Joint Task Assignment, Transmission, and Computing Resource Allocation in Multilayer Mobile Edge Computing Systems

Pengfei Wang, *Student Member, IEEE*, Chao Yao, *Member, IEEE*, Zijie Zheng, *Student Member, IEEE*, Guangyu Sun, *Member, IEEE*, and Lingyang Song, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a multilayer data flow processing system, i.e., EdgeFlow, to integrally utilize the computing capacity throughout the whole network, i.e., the cloud center (CC) on the top layer, the mobile edge computing (MEC) servers on the middle layer, and the edge devices (EDs) on the bottom layer. To realize the efficient data processing in EdgeFlow, we optimally assign the tasks to multiple layers, and allocate the wireless transmission resources between the MEC servers and EDs as well as the wired transmission resources between the CC and MEC servers. We prove that the system is naturally classified into two states, the nonblocking state and the blocking state, according to various data generation speed at the EDs. The system latency is minimized for the nonblocking state even though the problem is nonconvex. As for the blocking state, the recovery time is minimized through solving a min-max problem. Based on the analytical results, the EdgeFlow system is implemented on the universal software radio peripheral and the Intel next units of computing. A typical Internet of Things application, photo recording and face recognition, is used for the simulation and the experiment, and indicates that the EdgeFlow can achieve a low latency and recovery time than the previous distributed frameworks, e.g., the Cloudlet and the Markov decision process.**

***Index Terms*—Internet of Things (IoT), mobile edge computing (MEC), resource allocation, task assignment.**

I. Introduction

WITH the increasing number of electronic and intelligent devices connected in modern lives, the Internet of Things (IoT) has attracted broad attentions in both the industrial and the academic fields [1], [2]. Generally, the IoT is defined as the network of interconnected devices embedded with electronics and sensors [3], [4]. The potentialities offered by the IoT enable the development and the automation of a huge number of applications in the fields of transportation, healthcare [5], smart environment [6], [7], etc. Some of them require a very low latency for response, while some may generate large quantities of data intermittently, resulting in heavy loads to the IoT network [8]. As predicted, there will be 50 billion IoT devices connected to the Internet by 2020 [9]. Countless connected IoT devices will generate the massive data continuously, resulting in two main challenges.

1) Large amount of raw data and computing tasks need to be processed, while the computing capacity of each IoT device is limited.
2) A huge volume of data needs to be transmitted through the network with a low latency to fulfill the requirements of the real-time tasks [10], [11], while both the wireless and the wired transmission resources are inadequate in the networks.

A. Basic Concept in Cloud and Edge

In order to process a large amount of raw data, the cloud computing and mobile edge computing (MEC) are introduced, utilizing the computing capacity of the cloud center (CC) and MEC servers, respectively.

Cloud computing has been proposed to take use of the strong computing capacity in the data centers to process the data delivered from the IoT devices [12], however, it is not scalable or efficient for the IoT services due to the following reasons. The cloud computing usually needs a long link to deliver large quantities of raw data from the IoT devices to the CC, which results in huge transmission pressure over the limited frequency bandwidth. This may not fulfill some IoT applications. For example, in the scenario of an autonomous driving vehicle, one Gigabyte data needs to be processed in time for error-free decisions for driving safety [13], however, the latency of the cloud computing is too large to transmit the raw data.

To reduce the transmission time between the IoT devices and the cloud, the MEC has been proposed [14], defined as providing Internet service environment and enabling the computation to be performed at the edge of the mobile network [15], where the term "edge" refers to any computing and network resources between data sources and CCs [16]. The edge has the computing capacity, offering an opportunity to offload part of the computing tasks from the CC to the edge, which can evidently help to reduce the transmission time. Cloudlet is an early implementation of the edge computing platform [16]–[18], where the computing tasks are sent to the nearest deployed servers rather than the remote CC so that the transmission delay is significantly reduced. Moreover,

Manuscript received July 11, 2018; revised October 2, 2018; accepted October 11, 2018. Date of publication October 16, 2018; date of current version May 8, 2019. This work was supported by the National Nature Science Foundation of China under Grant 61625101. *(Corresponding author: Lingyang Song.)*

The authors are with the School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China (e-mail: wangpengfei13@pku.edu.cn; chao.yao@pku.edu.cn; zijie.zheng@pku.edu.cn; gsun@pku.edu.cn; lingyang.song@pku.edu.cn).

Digital Object Identifier 10.1109/JIOT.2018.2876198

the real-time applications, such as the assisted driving can obtain a shorter response time by applying MEC [19], [20]. Compared to cloud computing, the MEC enables proximity services with low latency and location awareness, bringing about small payload of transmission [21], [22].

B. Coupled Task Assignment, Transmission, and Computing Resource Allocation

Only relying on the cloud computing results in the additional long transmission time and huge transmission pressure, while the processing capacity of only MEC is still limited for those IoT applications with huge data volume [23]–[25], [30]. Hence, it is essential to combine the strong computing capacity of the cloud computing and the close-distance advantage of the MEC. In the unified system, we need to make full use of the transmission[1] and computing[2] resources in both cloud and edge. In order to optimally utilize these resources, the task generated at the ED can be split and assigned to the CC, MEC server or local, and thus, an overall task assignment strategy is called for.

Most existing works only discuss the task assignment [26], [27], some together with transmission resource allocation [28]–[30] or computing resource allocation [31], [32]. However, the three aspects are not considered jointly or the relationship between them is ignored. An MDP approach is proposed in [27], which schedules the computation tasks based on the queueing and the transmitting or processing execution state. The transmission resource allocation for multiuser mobile edge computational offloading constrained by the computation latency is studied in [28] and [29], and Guo *et al.* [30] discussed it in the ultradense IoT networks. However, the allocation of the computing resource is not taken into account. Ko *et al.* [31] analyzed the transmission latency and computation latency separatively with different mobile device density, taken the task assignment and computing rate control into consideration. The energy harvesting is studied in the computation latency constrained task assignment problem in [32], in order to minimizing the power consumption of the MEC server.

The task assignment, transmission, and computing resource allocation are coupled with close relationship, for the task assignment decision is directly influenced by the transmission and computing resource allocation, constrained by both the transmission latency and the computation latency. Unfortunately, the existing works in cloud computing or edge computing do not jointly consider the three closely correlated aspects, where only one or two aspects are taken into account.

C. Our Contribution

In this paper, we propose a multilayer data flow processing system, named by EdgeFlow. We combine the strong computing capacity of the CC and the close-distance advantage of the MEC, to integrally utilize the computing capacity throughout the whole network, i.e., the CC on the top layer, the MEC servers (or the APs) on the middle layer, and the IoT EDs on the bottom layer. There are two main challenges to design such a unified scheme or a system.

1) The task assignment on the different layers and different nodes are highly correlated with the computing and the transmission resources allocation.
2) The volume of data varies with time and IoT applications, and thus the task assignment strategy and resource allocation scheme need to be adjusted according to the data generation speed. Furthermore, due to the limited computing capacity and transmission resources of the network, the system may be blocked when large amount of data pours into the network, causing a complicated case.

In order to realize the data processing task, in EdgeFlow, we optimally assign the tasks on multiple layers and allocate both the wireless transmission resources between the EDs and the MEC servers as well as the wired transmission resources between the MEC servers and the CC. We prove that the system will be naturally classified into two states, the nonblocking state and the blocking state, according to various data generation speed on the EDs. A latency minimization algorithm is proposed for the nonblocking state to minimize the total latency even though the problem is nonconvex. As for the blocking state, the recovery time is minimized through solving a min-max problem. Based on the analytical results, the EdgeFlow system is realized and implemented on the universal software radio peripheral (USRP) and the Intel next units of computing (Intel-NUCs), where the demo code in the second version can be found in [36]. A typical IoT application, photo recording and face recognition are used to for the simulation and the experiment.

The main contributions and results of this paper are summarized as below.

