat{b}_{k_{v_m}}^{(1)}\Lambda_{k_{v_m}}^{(1)}}{P_U^T}+D_{k_{v_m}}^{(1,t)}\right)\le\tau_{v_m},\quad\forall m,v,k.
\end{aligned} \tag{20}$$

Then we transform ${\mathcal{P}_1}'$ into $\mathcal{P}_2$ by the following two steps (Seen in **Appendix A**).

1) ***Discrete Variable Relaxation:*** We first relax the discrete variables $\boldsymbol{\delta}$ and $\boldsymbol{c}$ into continuous variables as $0\le\delta_{v_m}\le 1$ and $0\le c_{v_m}\le C$, respectively.

2) ***Reformulation Linearization Technique (RLT) based Transformation:*** For the second order terms in the form of $x\cdot y$, we use the RLT [31] to linearize the problem by defining $\boldsymbol{X}=\{X_{v_m}\},X_{v_m}=\delta_{v_m}\hat{c}_{v_m}$, $\boldsymbol{Y}=\left\{Y_{v_m}^{(0)},Y_{k_{v_m}}^{(1)}\right\}$,

$Y_{v_m}^{(0)}=\delta_{v_m}\hat{b}_{v_m}^{(0)}, Y_{k_{v_m}}^{(1)}=\delta_{v_m}\hat{b}_{k_{v_m}}^{(1)}$.

$\mathcal{P}2$:

$$\max_{\substack{S_B,\boldsymbol{\delta},\hat{\boldsymbol{c}},\hat{\boldsymbol{b}}\\ \boldsymbol{X},\boldsymbol{Y}}} \frac{1}{M}\sum_{m=1}^{M}\eta_{v_m}^{b}e^{-\frac{\xi}{T_G}S_B}\left(\dot{R}+\frac{\varepsilon}{\chi_{v_m}}S_B\right)-\varpi_e Q_{v_m} \times\begin{bmatrix}(\hat{c}_{v_m}-X_{v_m})\dfrac{\kappa\left(f_{SBS}\right)^3}{\mu^{(0)}}+\left(\hat{b}_{v_m}^{(0)}-Y_{v_m}^{(0)}\right)\Lambda_{v_m}^{(0)}\\ +(1-\delta_{v_m})P_S^I D_{v_m}^{(0,q)}\\ +\sum_{k=1}^{K_{v_m}}\left(Y_{k_{v_m}}^{(1)}\Lambda_{k_{v_m}}^{(1)}+\delta_{v_m}\kappa {f_{k_{v_m}}}^3 D_{k_{v_m}}^{(1,t)}\right)\end{bmatrix}$$

$$s.t.\ C_6\ C_1{}': 0\le\delta_{v_m}\le 1,\quad \forall m,v$$

$$C_2{}'': \sum_{m=1}^{M}\sum_{v=1}^{V_m}\frac{1}{\hat{c}_{v_m}}\le C+N\theta_c, \hat{c}_{v_m}\le\frac{1}{\theta_c},\quad (\hat{c}_{v_m}-X_{v_m})C+\hat{c}_{v_m}\theta_c\ge 1,\quad \forall m,v$$

$$C_3{}'': \sum_{v=1}^{V_m}\frac{1}{\hat{b}_{v_m}^{(0)}}\le 1+V_m\theta_b,\quad \hat{b}_{v_m}^{(0)}\le\frac{1}{\theta_b}, \hat{b}_{v_m}^{(0)}-Y_{v_m}^{(0)}+\hat{b}_{v_m}^{(0)}\theta_b\ge 1,\quad \forall m,v$$

$$C_4{}'': \sum_{v=1}^{V_m}\sum_{k=1}^{K_{v_m}}\frac{1}{\hat{b}_{k_{v_m}}^{(1)}}\le 1+N\theta_b, \hat{b}_{k_{v_m}}^{(1)}\le\frac{1}{\theta_b},\quad Y_{k_{v_m}}^{(1)}+\hat{b}_{k_{v_m}}^{(1)}\theta_b\ge 1,\forall m,v,k$$

$$C_5{}''': (\hat{c}_{v_m}-X_{v_m})\frac{1}{\mu^{(0)}}+\left(\hat{b}_{v_m}^{(0)}-Y_{v_m}^{(0)}\right)\frac{\Lambda_{v_m}^{(0)}}{P_S^T} +(1-\delta_{v_m})D_{v_m}^{(0,q)}+Y_{k_{v_m}}^{(1)}\frac{\Lambda_{k_{v_m}}^{(1)}}{P_U^T} +\delta_{v_m}D_{k_{v_m}}^{(1,t)}\le\tau_{v_m},\quad \forall m,v,k$$

$$C_7: X_{v_m}\in\Xi_{v_m}^{c},\quad Y_{v_m}^{(0)}\in\Xi_{v_m}^{(b,0)}, Y_{k_{v_m}}^{(1)}\in\Xi_{k_{v_m}}^{(b,1)},\quad \forall m,v,k \tag{21}$$

As is mentioned above, the computational complexity is another main challenge, which would cause significant signaling overhead, especially in dense networks. In order to address this problem, we adopt the ADMM method to decouple $\mathcal{P}_2$ and further solve it in a distributed way.

C. Problem Decomposition

To address **the second difficulty** (i.e., large-scale issue), we attempt to solve $\mathcal{P}2$ in a distributed manner with the ADMM algorithm. ADMM is a simple but powerful algorithm that is well suited to solve distributed optimization problems due to its advantages in enabling quick convergence to a modest accuracy of the optimal solution [18]. Observing $\mathcal{P}_2$, we can find that S_B, $\hat{\boldsymbol{c}}$ and its related varibale $\boldsymbol{X}$ are global variables that make the optimization problem inseparable. Intuitively, we can introduce the local copies of $S_B, \hat{\boldsymbol{c}}, \hat{\boldsymbol{X}}$ for each small cell, and then try to make these copies agree. For small cell m, we denote $\tilde{\boldsymbol{S}}_{\boldsymbol{B}} := \left\{\tilde{S}_B(m)\right\}, \tilde{\boldsymbol{c}} := \left\{\tilde{c}_{v_i}(m)\right\}, \tilde{\boldsymbol{X}} := \left\{\tilde{X}_{v_i}(m)\right\}$, $v=1,\ldots,V_i, m, i=1,\ldots,M$ as the local copy of $S_B, \hat{\boldsymbol{c}}$, and $\hat{\boldsymbol{X}}$, so we have

$$\tilde{S}_B(m)=S_B,\quad \tilde{c}_{v_i}(m)=\hat{c}_{v_i},\ \tilde{X}_{v_i}(m)=X_{v_i},\ \forall m,i,v. \tag{22}$$