1) We propose the EdgeFlow system for IoT applications, which is a multilayer data flow processing system combining the cloud and edge, making full use of the computing and transmission resources of the whole network.
2) We jointly consider the transmission, computing resource allocation, and task assignment in our system, and derive the clear relationship between the task assignment strategy, the transmission, and computing resource allocation.
3) We point out the network can be classified into two states, the nonblocking state and the blocking state, according to various data generation speed on the EDs. We clearly derive the quantitive boundary between the two states.
4) As far as we know, we are the first to quantitively define and describe the different objectives that should be considered for different states. For the nonblocking state, we can minimize the total latency even though the problem is nonconvex. However, for the blocking state, we minimize the recovery time since the total latency is meaningless. Algorithms for both states are designed.

[1]The transmission resources include the wired transmission resource of the CC and the wireless transmission resource of the MEC servers, referring to time, frequency or power resources.

[2]The computing resources refer to the computing rate of the CC, MEC server or ED.

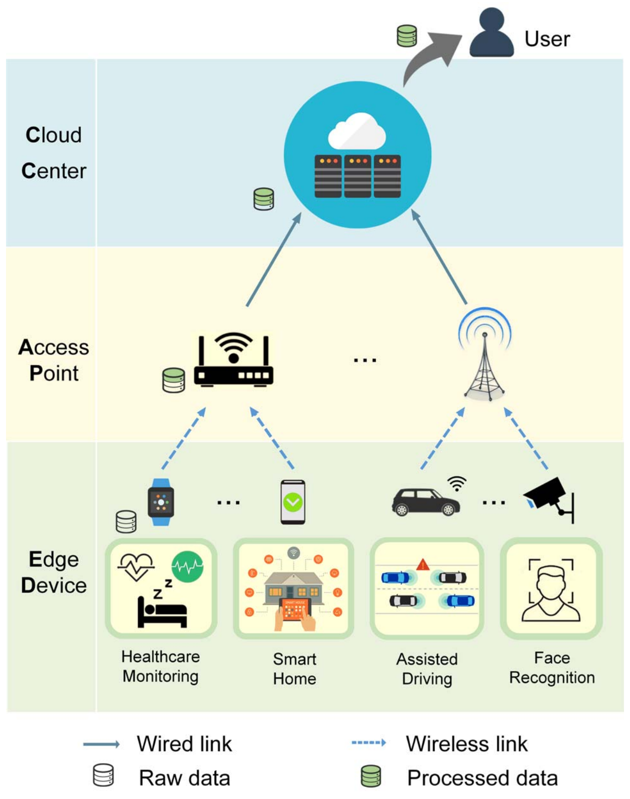


Fig. 1. Three-layer EdgeFlow architecture.

5) In the experiment, the EdgeFlow can achieve a low total latency and recovery time than the previous distributed frameworks and the systems, such as the Cloudlet and the MDP.

The rest of this paper is organized as follows. In Section II, the model of the three-layer EdgeFlow system is described. In Section III, we present the judgement of the system states, the blocking state, and the nonblocking state. We formulate the latency minimization problem in the nonblocking state and the recovery time minimization in the blocking state. Algorithms for the two optimization problems in the nonblocking and blocking state are designed in Section IV. Finally, we present the implementation of the EdgeFlow system in Section VI. The simulation results as well as the experiment results are given in Section VII. The conclusions are drawn in Section VIII.

II. System Model

As shown in Fig. 1, we consider a general communication network, with one CC, N APs, and M EDs. The EDs connect with the AP via wireless communication, while the APs connect with the CC through the wired links. We assume that each ED can connect at most one AP, and each AP can connect at most one CC. Each node in the network possesses a certain amount of computing capacity. The raw data is generated at the EDs in the IoT applications and the results of the data processing must be aggregated at the CC.[3] The downlink of the network is not considered in our scenario, since it is not necessary for the IoT scenario. In IoT applications, we focus on the collection and processing of the data generated at the IoT devices, i.e., the uplink. The processing of the raw data can be performed at any layer from the EDs to the CC, and the percentage of the data to process at each layer is adjustable. Moreover, once the data is processed at the edge of the network, i.e., at the ED or AP, only the results with the smaller size rather than the raw data need to be forwarded to the CC. In the rest of this part, we provide the details of the CC, the APs and the EDs.

[3]That is to say, we only consider an uplink IoT network. More complicated networks are left for future work.

A. Edge Device

The EDs on the bottom layer are responsible for generating the raw data, which usually include IoT devices, e.g., the mobile phones, cameras, and other sensors. The ED processes part of the raw data, and delivers the rest raw data together with the processing results to the AP via wireless link. Let $\lambda_{\text{ED}}^{j,i}$ denotes the data generation speed of ED i connected with AP j $(1 \le j \le N, 1 \le i \le M)$, and ρ denotes the compression ratio after the data processing. The ED i connected with AP j can process part of the raw data, and $s_{\text{ED}}^{j,i}$ represents its task division percentage, which satisfies that

$$0 \le s_{\text{ED}}^{j,i} \le 1. \tag{1}$$

The computing capacity and the wireless transmitting capacity of ED i connected with AP j per unit time is denoted by $\theta_{\text{ED}}^{j,i}$ and $\phi_{\text{ED}}^{j,i}$, respectively. The computing data volume is limited by its computing capacity

$$\lambda_{\text{ED}}^{j,i} s_{\text{ED}}^{j,i} \le \theta_{\text{ED}}^{j,i}. \tag{2}$$

The transmitting data volume is limited by the wireless transmitting capacity of ED i, which is closely related with the wireless transmission resources allocated by AP j

$$\rho \lambda_{\text{ED}}^{j,i} s_{\text{ED}}^{j,i} + \lambda_{\text{ED}}^{j,i}\left(1 - s_{\text{ED}}^{j,i}\right) \le \phi_{\text{ED}}^{j,i} \tag{3}$$

where the first part is the processing results, and the second part represents the remaining raw data to transmit. The total transmitting data volume of all EDs connected with AP j is *linearly* constrained by the wireless transmission resources of AP j, denoted by ϕ_{AP}^{j}, which can be expressed by

$$\sum_{i=1}^{M} \phi_{\text{ED}}^{j,i} \le \phi_{\text{AP}}^{j}. \tag{4}$$

Remark 1: The constraints in (4) can describe some wireless resources that influence the wireless data rate in a linear manner, for example, the spectrum and time resources. The power and the antenna resources cannot be modeled and discussed in the similar way, which are left in the future works.

B. Access Point

Being the middle layer of the three-layer EdgeFlow model, APs, including base stations, WiFi APs, and so on, receive the raw data and the processing results from the connected EDs. After processing part of the receiving raw data, the AP forwards the rest raw data together with the processing results of both the AP and EDs to the CC.

Considering that the raw data generated at ED i is transmitted to AP j, the equivalent raw data arriving speed at AP j can be calculated by

$$\lambda_{\mathrm{AP}}^{j,i} = \phi_{\mathrm{ED}}^{j,i} \cdot \frac{1 - s_{\mathrm{ED}}^{j,i}}{1 - s_{\mathrm{ED}}^{j,i} + \rho s_{\mathrm{ED}}^{j,i}} \tag{5}$$

where $\phi_{\mathrm{ED}}^{j,i}$ is the total data arriving speed to AP j from ED i, and only part of it is the raw data arriving speed. The raw data volume transmitted to AP j from ED i is $\lambda_{\mathrm{ED}}^{j,i}(1-s_{\mathrm{ED}}^{j,i})$, and the processed data volume is $\lambda_{\mathrm{ED}}^{j,i}\rho s_{\mathrm{ED}}^{j,i}$, therefore, the ratio of raw data in the total arriving data is $(1 - s_{\mathrm{ED}}^{j,i})/(1 - s_{\mathrm{ED}}^{j,i} + \rho s_{\mathrm{ED}}^{j,i})$. Accordingly, the processed data transmitted from ED i to AP j can be expressed by

$$\beta_{\mathrm{AP}}^{j,i} = \phi_{\mathrm{ED}}^{j,i} \cdot \frac{\rho s_{\mathrm{ED}}^{j,i}}{1 - s_{\mathrm{ED}}^{j,i} + \rho s_{\mathrm{ED}}^{j,i}}. \tag{6}$$

The AP can also process part of the received raw data, and $s_{\mathrm{AP}}^{j,i}$ denotes the task division percentage of AP j for the data from ED i $(1 \leq j \leq N, 1 \leq i \leq M)$, which satisfies that

$$0 \leq s_{\mathrm{AP}}^{j,i} \leq 1. \tag{7}$$

The computing capacity and the wired transmitting capacity of AP j for the task from ED i per unit time is denoted by $\theta_{\mathrm{AP}}^{j,i}$ and $\phi_{\mathrm{AP}}^{j,i}$, respectively. The computing data volume of AP j for the task from ED i is limited by the computing capacity allocated to the ED

$$\lambda_{\mathrm{AP}}^{j,i} s_{\mathrm{AP}}^{j,i} \leq \theta_{\mathrm{AP}}^{j,i}. \tag{8}$$

Moreover, the total computing capacity allocated to different EDs is no larger than the computing capacity of AP j, expressed by

$$\sum_{i=1}^{M} \theta_{\mathrm{AP}}^{j,i} \leq \theta_{\mathrm{AP}}^{j} \tag{9}$$

where θ_{AP}^{j} denotes the computing capacity of AP j.