Then the equivalent global consensus version of $\mathcal{P}_2$ is

$\mathcal{P}_3$:

$$\max_{\tilde{\boldsymbol{S}}_{\boldsymbol{B}},S_B;\boldsymbol{F}^l,\boldsymbol{F}^g}\frac{1}{M}\sum_{m=1}^{M}\sum_{v=1}^{V_m}\eta_{v_m}^{b}e^{-\frac{\xi\tilde{S}_B(m)}{T_G}}\left[\dot{R}+\frac{\varepsilon\tilde{S}_B(m)}{\chi_{v_m}}\right]-\varpi_e Q_{v_m} \left\{\begin{array}{l}\left[\tilde{c}_{v_i}(m)-\tilde{X}_{v_i}(m)\right]\dfrac{\kappa\left(f_{SBS}\right)^3}{\mu^{(0)}}\\ +\left(\hat{b}_{v_m}^{(0)}-Y_{v_m}^{(0)}\right)\Lambda_{v_m}^{(0)}+(1-\delta_{v_m})P_S^I D_{v_m}^{(0,q)}\\ +\sum_{k=1}^{K_{v_m}}\left(Y_{k_{v_m}}^{(1)}\Lambda_{k_{v_m}}^{(1)}+\delta_{v_m}\kappa {f_{k_{v_m}}}^3 D_{k_{v_m}}^{(1,t)}\right)\end{array}\right\}$$

$$s.t.\ C_1{}', C_3{}'', C_4{}''$$

$$C_2{}''': \sum_{m=1}^{M}\sum_{v=1}^{V_m}\frac{1}{\tilde{c}_{v_i}(m)}\le C+N\theta_c,\quad \tilde{c}_{v_i}(m)\le\frac{1}{\theta_c}, \left[\tilde{c}_{v_i}(m)-\tilde{X}_{v_i}(m)\right]C +\tilde{c}_{v_i}(m)\theta_c\ge 1,\quad \forall m,i,v$$

$$C_5{}'''': \left[\tilde{c}_{v_i}(m)-\tilde{X}_{v_i}(m)\right]\frac{1}{\mu^{(0)}} +\left(\hat{b}_{v_m}^{(0)}-Y_{v_m}^{(0)}\right)\frac{\Lambda_{v_m}^{(0)}}{P_S^T} +(1-\delta_{v_m})D_{v_m}^{(0,q)}+Y_{k_{v_m}}^{(1)}\frac{\Lambda_{k_{v_m}}^{(1)}}{P_U^T} +\delta_{v_m}D_{k_{v_m}}^{(1,t)}\le\tau_{v_m},\quad \forall m,i,v,k$$

$$C_6{}': \tilde{S}_B(m)\le\dot{S},\quad \forall m$$

$$C_7{}': \tilde{X}_{v_i}(m)\in\Xi_{v_m}^{c},\quad Y_{v_m}^{(0)}\in\Xi_{v_m}^{(b,0)}, Y_{k_{v_m}}^{(1)}\in\Xi_{k_{v_m}}^{(b,1)},\quad \forall m,i,v,k$$

$$C_8: \tilde{S}_B(m)=S_B,\quad \tilde{c}_{v_i}(m)=\hat{c}_{v_i}, \tilde{X}_{v_i}(m)=X_{v_i},\quad \forall m,i,v \tag{23}$$

where constraints C_8 ensure the consistency between the local variables and the corresponding global ones, and $\boldsymbol{F}^l=\left\{\mathcal{F}_m^l\right\}=\left\{\delta,\tilde{c},b,\tilde{X},Y\right\}_m$ and $\boldsymbol{F}^g=\left\{\mathcal{F}_m^g\right\}=\left\{\hat{c},X\right\}_m$ are denoted for convenience.

Note that $\tilde{\boldsymbol{S}}_{\boldsymbol{B}}, S_B$ (block size related) and $\boldsymbol{F}^l, \boldsymbol{F}^g$ (offloading scheduling and resource allocation related) are decoupled, so we can divide the local utility function into two parts:

$$U_m^{(1)}=\begin{cases}-\sum_{v=1}^{V_m}\eta_{v_m}^{b}e^{-\frac{\xi\tilde{S}_B(m)}{T_G}}\left[\dot{R}+\dfrac{\varepsilon\tilde{S}_B(m)}{\chi_{v_m}}\right], & if\ C_6{}'\ holds\\ \infty, & otherwise\end{cases} \tag{24}$$

and

$$U_m^{(2)} = \begin{cases} \sum_{v=1}^{V_m} \varpi_e Q_{v_m} \\ \left\{ \begin{array}{l} \left[\tilde{c}_{v_i}(m) - \tilde{X}_{v_i}(m)\right] \frac{\kappa (f_{SBS})^3}{\mu^{(0)}} \\ + \left(\hat{b}_{v_m}^{(0)} - Y_{v_m}^{(0)}\right) \Lambda_{v_m}^{(0)} + (1-\delta_{v_m}) P_S^I D_{v_m}^{(0,q)} \\ + \sum_{k=1}^{K_{v_m}} \left(Y_{k_{v_m}}^{(1)} \Lambda_{k_{v_m}}^{(1)} + \delta_{v_m} \kappa {f_{k_{v_m}}}^3 D_{k_{v_m}}^{(1,t)} \right) \end{array} \right\}, \\ \quad if\ C_1', C_2''', C_3'', C_4'', C_5'''', C_7'\ holds \\ \infty, \quad otherwise \end{cases} \tag{25}$$

With (24) and (25), we can rewrite $\mathcal{P}_3$ as

$$\begin{aligned} \mathcal{P}_3': \quad & \min_{\tilde{\boldsymbol{S}}_{\boldsymbol{B}}, S_B; \boldsymbol{F}^l, \boldsymbol{F}^g} \sum_{m=1}^{M} U_m^{(1)} + U_m^{(2)} \\ & C_8: \tilde{S}_B(m) = S_B, \quad \tilde{c}_{v_i}(m) = \hat{c}_{v_i}, \\ & \tilde{X}_{v_i}(m) = X_{v_i}, \quad \forall m, i, v \end{aligned} \tag{26}$$

Thanks to the introduction of local copies, $\mathcal{P}_3'$ can be decoupled into M sub-problems that can be independently solved in each small cell. Therefore, the ADMM can be used to solve $\mathcal{P}_3'$ in a distributed fashion, thus the computational complexity can be significantly reduced.

D. Problem Solutions With ADMM

According to [32] and [33], $\mathcal{P}_3'$ is so called *global consensus problems* that needs a network-wide consensus. First, we derive the augmented Lagrangian for $\mathcal{P}_3'$ as

$$\begin{aligned} & L_\rho\left(\left\{\tilde{\boldsymbol{S}}_B, \boldsymbol{F}^l\right\}, \{S_B, \boldsymbol{F}^g\}, \left\{\boldsymbol{w}^S, \boldsymbol{w}^c, \boldsymbol{w}^X\right\}\right) \\ & = \sum_{m=1}^{M} \left[U_m^{(1)} + U_m^{(2)}\right] + \sum_{m=1}^{M} w^S(m) \left[\tilde{S}_B(m) - S_B\right] \\ & \quad + \sum_{m=1}^{M} \sum_{i=1}^{M} \sum_{v=1}^{V_i} w_{v_i}^c(m) \left[\tilde{c}_{v_i}(m) - \hat{c}_{v_i}\right] \\ & \quad + \sum_{m=1}^{M} \sum_{i=1}^{M} \sum_{v=1}^{V_i} w_{v_i}^X(m) \left[\tilde{X}_{v_i}(m) - X_{v_i}\right] \\ & \quad + \frac{\rho}{2} \sum_{m=1}^{M} \left[\tilde{S}_B(m) - S_B\right]^2 \\ & \quad + \frac{\rho}{2} \sum_{m=1}^{M} \sum_{i=1}^{M} \sum_{v=1}^{V_i} \left[\tilde{c}_{v_i}(m) - \hat{c}_{v_i}\right]^2 \\ & \quad + \frac{\rho}{2} \sum_{m=1}^{M} \sum_{i=1}^{M} \sum_{v=1}^{V_i} \left[\tilde{X}_{v_i}(m) - X_{v_i}\right]^2, \end{aligned} \tag{27}$$

where $\boldsymbol{w}^S = \left\{w^S(m)\right\}, \boldsymbol{w}^c = \left\{w_{v_i}^c(m)\right\}, \boldsymbol{w}^X = \left\{w_{v_i}^X(m)\right\}, v = 1, \ldots, V_i, m, i = 1, \ldots, M$ are the Lagrange multipliers w.r.t. C_8, and $\rho \in \mathbb{R}_{++}$ is the penalty coefficient to promote consensuses.

Then based on ADMM, the the global consensus of block size, offloading scheduling and resource allocation for video transcoding can be achieved by the following updates. The whole optimization process includes the iteration of local variables, global variables as well as Lagrange multipliers.

1) Adaptive Block Size:

a) The adaptive block size for SBS m: To obtain the optimal block size, each SBS needs to solve the following local problem:

$$\begin{aligned} \mathcal{SP}_{1_L}: \ & \min_{\tilde{\boldsymbol{S}}_{\boldsymbol{B}}} U_m^{(1)} \\ & + \left\{ \left[w^S(m)\right]^{[t]} \tilde{S}_B(m) + \frac{\rho}{2} \left[\tilde{S}_B(m) - {S_B}^{[t]}\right]^2 \right\}. \end{aligned} \tag{28}$$

Observing $\mathcal{SP}_{1_L}$, we can find it's a non-convex problem due to the non-convexity of $U_m^{(1)}$, but a sub-optimal solution can be obtained using the gradient descent method [30].

b) Reach the global consensus on the block size: The updating of global variables can be considered as taking the average of all local copies w.r.t. the global variables to achieve a global consensus.

$$S_B = \frac{1}{M\rho} \sum_{m=1}^{M} \left[w^S(m)\right]^{[t]} + \frac{1}{M} \sum_{m=1}^{M} \left[\tilde{S}_B(m)\right]^{[t+1]}. \tag{29}$$

c) Multiplier updating at SBS m:

$$\left\{\boldsymbol{w}^S(m)\right\}^{[t+1]} = \left[\boldsymbol{w}^S(m)\right]^{[t]} + \rho\left(\left[\tilde{\boldsymbol{S}}_{\boldsymbol{B}}(m)\right]^{[t+1]} - S_B^{[t+1]}\right). \tag{30}$$

2) Offloading Scheduling and Resource Allocation:

a) Offloading scheduling and resource allocation decision updating at SBS m: Similarly, each SBS needs to solve the following local problem to obtain the offloading scheduling and resource allocation solution.

$$\begin{aligned} & \mathcal{SP}_{2_L}: \\ & \min_{\boldsymbol{F}^l} \ U_m^{(2)} \\ & + \sum_{i=1}^{M} \sum_{v=1}^{V_i} \left\{ \left[w_{v_i}^c(m)\right]^{[t]} \tilde{c}_{v_i}(m) + \frac{\rho}{2} \left[\tilde{c}_{v_i}(m) - \hat{c}_{v_i}^{[t]}\right]^2 \right\} \\ & + \sum_{i=1}^{M} \sum_{v=1}^{V_i} \left\{ \left[w_{v_i}^X(m)\right]^{[t]} \tilde{X}_{v_i}(m) + \frac{\rho}{2} \left[\tilde{X}_{v_i}(m) - {X_{v_i}}^{[t]}\right]^2 \right\}. \end{aligned} \tag{31}$$

It's obvious that $\mathcal{SP}_{2_L}$ is a convex problem, so it can be solved efficiently in polynomial time using standard software such as CVX, SeDuMi, etc.

b) Reach global consensus on computational resource allocation:

$$\hat{c}_{v_i} = \frac{1}{M\rho} \sum_{m=1}^{M} \left[w_{v_i}^c(m)\right]^{[t]} + \frac{1}{M} \sum_{m=1}^{M} \left[\tilde{c}_{v_i}(m)\right]^{[t+1]}, \tag{32}$$

$$X_{v_i} = \frac{1}{M\rho} \sum_{m=1}^{M} \left[w_{v_i}^X(m)\right]^{[t]} + \frac{1}{M} \sum_{m=1}^{M} \left[\tilde{X}_{v_i}(m)\right]^{[t+1]}. \tag{33}$$

c) Multiplier updating at SBS m:

$$\{\boldsymbol{w}^c(m)\}^{[t+1]} = [\boldsymbol{w}^c(m)]^{[t]} + \rho\Big([\tilde{\boldsymbol{c}}(m)]^{[t+1]} - \hat{\boldsymbol{c}}^{[t+1]}\Big), \quad (34)$$

$$\big\{\boldsymbol{w}^X(m)\big\}^{[t+1]} = \big[\boldsymbol{w}^X(m)\big]^{[t]} + \rho\bigg(\Big[\tilde{\boldsymbol{X}}(m)\Big]^{[t+1]} - \boldsymbol{X}^{[t+1]}\bigg). \quad (35)$$

Algorithm 1 ADMM-Based Offloading Scheduling and Resource Allocation Algorithm With Adaptive Block Size