The transmitting data volume of AP j is limited by its wired transmitting capacity, which is closely related with the wired transmission resources allocated by the CC

$$\rho \lambda_{\mathrm{AP}}^{j,i} s_{\mathrm{AP}}^{j,i} + \lambda_{\mathrm{AP}}^{j,i}\left(1 - s_{\mathrm{AP}}^{j,i}\right) + \beta_{\mathrm{AP}}^{j,i} \leq \phi_{\mathrm{AP}}^{j,i}. \tag{10}$$

The data to be transmitted to the CC includes three parts. The first part $\rho\lambda_{\mathrm{AP}}^{j,i} s_{\mathrm{AP}}^{(j)}$ is the processed data of the AP j, the second part $\lambda_{\mathrm{AP}}^{j,i}(1 - s_{\mathrm{AP}}^{j,i})$ is the remaining raw data to transmit, and the third part $\beta_{\mathrm{AP}}^{j,i}$ is the processed data delivered from the EDs. All the three parts need to be transmitted to the CC, which is limited by the allocated wired transmitting capacity $\phi_{\mathrm{AP}}^{j,i}$ of AP j. Moreover, the total transmitting data volume of all APs is limited by the wired transmission resources of the CC, denoted by ϕ_{CC}, which can be expressed by

$$\sum_{j=1}^{N} \sum_{i=1}^{M} \phi_{\mathrm{AP}}^{j,i} \leq \phi_{\mathrm{CC}}. \tag{11}$$

C. Cloud Center

The CC collects the data from APs via wired links. All raw data delivered to the CC is processed and the whole results are forwarded to the user who generates the task. Moreover, the CC determines the task assignment strategy of the whole network, that is, the task division percentage at each AP and ED.

The equivalent raw data arriving speed at the CC forwarded by AP j for the task from ED i can be calculated by

$$\lambda_{\mathrm{CC}}^{j,i} = \phi_{\mathrm{AP}}^{j,i} \cdot \frac{\left(1 - s_{\mathrm{AP}}^{j,i}\right)}{1 - s_{\mathrm{AP}}^{j,i} + \rho s_{\mathrm{AP}}^{j,i} + \frac{\rho s_{\mathrm{ED}}^{j,i}}{1-s_{\mathrm{ED}}^{j,i}}}. \tag{12}$$

The arriving data at the CC include three parts: 1) the remaining raw data; 2) the processing results of the APs; and 3) the processing results of the EDs. The raw data arriving speed is proportional to the remaining raw data volume percentage in the arriving data. Moreover, the computing capacity allocated by the CC to the task transmitted from AP j and ED i is denoted by $\theta_{\mathrm{CC}}^{j,i}$.

We summarize the whole data processing as follows: the whole task flow starts from the generation of the data at the EDs and ends at finishing processing at the CC. After being generated at the EDs, part of the raw data are processed at the EDs, and the processing results together with the remaining raw data are transmitted to the APs. Once receiving the data from the corresponding EDs, the APs offload a part of the raw data to process, and deliver the left raw data, the processing results of themselves as well as the received processing results of the EDs to the CC. The CC will process the remaining raw data and aggregate the processing results at different layers. During the processing and transmitting, the task assignment strategy $\boldsymbol{s}$, the computing capacity each AP to the tasks from different EDs, $\theta_{\mathrm{AP}}^{i,j}$, the computing capacity of the CC allocated to the data delivered by different APs from EDs, $\theta_{\mathrm{CC}}^{j,i}$, the wireless transmission resources allocation $\phi_{\mathrm{ED}}^{j,i}$ and the wired transmission resources allocation, $\phi_{\mathrm{AP}}^{j,i}$, can all be adjusted. The adjustment of the aforementioned variables will be analyzed in the next section.

III. Problem Formulation

In this section, we first clarify the system can be naturally classified into the blocking state and nonblocking state and quantitively derive the boundary of two states. Then. we formulate objectives for both states, respectively.

A. System State Judgement

The total computing capacity of each node, the total wireless transmission resources of each AP, and the total wired transmission resources of the CC in our EdgeFlow model are finite, however, the data generation speed can vary related to the IoT applications, which can even reach a very high speed. Therefore, there is an intuition that when the data generation speed at the EDs exceeds a certain bound, the whole network cannot follow up the data generation speed and the data will accumulate in the buffer of the nodes.[4] We define the *blocking state* as follows.

Definition 1: The blocking state is the state that *no matter how* the system adjusts its computing and transmission resources allocation or adjusts the task division on every node, the data will accumulate in the buffer at least one node.

We then derive the boundary between the nonblocking state and the blocking state. Specifically, the nonblocking system indicates that each layer is nonblocking. Thus, we provide the nonconditions of the EDs, the APs, and the CC, respectively.

1) Nonblocking Conditions of the ED Layer: Since the wireless transmission resources allocated to each ED is determined by its connected AP, the blocking of the ED layer is judged by the AP. The ED layer blocks when the offloaded data volume surpasses the computing capacity of the ED, or the transmitting capacity of the AP is insufficient to support the data transmission from all EDs. We consider the case that all EDs fully use their computing capacity, expressed by

$$\lambda_{\mathrm{ED}}^{j,i} s_{\mathrm{ED}}^{j,i} = \theta_{\mathrm{ED}}^{j,i},\ \forall 1 \le j \le N, 1 \le i \le M \tag{13}$$

which implies that the transmission pressure of the ED is minimum. Under the aforementioned circumstance, if the transmission resources of the AP cannot support the data transmission of all connected EDs, the ED layer will block. Hence, the nonblocking conditions for transmitting are expressed in (3) and (4), when (13) is satisfied.

2) Nonblocking Conditions of the AP Layer: Similarly with EDs, the wired transmission resources allocated to each AP is determined by the CC. The AP layer blocks when the offloaded data volume surpasses the computing capacity of the AP, or the transmitting capacity between the AP layer and the CC is insufficient for the data transmission from all APs. We also consider the case that all APs fully use their computing capacity, expressed by

$$\lambda_{\mathrm{AP}}^{j,i} s_{\mathrm{AP}}^{j,i} = \theta_{\mathrm{AP}}^{j,i},\ \forall 1 \le j \le N, 1 \le i \le M \tag{14}$$

which implies that the transmission pressure of the AP is minimum. Under the aforementioned circumstance, the computing is nonblocking when (9) is satisfied, representing that the total computing capacity allocated to the connected EDs is no more than the computing capacity of the AP. If the allocated wired transmission resources of the AP cannot support its data transmission, or the wired transmission resources of the CC cannot support the data transmission of all APs, the AP layer will block in the transmission. Hence, the nonblocking conditions for transmitting are expressed in (10) and (11), when (14) is satisfied.

3) Nonblocking Conditions of the CC Layer: The CC need to process all remaining raw data transmitted from APs. Hence, the nonblocking conditions of computing at the CC is

$$\lambda_{\mathrm{CC}}^{j,i} \le \theta_{\mathrm{CC}}^{j,i} \tag{15}$$

$$\sum_{j=1}^{N}\sum_{i=1}^{M} \theta_{\mathrm{CC}}^{j,i} \le \theta_{\mathrm{CC}} \tag{16}$$

where (15) represents that the computing capacity allocated by the CC to each task should be able to process the received raw data, and (16) implies that the sum of allocated computing capacity to each task is no more than the total computing capacity of the CC.

Summarizing the nonblocking conditions for the EDs, the APs, and the CC, we have Proposition 1 to clarity the boundary between the nonblocking state and the blocking state.

Proposition 1: The EdgeFlow system is nonblocking *if and only if* the constraints that (1), (3), (4), (7), (9)–(11), (13)–(16) are satisfied.

B. Nonblocking State: Latency Minimization

In the nonblocking state, we address a general objective in edge computing system: to minimize the system latency. The latency of a task is defined as the sum of the computing time and transmitting time from the ED layer to the CC.

When the data is processed at ED i, the latency for the data with the unit size can be calculated as follows:

$$L_{\mathrm{ED}}^{j,i} = \frac{1}{\theta_{\mathrm{ED}}^{j,i}} + \frac{\rho}{\phi_{\mathrm{ED}}^{j,i}} + \frac{\rho}{\phi_{\mathrm{AP}}^{j,i}}. \tag{17}$$

The first term is the processing time for the data at the ED, the second term is the transmission time between the ED and AP, and the third term is the transmission time between the AP and CC. Since the data is processed at ED i, only the results with compressed ratio, ρ, need to be transmitted.

When the data is processed at AP j, the latency for the data with the unit size can be calculated as follows:

$$L_{\mathrm{AP}}^{j,i} = \frac{1}{\phi_{\mathrm{ED}}^{j,i}} + \frac{1}{\theta_{\mathrm{AP}}^{j,i}} + \frac{\rho}{\phi_{\mathrm{AP}}^{j,i}}. \tag{18}$$

The first term is the transmission time between the ED and AP, the second term is the processing time for the data at the AP, and the third term is the transmission time between the AP and CC. Since the data is processed at AP j, only the results with compressed ratio, ρ, need to be transmitted to the CC.

When the data is processed at the CC, the latency for the data with the unit size can be calculated as follows:

$$L_{\mathrm{CC}}^{j,i} = \frac{1}{\phi_{\mathrm{ED}}^{j,i}} + \frac{1}{\phi_{\mathrm{AP}}^{j,i}} + \frac{1}{\theta_{\mathrm{CC}}^{j,i}}. \tag{19}$$

The first term is the transmission time between the ED and AP, the second term is the transmission time between the AP and CC, and the third term is the processing time for the data at the CC.

Definition 2: The *system latency* is the total latency of all tasks created at the EDs per unit time.

Considering the task produced at ED i, the data generation speed of which is $\lambda_{\mathrm{ED}}^{j,i}$, the task division percentage at ED i, AP j and the CC is $s_{\mathrm{ED}}^{j,i}, s_{\mathrm{AP}}^{j,i}$ and $s_{\mathrm{CC}}^{j,i} = 1 - s_{\mathrm{ED}}^{j,i} - s_{\mathrm{AP}}^{j,i}$, respectively. Therefore, the system latency of all links can be formulated as

$$L = \sum_{j=1}^{N}\sum_{i=1}^{M}\Big(\lambda_{\mathrm{ED}}^{j,i} \cdot \Big[s_{\mathrm{ED}}^{j,i} L_{\mathrm{ED}}^{j,i} + s_{\mathrm{AP}}^{j,i} L_{\mathrm{AP}}^{j,i} + s_{\mathrm{CC}}^{j,i} L_{\mathrm{CC}}^{j,i}\Big]\Big). \tag{20}$$

[4]The space of the buffer in each node is viewed as infinite, that is, the data only accumulates in the buffer and no data loss happens when the node blocks.

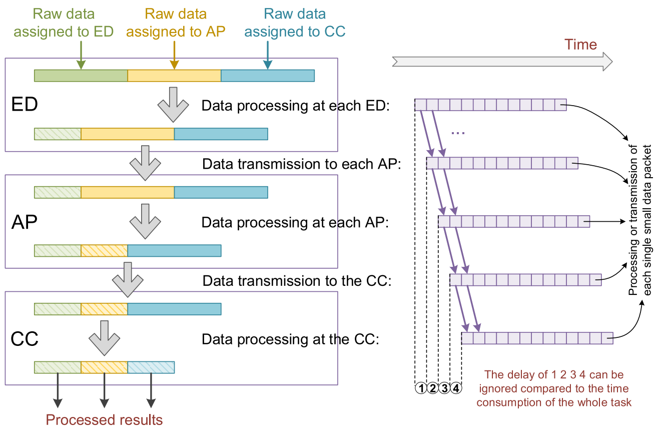

Raw data assigned to ED
Raw data assigned to AP
Raw data assigned to CC
Time
ED
Data processing at each ED:
Data transmission to each AP:
AP
Data processing at each AP:
Data transmission to the CC:
CC
Data processing at the CC:
Processed results
Processing or transmission of each single small data packet
The delay of 1 2 3 4 can be ignored compared to the time consumption of the whole task

Fig. 2. Pipeline of the data processing and transmitting.

Hence, the total latency minimization problem in the nonblocking state can be formulated as

$$\min_{\mathbf{s},\boldsymbol{\theta},\boldsymbol{\phi}} \ L \tag{21}$$
$$s.t. \ (1), (3), (4), (7), (9)-(11), (13)-(16).$$

C. Blocking State: Recovery Time Minimization

In the blocking state, the historical data has already accumulated in the buffer, and thus, the new generated data cannot be processed until accumulated data is processed. Thus, we have the following remark.

Remark 2: The system latency in the blocking state is meaningless since the data is no longer the real-time data.

When the network is blocking, the primary target is naturally set to process the historical data in the buffer and let the network recover to be nonblocking state as far as possible when the data generation speed slows down. We aim to minimize the time of clearing the buffer from the perspective of the whole system.

In the rest of this part, we will quantitively formulate the recovery time minimization.

There are five stages for the data from the time it generates at the EDs to the time it arrives at the CC.

1) *Processing at Each ED:* The processing time of ED i connected with AP j can be expressed by

$$T_{\text{ED}}^{j,i} = \frac{\lambda_{\text{ED}}^{j,i} s_{\text{ED}}^{j,i}}{\theta_{\text{ED}}^{j,i}}. \tag{22}$$

2) *Transmitting to Each AP:* The data to be transmitted to the AP includes the data processed at the ED and the remaining raw data. The transmitting time from ED i to AP j is denoted by

$$t_{\text{ED}}^{j,i} = \frac{\rho\lambda_{\text{ED}}^{j,i} s_{\text{ED}}^{j,i} + \lambda_{\text{ED}}^{j,i}\left(1 - s_{\text{ED}}^{j,i}\right)}{\phi_{\text{ED}}^{j,i}}. \tag{23}$$

3) *Processing at Each AP:* The processing time of AP j is denoted by

$$T_{\text{AP}}^{j} = \frac{\left(\sum_{i=1}^{M} \lambda_{\text{AP}}^{j,i}\right) s_{\text{AP}}^{j}}{\theta_{\text{AP}}^{j}} \tag{24}$$

where s_{AP}^{j} denotes the equivalent task assignment percentage at AP j without regards to the data source.

4) *Transmitting to the CC:* The data to be transmitted from the AP to the CC consists of the data processed at the ED and AP as well as the remaining raw data. The transmitting time from AP j to the CC is represented by

$$t_{\text{AP}}^{j} = \frac{\left(\sum_{i=1}^{M} \lambda_{\text{AP}}^{j,i}\right)\left(1 - s_{\text{AP}}^{j} + \rho s_{\text{AP}}^{j}\right) + \sum_{i=1}^{M} \beta_{\text{AP}}^{j,i}}{\phi_{\text{AP}}^{j}}. \tag{25}$$

5) *Processing at the CC:* The processing time of the CC is denoted by

$$T_{\text{CC}} = \frac{\sum_{j=1}^{N}\sum_{i=1}^{M} \lambda_{\text{CC}}^{j,i}}{\theta_{\text{CC}}}. \tag{26}$$

In the high load situation, the aforementioned five stages can be viewed as a *pipeline*, as shown in Fig. 2, and thus, the longest time among the five stages is the bottleneck for the system to recover from the blocking state.