1: **Initialization**
a) Initialize the feasible global solution $(S_B^{(0)}, \{\boldsymbol{c}\}^{(0)}, \{\boldsymbol{X}\}^{(0)})$, and microscales (θ_c, θ_b);
b) Each SBS m determines its initial Lagrange multiplier vector $\boldsymbol{w}^S$, $\boldsymbol{w}^c$ and $\boldsymbol{w}^X$;
c) Each transcoder TC_{v_m} finds a group of nearby D2D nodes within the range of D_{ts} in small cell m.
d) Each transcoder TC_{v_m} calculates the transcoding reward as well as output size, delay and energy consumption w.r.t. *mode 0* and *mode 1* according to Section IV.
$t = 0$;
2: **Repeat** To reach the global consensus on
a) *Adaptive Block Size S_B:*
Each SBS m updates the local optimal block size $\{S_B\}^{[t+1]}$ by solving $\mathcal{SP}_{1_L}$ as (28), updates the global variables $\left\{\tilde{\boldsymbol{S}}_{\boldsymbol{B}}\right\}^{[t+1]}$ as (29), and updates the the Lagrange multipliers $\left\{\boldsymbol{w}^S\right\}^{[t+1]}$ as (30).
b) *Offloading Scheduling & Resource Allocation $\{\boldsymbol{\delta}, \boldsymbol{c}, \boldsymbol{b}\}$:*
Each SBS m updates its local offloading scheduling and resource allocation decisions $\left\{\boldsymbol{F}^l\right\}^{[t+1]}$ by solving $\mathcal{SP}_{2_L}$ as (31), updates the global variables $\left\{\boldsymbol{F}^g\right\}^{[t+1]}$ as (32)-(33), and updates the the Lagrange multipliers $\left\{\boldsymbol{w}^c, \boldsymbol{w}^X\right\}^{[t+1]}$ as (34)-(35).
$t = t + 1$;
Until stopping criteria (36)-(41) are satisfied.
3: Discrete variables recovery for $\boldsymbol{\delta}$ and $\boldsymbol{c}$.
4: Obtain the solution $\{S_B, \boldsymbol{\delta}, \boldsymbol{c}, \boldsymbol{b}\}^*$.

3) Stopping Criterion: In the implementation process, we employ the stopping criteria as proposed in [33]. Specifically, the residuals for both the primal and dual feasibility condition of small cell m in iteration $[t+1]$ should be small enough such that

$$\left\|\tilde{\boldsymbol{S}}_{\boldsymbol{B}}(m)^{[t+1]} - {S_B}^{[t+1]}\right\|_2 \le \iota_{pri}, \quad \forall m, \quad (36)$$

$$\left\|\tilde{\boldsymbol{c}}(m)^{[t+1]} - \hat{\boldsymbol{c}}^{[t+1]}\right\|_2 \le \iota_{pri}, \quad \forall m, \quad (37)$$

$$\left\|\tilde{\boldsymbol{X}}(m)^{[t+1]} - \boldsymbol{X}^{[t+1]}\right\|_2 \le \iota_{pri}, \quad \forall m, \quad (38)$$

and

$$\left\|{S_B}^{[t+1]} - {S_B}^{[t]}\right\|_2 \le \iota_{dual}, \quad (39)$$

$$\left\|\hat{\boldsymbol{c}}^{[t+1]} - \hat{\boldsymbol{c}}^{[t]}\right\|_2 \le \iota_{dual}, \quad (40)$$

$$\left\|\boldsymbol{X}^{[t+1]} - \boldsymbol{X}^{[t]}\right\|_2 \le \iota_{dual}, \quad (41)$$

where $\iota_{pri} > 0$ and $\iota_{dual} > 0$ are suitably defined tolerances for the primal and dual feasibility conditions, respectively.

4) Discrete Variables Recovery: Recall that in Section V-B, we relax the discrete variables $\boldsymbol{\delta}$ and $\boldsymbol{c}$ into continuous ones. Therefore, we need to recover the discrete variables after the proposed algorithm is converged. For simplicity, we recover the discrete variables by the rounding-off method. Based on the previous discussions, the optimal offloading scheduling and resource allocation for video transcoding with adaptive block size can be obtained while achieving the maximum average transcoding reward. Details of the proposed ADMM based algorithm are summarized in **Algorithm** 1.

5) Optimality and Complexity of the Algorithm: Please note that the objective function and all the variables of $\mathcal{P}3$ are bounded. Therefore, when the optimal solution reaches, the inequality $\sum_{m=1}^{M} U_m < \infty$ holds. According to [33], the objective function of $\mathcal{SP}_{2_L}$ is convex, closed and proper due to its convexity. Thus, there exists saddle points for the Lagrangian (27) and the proposed RLT-ADMM based algorithm meets the objective convergence, residual convergence as well as dual variable convergence when $t \to \infty$. Therefore, the optimality of $\mathcal{SP}_{2_L}$ can be guaranteed. For the non-convex local problem $\mathcal{SP}_{1_L}$, a sub-optimal solution is obtained as long as the convergence condition $\left\|\tilde{\boldsymbol{S}}_{\boldsymbol{B}}(m)^{[t+1]} - {S_B}^{[t+1]}\right\|_2 \le \iota_{pri}, \forall m$ is satisfied [30].

Using the ADMM method, the whole problem can be broken down into solving local optimization problems at each SBS. In this way, the computation complexity can be significantly reduced from $\mathcal{O}\Big(M\dot{N}\left(\dot{K}+2\right)+1\Big)^x$ to $\mathcal{O}\Big(\dot{N}\left(\dot{K}+2\right)+1\Big)^x Q$, where $x > 1$ means a polynomial time algorithm, Q denotes the number of iterations required for algorithm convergence, and $\dot{N}$ and $\dot{K}$ are the average number of nodes in each MBS and the D2D links between each two nodes, respectively.

VI. Simulation Results and Discussions

In the simulation part, we set the network coverage radius of the MBS as 1000m while the SBS density and user density are set at $10^{-4}/m^2$ and $10^{-3}/m^2$, respectively. The SBSs' transmit power and idle power are $P_S^T = 0.1W$ and $P_S^I = 0.01W$ while the transmit power of users is $P_U^T = 0.02W$. Besides, the blockchain related parameters are set as follows: block size limit $\dot{S} = 0.5 \sim 2$ *MB*, block generation time $T_G = 10 \sim 20$ *s*, average transaction size $\chi_{v_m} = 200 \sim 500$ *B*, the percentage of block reward that transcoders can get $\eta_{v_m}^b = 1\%$, variable reward factor $\varepsilon = 0.1$, the marginal time needed to reach consensus $\xi = 0.01$ *s/kB*, fixed reward $\dot{R} = 5$ *token*. Other simulation parameters are listed in Table II.

To testify the effectiveness of our proposed algorithm (i.e., ADMM (joint)), we compare the simulation results with that of the centralized scheme, the proposed algorithm without using blockchain [20], and several baseline schemes including *Mode 0*-only scheme (all the transcoders offload the transcoding task to the serving SBSs), *Mode 1*-only scheme (all the transcoders offload the transcoding task to a group of nearby D2D nodes),

TABLE II
SIMULATION PARAMETERS

Parameter	Value
The length of each video segment L_{v_m}	2-10s
The bitrate of different required versions r_{v_m}	200-5000kb/s
The pathloss exponent α	4
The available bandwidth for cellular/D2D networks B_S/B_U	20MHz/10MHZ
The power of noise σ_w^2	-174dBm/Hz
The mean value of Rayleigh fading ς	1
Computation worload/intensity X_{v_m}	18000 CPU cycles/bit
Computation energy efficiency coefficient of the processor's chip in the SBSs/users κ	10^{-26}
The number of CPU cores at SBS m c_{v_m}	10–50
CPU-cycle frequency of the users $f_{k_{v_m}}^U$	10–100 GHz CPU cycles/s
The unit price of energy ϖ_e	0.1 token/J

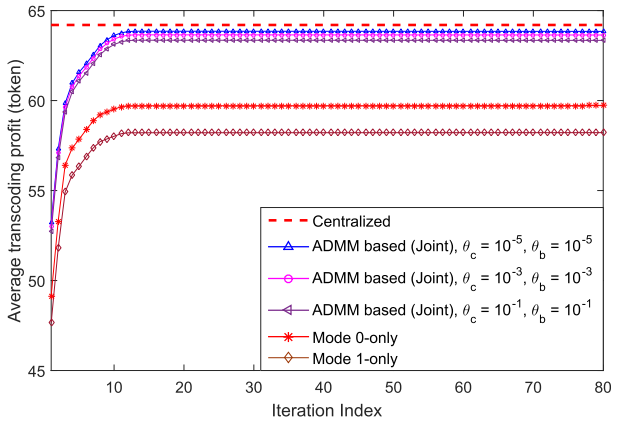
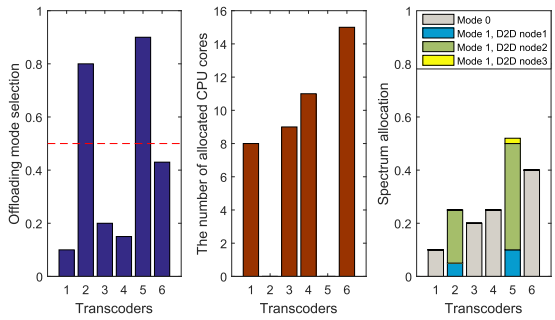


Fig. 2. Convergence performance of the proposed ADMM based algorithm with different microscales θ_c and θ_b.

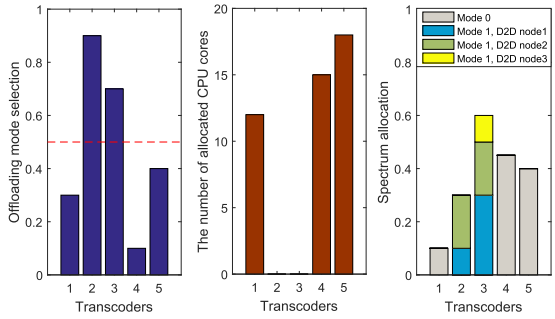
fixed block size, uniform computing, and uniform spectrum allocation solutions.

A. The Convergence of the Proposed ADMM Based Algorithm

First, the convergence performance of the proposed ADMM based algorithm with different values of microscales θ_c and θ_b is presented in Fig. 2. Several interesting observations can be obtained: **1)** In the first 10 iterations, the average transcoding profit of each transcoder in the curves increase quickly and gradually reach a stable state within the first 20 iterations, which verifies the good convergence of our proposed ADMM based algorithm. **2)** The gap between the proposed ADMM based algorithm and the centralized one is rather small. Besides, the average profits obtained from the single modes (*mode 0*-only or *mode 1*-only scheme) are much lower. This is because for those two single modes, the computational and spectrum resources are not fully exploited. **3)** The average transcoding profit grows with the decrease of microscales θ_c and θ_b. Note that the growth of the average transcoding profit is very small (<5%) when θ_c and θ_b decrease from 10^{-3} to 10^{-5}, which demonstrates that our proposed ADMM based algorithm can provide a very tight lower bound solution for the original optimization problem.



(a) *Small cell 1*



(b) *Small cell 2*

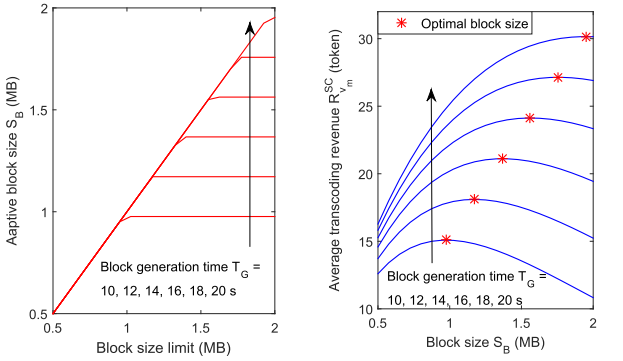
Fig. 3. An illustration of offloading mode selection and resource allocation.

B. An Illustration of Resource Allocation Profile

Next, we give an illustration of the offloading mode selection (before discrete variable recovery) and resource allocation scheme w.r.t. *small cell 1* and *small cell 2* obtained from our proposed algorithm. It can be seen that there 6 and 5 transcoders (required versions) in *small cell 1* and *2*, respectively. The left sub-figures in Fig. 3(a) and Fig. 3(b) describe the offloading decision while the middle sub-figures show the computational resource allocation. Furthermore, spectrum allocation in *mode 0* and *mode 1* are illustrated in the right sub-figures. We can observe that the number of D2D nodes varies from transcoder to transcoder. For example, there are 2 D2D nodes for transcoder 2 while there are 3 for transcoder 5 in *small cell 1*, which depends on the accessible D2D communication distance. Obviously, the optimal policy is not uniform among different transcoders or small cells, striving for the maximum average transcoding profit.

C. Effects of Different Parameters

In this part, the effects of several parameters including block size limit & block generation time, delay tolerance and available computational resources are investigated in Fig. 4, Fig. 5-6 and Fig. 7-8 respectively. For blockchain systems, block size limit and block generation time are two popular topics. Theoretically, the approach of increasing block size or reducing block generation time can bring higher block reward as well as higher orphaning probability for blocks, which would involve a trade-off. Besides, higher delay tolerance and more available computational resources can introduce higher transcoding profit. All these theoretical analysis will be demonstrated in the relevant simulation results in Fig. 4-Fig. 8.



(a) *An illustration of the proposed adaptive block size scheme* (b) *Average transcoding revenue vs. block size S_B (when $\dot{S}$= 2MB)*

Fig. 4. The effect of block size limit $\dot{S}$ and block generation time T_G.