Definition 3: The *recovery time* is the longest time among the processing time at the EDs, the APs and the CC, as well as the transmission time to the APs and the CC.

Therefore, the recovery time can be expressed by

$$T_r = \max_{1\le i\le M, 1\le j\le N}\left\{T_{\text{ED}}^{j,i}, t_{\text{ED}}^{j,i}, T_{\text{AP}}^{j}, t_{\text{AP}}^{j}, T_{\text{CC}}\right\}. \tag{27}$$

The recovery time minimization problem can be formulated as follows:

$$\min_{\mathbf{s},\boldsymbol{\theta},\boldsymbol{\phi}} \quad T_r, \tag{28}$$

$$s.t. \quad T_r > 1.$$
$$(4),\ (9),\ (11),\ (16). \tag{28a}$$

The constraint (28a) implies that the recovery time is larger than a unit time, which indicates the system is blocking. Other constraints represent that the allocated computing capacity or wireless transmission resources to EDs cannot surpass the those of the AP, and the allocated wired transmission resources to APs cannot surpass that of the CC.

IV. Nonblocking State: Latency Minimization Algorithm

In this section, we design the task assignment and resource allocation algorithm for the latency minimization problem in the nonblocking state. When the EdgeFlow system is in the nonblocking state, we aim to minimize the system latency, as described in (21), which is nonconvex. The latency can be rewritten as

$$\begin{aligned} L(\mathbf{s},\boldsymbol{\theta},\boldsymbol{\phi}) = &\sum_{j=1}^{N}\sum_{i=1}^{M} \lambda_{\text{ED}}^{j,i}\cdot\left(\frac{s_{\text{ED}}^{j,i}}{\theta_{\text{ED}}^{j,i}} + \frac{s_{\text{AP}}^{j,i}}{\theta_{\text{AP}}^{j,i}} + \frac{s_{\text{CC}}^{j,i}}{\theta_{\text{CC}}^{j,i}}\right)\\ &+\sum_{j=1}^{N}\sum_{i=1}^{M} \lambda_{\text{ED}}^{j,i}\cdot\frac{\rho s_{\text{ED}}^{j,i} + s_{\text{AP}}^{j,i} + s_{\text{CC}}^{j,i}}{\phi_{\text{ED}}^{j,i}}\\ &+\sum_{j=1}^{N}\sum_{i=1}^{M} \lambda_{\text{ED}}^{j,i}\cdot\frac{\rho s_{\text{ED}}^{j,i} + \rho s_{\text{AP}}^{j,i} + s_{\text{CC}}^{j,i}}{\phi_{\text{AP}}^{j,i}}. \end{aligned} \tag{29}$$

By utilizing the Cauchy–Schwarz inequality [33], (29) can be divided into several subproblem with the task assignment strategy **s**, computing capacity allocation $\boldsymbol{\theta}$, and transmitting capacity allocation $\boldsymbol{\phi}$ separated. We consider that no spare computing capacity or transmission resource is left, i.e., the equalities are satisfied in (4), (9), and (11).

Hence, we can obtain the following inequation:

$$\begin{aligned} L(\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{\phi}) &\geq L_m(\boldsymbol{s}) \\ &= \frac{\left[\sum_{j=1}^{N}\sum_{i=1}^{M}\left(\sqrt{\lambda_{\text{ED}}^{j,i}s_{\text{ED}}^{j,i}}+\sqrt{\lambda_{\text{ED}}^{j,i}s_{\text{AP}}^{j,i}}+\sqrt{\lambda_{\text{ED}}^{j,i}s_{\text{CC}}^{j,i}}\right)\right]^2}{\Theta_{\text{total}}} \\ &+ \frac{\left(\sum_{j=1}^{N}\sum_{i=1}^{M}\sqrt{\lambda_{\text{ED}}^{j,i}\left(\rho s_{\text{ED}}^{j,i}+s_{\text{AP}}^{j,i}+s_{\text{CC}}^{j,i}\right)}\right)^2}{\Phi_{\text{wireless}}} \\ &+ \frac{\left(\sum_{j=1}^{N}\sum_{i=1}^{M}\sqrt{\lambda_{\text{ED}}^{j,i}\left(\rho s_{\text{ED}}^{j,i}+\rho s_{\text{AP}}^{j,i}+s_{\text{CC}}^{j,i}\right)}\right)^2}{\Phi_{\text{wired}}} \end{aligned} \tag{30}$$

where

$$\Theta_{\text{total}} = \sum_{j=1}^{N}\sum_{i=1}^{M}\theta_{\text{ED}}^{j,i} + \sum_{j=1}^{N}\theta_{\text{AP}}^{j} + \theta_{\text{CC}} \tag{31}$$

$$\Phi_{\text{wireless}} = \sum_{j=1}^{N}\phi_{\text{AP}}^{j} \tag{32}$$

$$\Phi_{\text{wired}} = \phi_{\text{CC}}. \tag{33}$$

Proposition 2: The inequality in (30) is transformed into an equation if and only if the following conditions are satisfied:

$$\begin{aligned} \frac{\theta_{\text{ED}}^{j,i}}{\theta_{\text{ED}}^{j',i'}} &= \frac{\sqrt{\lambda_{\text{ED}}^{j,i}s_{\text{ED}}^{j,i}}}{\sqrt{\lambda_{\text{ED}}^{j',i'}s_{\text{ED}}^{j',i'}}} \\ \theta_{\text{ED}}^{j,i} : \theta_{\text{AP}}^{j,i} : \theta_{\text{CC}}^{j,i} &= \sqrt{\lambda_{\text{ED}}^{j,i}s_{\text{ED}}^{j,i}} : \sqrt{\lambda_{\text{ED}}^{j,i}s_{\text{AP}}^{j,i}} : \sqrt{\lambda_{\text{ED}}^{j,i}s_{\text{CC}}^{j,i}} \\ \frac{\phi_{\text{ED}}^{j,i}}{\phi_{\text{ED}}^{j',i'}} &= \frac{\sqrt{\lambda_{\text{ED}}^{j,i}\left(\rho s_{\text{ED}}^{j,i}+s_{\text{AP}}^{j,i}+s_{\text{CC}}^{j,i}\right)}}{\sqrt{\lambda_{\text{ED}}^{j',i'}\left(\rho s_{\text{ED}}^{j',i'}+s_{\text{AP}}^{j',i'}+s_{\text{CC}}^{j',i'}\right)}} \\ \frac{\phi_{\text{AP}}^{j,i}}{\phi_{\text{AP}}^{j',i'}} &= \frac{\sqrt{\lambda_{\text{ED}}^{j,i}\left(\rho s_{\text{ED}}^{j,i}+\rho s_{\text{AP}}^{j,i}+s_{\text{CC}}^{j,i}\right)}}{\sqrt{\lambda_{\text{ED}}^{j',i'}\left(\rho s_{\text{ED}}^{j',i'}+\rho s_{\text{AP}}^{j',i'}+s_{\text{CC}}^{j',i'}\right)}} \\ &\forall 1 \leq j, j' \leq N, 1 \leq i, i' \leq M. \end{aligned} \tag{34}$$

The equations in (34) imply that the computing capacity division and transmission resources allocation, θ and ϕ, can be derived once the task assignment division percentage, **s**, is determined.

Hence, the latency minimization problem (21) can be converted into

$$\begin{aligned} \min_{s}\ & L_m(\boldsymbol{s}) \\ s.t.\ & (1)-(4),\ (7)-(11),\ (15)-(16) \\ & (31)-(34). \end{aligned} \tag{35}$$

Algorithm 1 Latency Minimization Algorithm

```
Input: Computing capacity $\theta_{ED}^{j,i}, \theta_{AP}^{j}, \theta_{CC}$, wireless transmission resources of each AP $\phi_{AP}^{j}$, wired transmission resources $\phi_{CC}$, data generation speed $\boldsymbol{\lambda}$.
Output: Task assignment strategy $\boldsymbol{s}^*$, resources allocation scheme $\boldsymbol{\theta}^*, \boldsymbol{\phi}^*$.
1: Convert the proportional optimization problem in (29) into the of task assignment problem in (30) by utilizing Cauchy-Schwarz inequality.
2: for all Vertex of the feasible set do
3:     Obtain the corresponding task assignment strategy $\boldsymbol{s}$.
4:     Obtain the resource allocation scheme $\boldsymbol{\theta}, \boldsymbol{\phi}$ according to $\boldsymbol{s}$ and (34).
5:     if Non-blocking constraints are satisfied then
6:         if $L_{min}(\boldsymbol{s}) < L_{min}(\boldsymbol{s}^*)$ then
7:             $L_{min}(\boldsymbol{s}^*) = L_{min}(\boldsymbol{s})$.
8:             Update the optimal $\boldsymbol{s}^*, \boldsymbol{\theta}^*$ and $\boldsymbol{\phi}^*$.
```

Proposition 3: The objective function of the latency minimization problem (35) is concave, and the optimal results $\boldsymbol{s}^*$ are at the vertex of the feasible set bounded by the constraints.

Proof: The function $L_{\min}$ in (35) depends on the vector $\boldsymbol{s} = \{s_{\text{ED}}^{1,1}, \ldots, s_{\text{ED}}^{N,M}, s_{\text{AP}}^{1,1}, \ldots, s_{\text{AP}}^{N,M}\}$. After analyzing the sign of elements in the Hessian matrix of $L_{\min}$, we note that the Hessian matrix is a seminegative definite matrix, implying that the function $L_{\min}$ is concave [34]. Hence, the minimum value of a concave function is obtained at the vertex of the feasible set bounded by the nonblocking constraints presented in Proposition 1.

Take the two-layer subsystem of EdgeFlow as an example, and the subsystem consists of one AP and two EDs. Under different conditions, the minimum value of $L_{\min}$ is at different vertex of the feasible set, as shown in Fig. 3. The computing nonblocking and transmitting nonblocking constraints are shown as the straight lines in the figure, and the value of $L_{\min}$ decreases as the background color becomes darker. The optimal result is marked with the dark spot, which may appear at the crossing point of the computing and transmitting nonblocking constraints [Fig. 3(a) and (b)], or the crossing point of the computing nonblocking constraints [Fig. 3(c)]. ■

According to Proposition 3, we can obtain the optimal results of the latency minimization problem by searching all the vertexes of the feasible set bounded by the constraints. The latency minimization algorithm in the nonblocking state is summarized in Algorithm 1.

V. Blocking State: Recovery Time Minimization Algorithm Design

Considering the case that all APs and EDs fully utilize their computing capacity, implying that the processed data volume of APs and EDs equals the data volume assigned to them, the system blocks if the data volume to be transmitted of any AP or ED surpasses its transmitting capacity, or the amount of the remaining raw data forwarded to the CC surpasses the computing capacity of the CC.

The key idea of minimizing the recovery time, i.e., solving the min-max problem in (28), is that make the computing time and transmitting time equal on the blocking layer. Specifically,

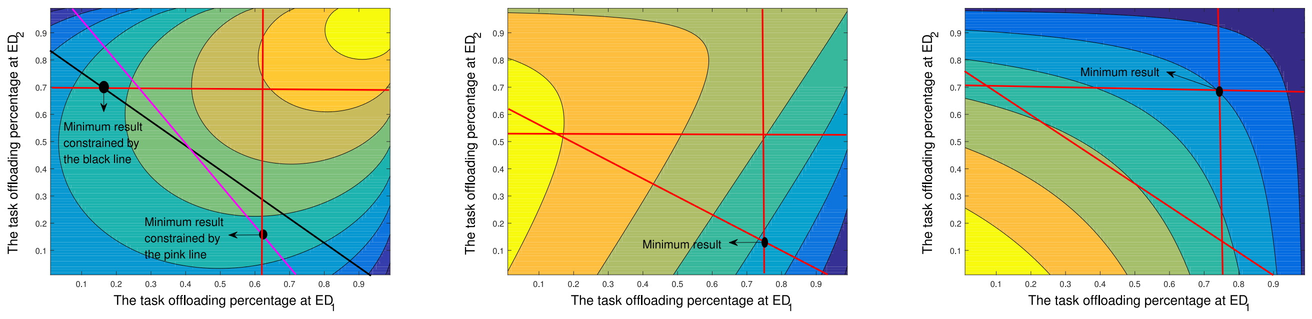


Fig. 3. Typical latency minimization results in the nonblocking state of one AP-two EDs subsystem.

minimize the recovery time of the whole network through minimizing the recovery time from the bottom layer (the EDs) to the top layer (the CC).

A. Minimizing Recovery Time on the ED Layer

When the ED layer blocks, representing that the wireless transmission resources of the AP are insufficient to transmit the processing results as well as the remaining raw data of its connected EDs, the task assignment strategy of the ED layer and the transmission resource allocation of the AP to its connected EDs need to be adjusted. The target is to make equal the computing time and transmitting time of all EDs (including the blocking ED node) connected with the same AP.

We consider the case that the blocking appears between AP j and its connected EDs. Let

$$s_{\mathrm{ED}}^{j,1} = \alpha \in [0, 1] \tag{36}$$

and the total wireless transmission resources of AP j is ϕ_{AP}^{j}. The computing time of EDs and AP j are equal, which can be expressed by

$$T_{\mathrm{ED}}^{j,i} = T_{\mathrm{ED}}^{j,i'}, \ \forall 1 \le i, i' \le M. \tag{37}$$

Therefore, the task assignment percentage of ED i, $2 \le i \le M$ is

$$s_{\mathrm{ED}}^{j,i} = \begin{cases} \frac{\theta_{\mathrm{ED}}^{j,i}\lambda_{\mathrm{ED}}^{j,1}}{\theta_{\mathrm{ED}}^{j,1}\lambda_{\mathrm{ED}}^{j,i}} s_{\mathrm{ED}}^{j,1} = k_i\alpha, & 0 \le \alpha \le \frac{1}{k_i} \\ 1, & \frac{1}{k_i} < \alpha \le 1. \end{cases} \tag{38}$$

The computing time and transmitting time of EDs are equal, which can be expressed by

$$T_{\mathrm{ED}}^{j,i} = t_{\mathrm{ED}}^{j,i}, \ \forall 1 \le i \le M. \tag{39}$$

Hence, the transmission resources of AP j allocated to ED i, $1 \le i \le M$ is

$$\phi_{\mathrm{ED}}^{j,i} = \theta_{\mathrm{ED}}^{j,i} \frac{1 - s_{\mathrm{ED}}^{j,i} + \rho s_{\mathrm{ED}}^{j,i}}{s_{\mathrm{ED}}^{j,i}} = \theta_{\mathrm{ED}}^{j,i} f\left(s_{\mathrm{ED}}^{j,i}\right). \tag{40}$$

Since the total wireless transmission resources of AP j are fixed, that is

$$\sum_{i=1}^{M} \phi_{\mathrm{ED}}^{j,i} = \phi_{\mathrm{AP}}^{j}. \tag{41}$$

Therefore, the task assignment strategy and resources allocation scheme can be obtained by solving the simultaneous of (36), (38), (40), and (41).

B. Minimizing Recovery Time on the AP Layer

When the AP layer blocks, representing that the wired transmission resources of the CC are insufficient to transmit the processing results as well as the left raw data of all APs, the task assignment strategy of the ED layer and AP layer as well as the transmission resources allocation of the APs and the CC need to be adjusted. The target is to make equal the computing time and transmission time of all APs as well as the computing time of all EDs.