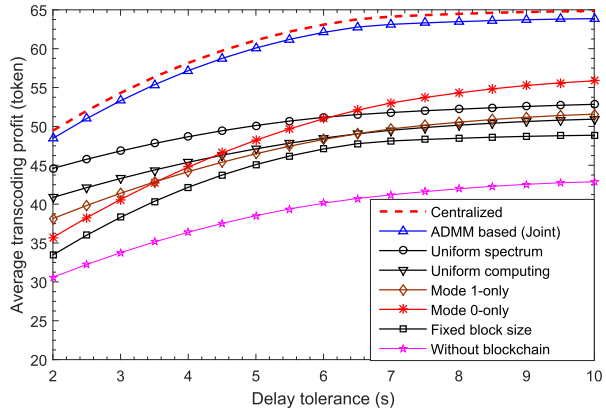
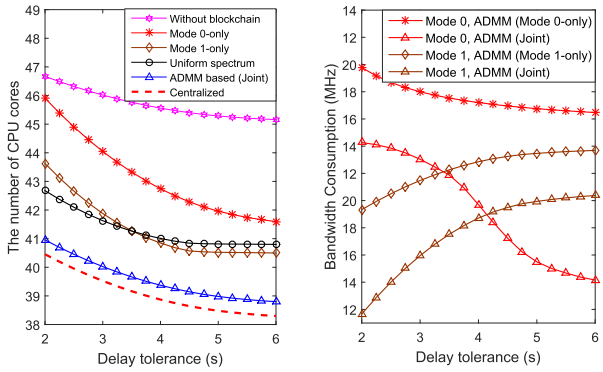


Fig. 5. Average transcoding profit vs. delay tolerance.



(a) *Computational resource consumption* (b) *Bandwidth consumption*

Fig. 6. Resource consumption vs. delay tolerance.

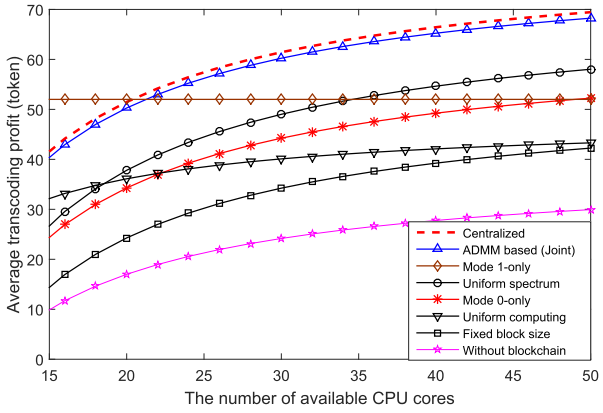


Fig. 7. Average transcoding profit vs. available CPU cores.

1) Effects of Block Size Limit & Block Generation Time: Fig. 4 illustrates our proposed adaptive block size scheme and discusses the effects of block size limit $\dot{S}$ as well as block generation time T_G. Observing Fig. 4(a), we can find that the block size S_B obtained from our proposed scheme first increases and then reaches stable, and the stable point varies with different $\dot{S}$ and T_G, which shows the obtained block size is adaptive to the variation of block size limit and block generation time. The reasons behind can be found in Fig. 4(b). For one thing, larger block size brings higher block reward as well as induces higher orphaning probability, thus the average transcoding revenue first grows and then declines, where the highest revenue is achieved at the optimal block size. Besides, it can be seen that the increase of block generation time contributes to higher average transcoding revenue due to the less chance of block orphaning. However, in practice, large block generation interval would also decrease the transactional rate, which introduces another tradeoff left for future works.

2) Effects of Delay Tolerance: In Fig. 5, the average transcoding profit obtained from our proposed ADMM based algorithm with varying delay tolerance is compared with the centralized algorithm and the other baseline schemes. We can find that the average transcoding profit of our proposed algorithm is very close to that achieved by the centralized algorithm. Besides, all the schemes obtain higher average profit with the increasing delay tolerance. The reason lies in: as the delay constraint becomes more relaxed, more transcoders incline to select *mode 1* which would cause longer delay but can save the energy cost for transcoding, thus higher average transcoding profit can be achieved. Additionally, it's no wonder to find that the solution with uniform computing and uniform spectrum get lower average profits because uniform resource allocation schemes usually cannot reach the optimal policy. Compared with the proposed adaptive block size scheme, the transcoding profit achieved by the fixed block size scheme is much lower because it doesn't get the highest block reward. Moreover, it can be seen that the solution without blockchain [17] receives the lowest transcoding profit because the MEC nodes needs to download the video chunks from the cloud and send the transcoded version back through backhaul links, which introduces longer delay compared with the blockchain-based solutions with peer-to-peer (P2P) transmission.

For the same reason, the computational resource and bandwidth consumption in *mode 0* decrease with the increasing delay tolerance while *mode 1* shows an increase in bandwidth consumption in Fig. 6. Additionally, Fig. 6 also illustrates that the proposed ADMM based algorithm has an advantage over one single mode (*mode 0*-only scheme or *mode 1*-only scheme) and the one without using blockchain in the terms of saving resources.

3) Effects of Available Computational Resources: The effects of available computational resource on average transcoding profit and average delay are investigated in Fig. 7 and Fig. 8, respectively. We can see that all the schemes can

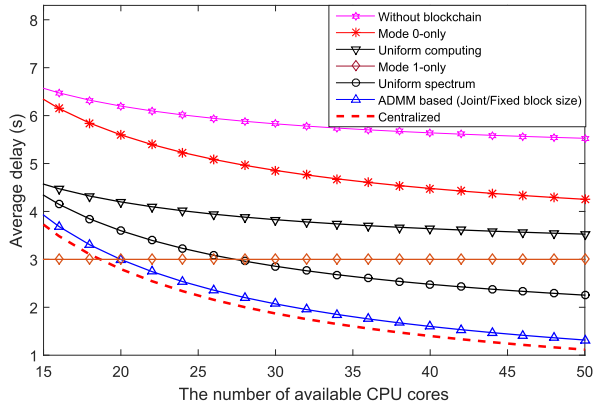


Fig. 8. Average delay vs. available CPU cores.

achieve higher transcoding profit and takes shorter time to complete the transcoding job with more available CPU cores, except the *mode 1*-only solution that remains unchanged. It is obvious that the gap between our proposed algorithm and the centralized algorithm is not wide. Besides, the reaction of the solution with uniform computing is the least distinct with the increase of available CPU cores among all the schemes. Moreover, it can be observed that the solution without using blockchain shows the poorest performance from the aspects of transcoding profit and average delay due to more time cost on downloading the video chunks from the cloud and sending the transcoded versions back through the backhaul links.