The task division percentage of each ED and AP should be smaller than one, expressed by

$$0 \le s_{\mathrm{ED}}^{j,i} \le 1, 0 \le s_{\mathrm{AP}}^{j} \le 1, \ \forall 1 \le j \le N, 1 \le i \le M. \tag{42}$$

To fully utilize the computing capacity of all EDs and APs, the computing times of all EDs and APs are equal

$$T_{\mathrm{ED}}^{j,i} = T_{\mathrm{ED}}^{j',i'} = T_{\mathrm{AP}}^{j}, \ \forall 1 \le j, j' \le N, 1 \le i, i' \le M. \tag{43}$$

The computing time and transmitting time of each AP are equal, which can be expressed by

$$T_{\mathrm{AP}}^{j} = t_{\mathrm{AP}}^{j}, \ \forall 1 \le j, j' \le N. \tag{44}$$

Moreover, the summations of allocated wired transmission resources are equal to that of the CC

$$\sum_{j=1}^{N} \phi_{\mathrm{AP}}^{j} = \phi_{\mathrm{CC}}. \tag{45}$$

Similarly as the analyzing the case of ED layer blocking, the task assignment strategy and resources allocation scheme can be obtained by solving the simultaneous of (42)–(45).

C. Minimizing Recovery Time on the CC

When the CC blocks, representing that the computing capacity of the CC are insufficient to process the remaining raw data, the task assignment strategy of the CC, AP, and ED layer as well as the transmission resources allocation of the APs and the CC need to be adjusted. The target is to make equal the computing time and transmission time of the CC as well as the computing time of all APs and EDs.

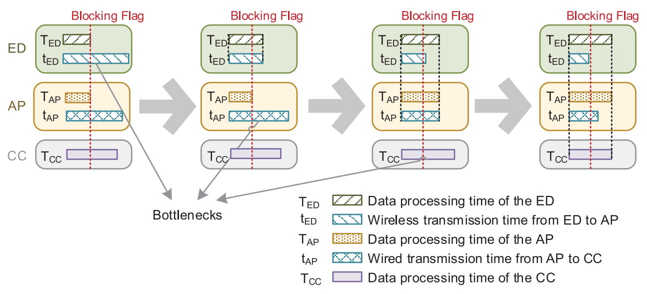

Blocking Flag
ED
AP
CC
T_{ED}
t_{ED}
T_{AP}
t_{AP}
T_{CC}
Bottlenecks
T_{ED} Data processing time of the ED
t_{ED} Wireless transmission time from ED to AP
T_{AP} Data processing time of the AP
t_{AP} Wired transmission time from AP to CC
T_{CC} Data processing time of the CC

Fig. 4. Illustration of task assignment strategy in the *one ED-one AP-CC* system.

Algorithm 2 Recovery Time Minimization Algorithm

```
Input: Computing capacity $\theta_{ED}^{j,i}, \theta_{AP}^{j}, \theta_{CC}$, wireless transmission resources of each AP $\phi_{AP}^{j}$, wired transmission resources $\phi_{CC}$, data generation speed $\boldsymbol{\lambda}$.
Output: Task assignment strategy $\boldsymbol{s}*$, resources allocation scheme $\boldsymbol{\theta}*, \boldsymbol{\phi}*$.

ED layer optimization:
1: Fully utilize the computing capacity of all EDs.
2: if The ED layer blocks then
3:     Make equal of the computing and transmitting time of all EDs connected with the same AP in the blocking area.
4:     Update $\boldsymbol{s}*$, $\boldsymbol{\theta}*$ and $\boldsymbol{\phi}*$.
AP layer optimization:
1: Fully utilize the computing capacity of all APs and EDs.
2: if The AP layer blocks then
3:     Make equal of the computing time of all APs and EDs.
4:     Make equal of the computing and transmitting time of APs.
5:     Update $\boldsymbol{s}*$, $\boldsymbol{\theta}*$ and $\boldsymbol{\phi}*$.
CC layer optimization:
1: Fully utilize the computing capacity of all APs, EDs and the CC.
2: if The CC layer blocks then
3:     Make equal of the computing time of all APs, EDs and CC.
4:     Update $\boldsymbol{s}*$, $\boldsymbol{\theta}*$ and $\boldsymbol{\phi}*$.
```

Similarly as the analyzing the case of ED layer blocking, the task assignment strategy and resources allocation scheme can be obtained by solving the simultaneous equations that follows:

$$0 \le s_{\mathrm{ED}}^{j,i} \le 1, 0 \le s_{\mathrm{AP}}^{j} \le 1, \ \forall 1 \le j \le N, 1 \le i \le M \quad (46)$$

$$T_{\mathrm{ED}}^{j,i} = T_{\mathrm{ED}}^{j',i'} = T_{\mathrm{AP}}^{j} = T_{\mathrm{CC}}, \forall 1 \le j, j' \le N, 1 \le i, i' \le M. \quad (47)$$

As an example, the task assignment strategy in the *one ED-one AP-CC* system is illustrated in Fig. 4.

The recovery time minimization algorithm in the blocking state is summarized in Algorithm 2.

VI. Implementation of the Experiment

In this section, we establish the three-layer EdgeFlow system consisting of the CC, APs, and EDs based on the Linux system, USRP, and the Intel NUCs[5] [35]. The EdgeFlow platform is available in [36].

[5]Next unit of computing (NUC) is a small-form-factor personal computer designed by Intel.

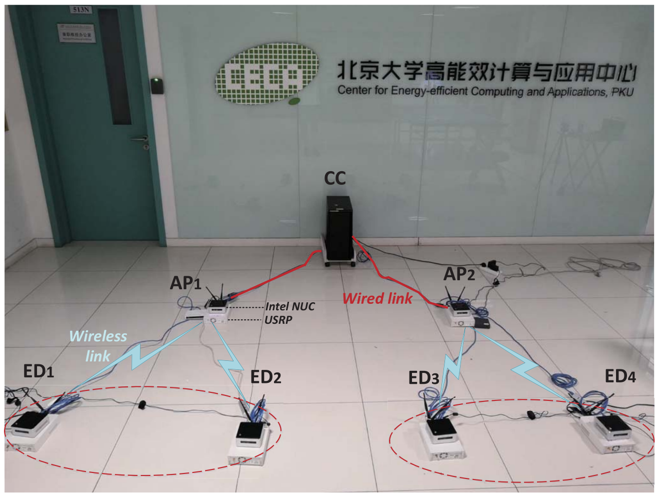

北京大学高能效计算与应用中心
Center for Energy-efficient Computing and Applications, PKU
CC
AP1
AP2
Intel NUC
USRP
Wired link
Wireless link
ED1
ED2
ED3
ED4

Fig. 5. EdgeFlow system implementation.

The USRP is a range of software-defined radios, which can realize the general radio communication system [37]. Most USRPs is composed with the hardware part as the radio front end, and the software part, GNU Radio, which is a free software toolkit that provides signal processing blocks [38]. The operating principle of the USRP is that a host computer process the signals based on the GNU Radio, and the processed signals are delivered to the USRP performing as the radio front end through the wired links [39].

The implementation of the EdgeFlow system is presented in Fig. 5. One single server stands for the CC layer, and two NUC nodes communicate with it performing as two APs. With the USRP devices, each AP node connects to two ED nodes over the wireless links with limited resources.

The EdgeFlow system is based on the Java and Python environment, where Java environment is responsible for the task assignment and processing and Python environment realizes the time division multiple access resources management of the network. The main modules for the implementation of the EdgeFlow system are introduced as below.

1) *Network Initialization Module:* The network initialization module has three main functions. First, it establishes the network connection of the wireless channel . Second, it virtualize and manage the transmission resources for the wireless and wired links. Third, it provides the application program interface to the upper framework.
2) *System Management Module:* The system management module includes the system initialization, logical graph establishment and management. The system is connected according to the communication links and Internet protocol address provided by the network initialization module. Moreover, it is responsible for the node registration and resource configuration information updating. The nodes estimate their idle computing and transmission resources, and register on the CC. The CC can then create a logical graph of the nodes with their information of the available resources.