VII. Conclusion and Future Work

In this paper, we proposed a novel blockchain-based framework for video streaming systems with MEC. We formulated the joint design of offloading scheduling, resource allocation and adaptive block size scheme as an optimization problem to maximize the average transcoding profit for the transcoders. First, we presented the offloading framework with two offloading modes and introduced an incentive mechanism to facilitate the cooperations among content creators, transcoders and consumers. Then the original optimization problem was reformulated using the RLT. Further, a low-complexity ADMM-based algorithm was put forward to solve the problem in a distributed and efficient way. Finally, simulation results verified the effectiveness of the adaptive block size scheme and manifested that the proposed ADMM-based algorithm can provide a tight lower bound for the original problem and outperforms other baseline schemes. In our future works, we will study the transcoders selection scheme considering a wide range of factors, such as the held stake, reputation value, communication and computing capability. Another interesting direction is to regard the smart contract as the ADMM coordinator, which is an effective way to enable distributed optimization among non-trusting nodes.

Appendix
The Transformation of $\mathcal{P}_1{}'$ into $\mathcal{P}_2$

After the discrete variable relaxation, i.e., $0 \le \delta_{v_m} \le 1$ and $0 \le c_{v_m} \le C$, we can derive that δ_{v_m} and $\hat{c}_{v_m}$ are bonded by $0 \le \delta_{v_m} \le 1$ and $\frac{1}{C+\theta_c} \le \hat{c}_{v_m} \le \frac{1}{\theta_c}$, respectively. Therefore, we can obtain the *RLT bound-factor product constraints* for X_{v_m} as

$$\Xi^{c}_{v_m} = \begin{cases} X_{v_m} - \dfrac{1}{C+\theta_c}\delta_{v_m} \ge 0, \\ \dfrac{1}{\theta_c} - \hat{c}_{v_m} - \dfrac{1}{\theta_c}\delta_{v_m} + X_{v_m} \ge 0, \\ \dfrac{1}{\theta_c}\delta_{v_m} - X_{v_m} \ge 0, \\ \hat{c}_{v_m} - \dfrac{1}{C+\theta_c} - X_{v_m} + \dfrac{1}{C+\theta_c}\delta_{v_m} \ge 0. \end{cases} \tag{42}$$

Similarly, δ_{v_m} and $\hat{b}^{(0)}_{v_m}$ are restricted by $0 \le \delta_{v_m} \le 1$ and $\frac{1}{1+\theta_b} \le \hat{b}^{(0)}_{v_m} \le \frac{1}{\theta_b}$, thus we can obtain the *RLT bound-factor product constraints* for $Y^{(0)}_{v_m}$ as

$$\Xi^{(b,0)}_{v_m} = \begin{cases} Y^{(0)}_{v_m} - \dfrac{1}{1+\theta_b}\delta_{v_m} \ge 0, \\ \dfrac{1}{\theta_b} - \hat{b}^{(0)}_{v_m} - \dfrac{1}{\theta_b}\delta_{v_m} + Y^{(0)}_{v_m} \ge 0, \\ \dfrac{1}{\theta_b}\delta_{v_m} - Y^{(0)}_{v_m} \ge 0, \\ \hat{b}^{(0)}_{v_m} - \dfrac{1}{1+\theta_b} - Y^{(0)}_{v_m} + \dfrac{1}{1+\theta_b}\delta_{v_m} \ge 0. \end{cases} \tag{43}$$

The constraints for $Y^{(1)}_{k_{v_m}}$ is denoted by $\Xi^{(b,1)}_{k_{v_m}}, \forall m, v, k$, which has the similar form with $\Xi^{(b,0)}_{v_m}$. After substituting X_{v_m}, $Y^{(0)}_{v_m}$ and $Y^{(1)}_{k_{v_m}}$ into $\Omega_{v_m}{}''$, $C_3{}'$, $C_4{}'$ and $C_5{}''$, the original optimization problem is reformulated as $\mathcal{P}_2$.

References

[1] "Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021," Cisco Mobile VNI, San Jose, CA, USA, Tech. Rep. 1454457600805266, Feb. 2017.
[2] D. Magazzeni, P. McBurney, and W. Nash, "Validation and verification of smart contracts: A research agenda," *Computer*, vol. 50, no. 9, pp. 50–57, 2017.
[3] 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.
[4] 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.
[5] G. Gao, Y. Wen, and C. Westphal, "Resource provisioning and profit maximization for transcoding in information centric networking," in *Proc. IEEE INFOCOM WKSHPS*, San Francisco, CA, USA, Apr. 2016, pp. 97–102.
[6] S. Andrew. (2015). *An Examination of Single-Transaction Blocks and Their Effect on Network Throughput and Block Size*. [Online]. Available: https://www.bitcoinunlimited.info/resources/1txn.pdf
[7] P. R. Rizun, "A transaction fee market exists without a block size limit," Block Size Limit Debate Work. Paper, Tech. Rep., Aug. 2015.
[8] BGroup. (Sep. 2015). *Block Size Increase V1.1*. [Online]. Available: http://bitfury.com/content/5-white-papers-research/block-size-1.1.1.pdf
[9] U. Klarman, S. Basu, A. Kuzmanovic, and E. G. Sirer, "bloXroute: A scalable trustless blockchain distribution network whitepaper v1.0," Bloxroute Labs, Evanston, IL, USA, White Paper, Mar. 2018.
[10] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Joint computation offloading and content caching for wireless blockchain networks," in *Proc. IEEE INFOCOM WKSHPS*, Honolulu, HI, USA, Apr. 2018, pp. 1–6.
[11] T. T. Nguyen and L. B. Le, "Computation offloading leveraging computing resources from edge cloud and mobile peers," in *Proc. IEEE Int. Conf. Commun.*, May 2017, pp. 1–6.
[12] T. T. Nguyen and L. B. Le, "Computation offloading in MIMO based mobile edge computing systems under perfect and imperfect CSI estimation," in *Proc. IEEE ICC*, Kansas City, MO, USA, May 2018, pp. 1–6.
[13] Y. Zhou, F. R. Yu, J. Chen, and Y. Kuo, "Video transcoding, caching, and multicast for heterogeneous networks over wireless network virtualization," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 141–144, Jan. 2018.

[14] X. Xu, J. Liu, and X. Tao, "Mobile edge computing enhanced adaptive bitrate video delivery with joint cache and radio resource allocation," *IEEE Access*, vol. 5, pp. 16406–16415, Aug. 2017.
[15] Y. Li, P. A. Frangoudis, Y. Hadjadj-Aoul, and P. Bertin, "A mobile edge computing-based architecture for improved adaptive HTTP video delivery," in *Proc. IEEE Conf. Standards Commun. Net. (CSCN)*, Berlin, Germany, Oct./Nov. 2016, pp. 1–6
[16] X. Zhou, Z. Chang, C. Sun, and X. Zhang, "Demo abstract: Context-aware video streaming with q-learning for MEC-enabled cellular networks," in *Proc. IEEE INFOCOM WKSHPS*, Honolulu, HI, USA, Apr. 2018, pp. 1–2.
[17] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia IoT systems," *IEEE Trans. Multimedia*, vol. 20, no. 5, pp. 1126–1139, May 2018.
[18] C. Liang, Y. He, F. R. Yu, and N. Zhao, "Enhancing video rate adaptation with mobile edge computing and caching in software-defined mobile networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 10, pp. 7013–7026, Aug. 2018.
[19] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.
[20] N. C. Luong, Z. Xiong, P. Wang, and D. Niyato, "Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach," in *Proc. IEEE ICC*. Kansas City, MO, USA, May 2018, pp. 1–6.
[21] Y. Jiao, P. Wang, D. Niyato, and Z. Xiong, "Social welfare maximization auction in edge computing resource allocation for mobile blockchain," in *Proc. IEEE ICC*, Kansas City, MO, USA, May 2018, pp. 1–6.
[22] Z. Xiong, S. Feng, D. Niyato, P. Wang, and Z. Han, "Optimal pricing-based edge computing resource management in mobile blockchain," in *Proc. IEEE ICC*, Kansas City, MO, USA, May 2018, pp. 1–6.
[23] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han. (2017). "Cloud/fog computing resource management and pricing for blockchain networks." [Online]. Available: https://arxiv.org/abs/1710.01567
[24] D. Petkanic and E. Tang. (Dec. 2017). *Livepeer Whitepaper: Protocol and Economic Incentives for a Decentralized Live Video Streaming Network*. [Online]. Available: https://github.com/livepeer/wiki/blob/master/WHITEPAPER.md
[25] Flixxo. (Oct. 2017). *Flixxo: Community Based Video Distribution White Paper V0.5*. [Online]. Available: https://www.flixxo.com
[26] J. G. Andrews, F. Baccelli, and R. K. Ganti, "A tractable approach to coverage and rate in cellular networks," *IEEE Trans. Commun.*, vol. 59, no. 11, pp. 3122–3134, Nov. 2011.
[27] P. Zhao, H. Tian, C. Qin, and G. Nie, "Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing," *IEEE Access*, vol. 5, pp. 11255–11268, Jun. 2017.
[28] L. Muche, "Contact and chord length distribution functions of the poisson-voronoi tessellation in high dimensions," *Adv. Appl. Probab.*, vol. 42, no. 1, pp. 48–68, Oct. 2010.
[29] N. Houy. (Mar. 2014). *The Bitcoin Mining Game*. [Online]. Available: https://halshs.archives-ouvertes.fr/file/index/docid/958224/filename/1412.pdf
[30] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[31] E. Dalkiran and H. D. Sherali, "RLT-POS: Reformulation-Linearization Technique-based optimization software for solving polynomial programming problems," *Math. Program. Comput.*, vol. 8, no. 3, pp. 337–375, Feb. 2016.
[32] H. Tabrizi, B. Peleato, G. Farhadi, J. M. Cioffi, and G. Aldabbagh, "Spatial reuse in dense wireless areas: A cross-layer optimization approach via ADMM," *IEEE Trans. Wireless Commun.*, vol. 14, no. 12, pp. 7083–7095, Dec. 2015.
[33] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

Mengting Liu received the B.S. degree from the Minzu University of China, Beijing, China, in 2013. She is currently pursuing the Ph.D. degree with the Beijing University of Posts and Telecommunications, Beijing. From 2017 to 2018, she was a Visiting Ph.D. Student with the University of British Columbia, Vancouver, Canada. Her current research interests include Blockchain, deep reinforcement learning, and mobile edge computing.

F. Richard Yu (S'00–M'04–SM'08–F'18) received the Ph.D. degree in electrical engineering from The University of British Columbia in 2003. From 2002 to 2006, he was with Ericsson, Lund, Sweden, and with a start-up in California, USA. He joined Carleton University in 2007, where he is currently a Professor. His research interests include wireless cyber-physical systems, connected/autonomous vehicles, security, distributed ledger technology, and deep learning. He received the Leadership Opportunity Fund Award from the Canada Foundation of Innovation in 2009, the Excellent Contribution Award at the IEEE/IFIP TrustCom 2010, the Ontario Early Researcher Award (formerly Premiers Research Excellence Award) in 2011, the Carleton Research Achievement Award in 2012, the IEEE Outstanding Leadership Award in 2013, the IEEE Outstanding Service Award in 2016, and the Best Paper Awards at the IEEE ICNC 2018, VTC 2017 Spring, ICC 2014, Globecom 2012, the IEEE/IFIP TrustCom 2009, and the International Conference on Networking 2005.

Dr. Yu is a fellow of the Institution of Engineering and Technology. He has served as the technical program committee co-chair for numerous conferences. He is a Distinguished Lecturer, the Vice President (Membership), and an Elected Member of the Board of Governors (BoG) of the IEEE Vehicular Technology Society. He serves on the Editorial Boards of several journals, as the Co-Editor-in-Chief for *Ad Hoc & Sensor Wireless Networks*, and as the Lead Series Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, and IEEE COMMUNICATIONS SURVEYS & TUTORIALS. He is a registered Professional Engineer in the province of Ontario, Canada.

Yinglei Teng (M'12) received the B.S. degree from Shandong University, China, in 2005, and the Ph.D. degree in electrical engineering from the Beijing University of Posts and Telecommunications (BUPT) in 2011. She is currently an Associate Professor with the School of Electronic Engineering, BUPT. Her current research interests include UDNs and massive MIMO, IoTs, and Blockchains.

Victor C. M. Leung (S'75–M'89–SM'97–F'03) is currently a Professor of electrical and computer engineering and holds the TELUS Mobility Research Chair, The University of British Columbia (UBC). His research is in the broad areas of wireless networks and mobile systems. He has co-authored over 1100 technical papers in archival journals and refereed conference proceedings, several of which has received the best paper awards. He is a fellow of the Royal Society of Canada, the Canadian Academy of Engineering, and the Engineering Institute of Canada. He received the IEEE Vancouver Section Centennial Award, the 2011 UBC Killam Research Prize, the 2017 Canadian Award for Telecommunications Research, and the 2018 IEEE ComSoc TGCC Distinguished Technical Achievement Recognition Award. He has co-authored papers that received the 2017 IEEE ComSoc Fred W. Ellersick Prize, the 2017 IEEE Systems Journal Best Paper Award, and the 2018 IEEE ComSoc CSIM Best Journal Paper Award. He is serving on the Editorial Boards of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE ACCESS, and several other journals.

Mei Song received the B.E. and M.E. degrees from Tianjin University, China. She is currently a Professor with the Beijing University of Posts and Telecommunications. Her current research interests include integrated design technology, VLSI&CAD system, resource allocation, and mobility management in heterogeneous and cognitive networks, cooperative communication, and other advanced technology in future communication networks.