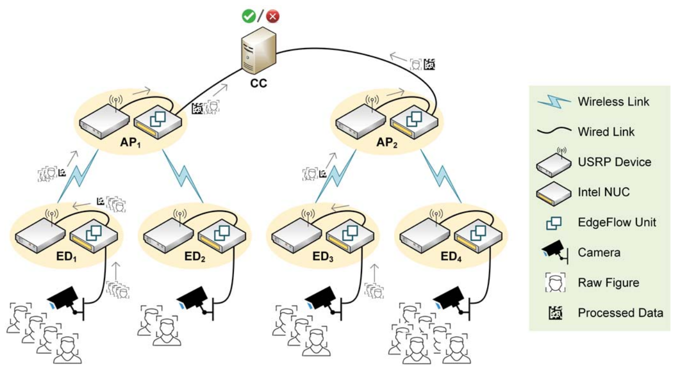


Fig. 6. IoT face recognition scenario based on the EdgeFlow system.

3) *Resource Management Module:* The idle computing and transmission resources are evaluated in the resource management module.
4) *Task Assignment Module:* The task assignment module targets at obtaining the task assignment strategy based on the idle resources of each node.
5) *Task Execution Module:* The main function of the task execution module is the management of the task assignment strategy. According to the task assignment configuration file, which instructs the task assignment strategy of each node, the module offloads the task and manage the task queue of offloading and processing.

The procedures of running the EdgeFlow system consist of the task notification, system registration, task assignment, and data processing. The task is submitted to the CC by the user, and broadcast to the APs as well as EDs. After receiving the task notification, each node estimates its idle computing and transmission resources and uploads them as well as the registration information to the CC. The CC then designs the task assignment strategy and resources allocation scheme based on the logical graph of the EdgeFlow system, and broadcasts to the whole system. Different nodes in the EdgeFlow system process and transmit data according to the received task assignment strategy and resources allocation scheme.

VII. Simulation and Experiment Results

In this section, we evaluate the performance of the EdgeFlow system. The evaluations are accomplished by simulating a typical IoT scenario similar to the face recognition application, which is general in the IoT sensing network, such as smart cities. Based on the face information, the computing servers can provide the intelligent service to the users.

A. Experiment Scenario and Setup

As shown in Fig. 6, in our face recognition scenario, each ED is connected with a camera which collects the image data. The application aims to recognize the pedestrian faces and slice out the face part, which will be delivered to the CC. After the analysis of the face part, the CC will perform the appropriate action. The face recognition is based on openCV, which is a cross-platform open-source computer vision library that suits both servers and mobile devices.

TABLE I
EdgeFlow Platform Specifications

The CPU frequency of the ED device	$1 * 10^9$ Hz
The CPU frequency of the AP device	$3 * 10^9$ Hz
The CPU frequency of the CC device	$1 * 10^{10}$ Hz
Wireless transmission resources of each AP	300 Kbps
Wired transmission resources	1 Mbps
The transmission power of USRP devices	20 dbm
The wireless transmission bandwidth	5 Mhz
The volume of the data file	60 Kbytes
Compression ratio ρ	10%

We run the numerical simulation based on the Java platform to simulate the computation and transmission procedures. The simulation and experimental parameters are listed in Table I. To distinguish the computing capabilities among various layers, the computing frequency is used to measure the computing capacity. Each data file represents one image of the camera, and the data generation speed λ is the number of face images captured by the camera per unit time. Moreover, in our experiment, the data generation follows Poisson point process. The system performance is evaluated by the following indicators.

1) *System Latency:* The system latency represents the response time from the data generation of the task to the end of processing at the CC.
2) *Processing Rate:* This is the average volume of data processed by the EdgeFlow system per unit time.
3) *System Robustness:* This reflects the average number of the unprocessed packages in the network, which represents the degree of blocking in the system.

B. Simulation and Experimental Results

In this section, we evaluate the performance of the dynamic task assignment strategy in the EdgeFlow system. To attest the effect of our proposed algorithm, we compare our algorithm with the following solutions.

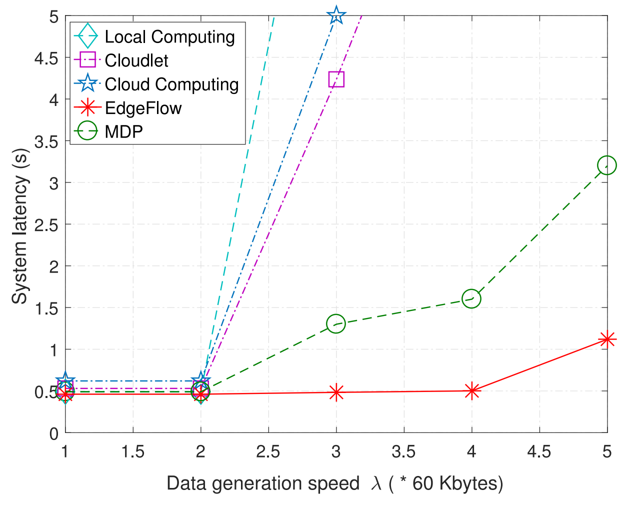
1) Local Computing: Each ED processes all the tasks locally and delivers the results to the CC. This is suitable for the case that the task load is light and the ED is able to process the tasks timely.

2) Cloudlet: The ED assigns the tasks to the corresponding AP [17]. The AP will process all tasks and deliver the results to the CC. This is suitable for the case that the tasks are resource-intensive and the AP possesses abundant computing and transmission resources.

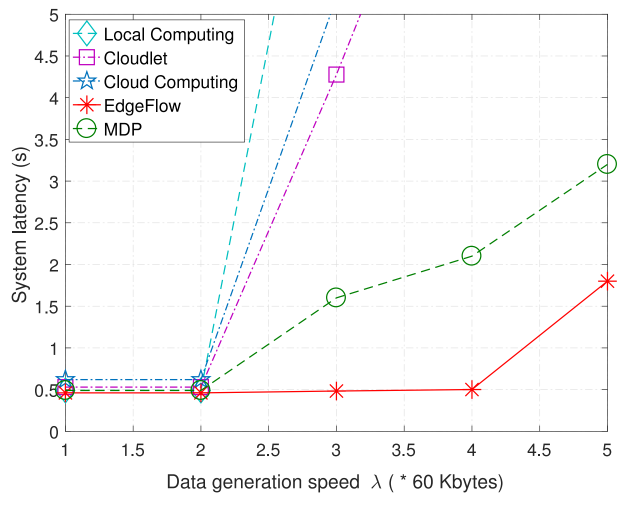
3) Cloud Computing: The input data stream is forwarded to CC directly, and all the tasks is processed centrally at the CC. This is suitable for the case that the tasks are resource-intensive, while the edge nodes (including the APs and EDs) do not possesses enough resources to process the tasks.

4) MDP: Based on MDP, a partial assignment scheme is designed based on the queueing state, the execution state, and the transmission state to minimize the task latency [27].

In our experiment, we observe the average task latency with different data generation speeds. More task data requires more computing and transmission resources. This experiment evaluates the effect of the proposed strategy given different task loads. As depicted in Fig. 7, the consistency of the numerical



(a)



(b)

Fig. 7. System latency versus the data generation speed in the simulation and experiment. (a) Simulation results. (b) Experimental results.

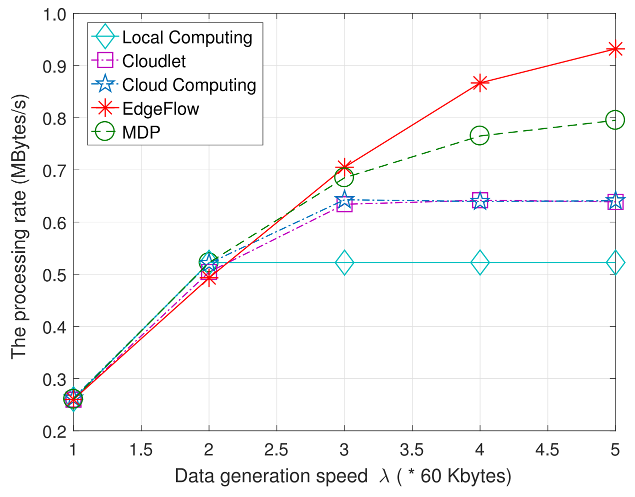


Fig. 8. Processing rate with the increase of the data generation speed on the actual experimental platform.

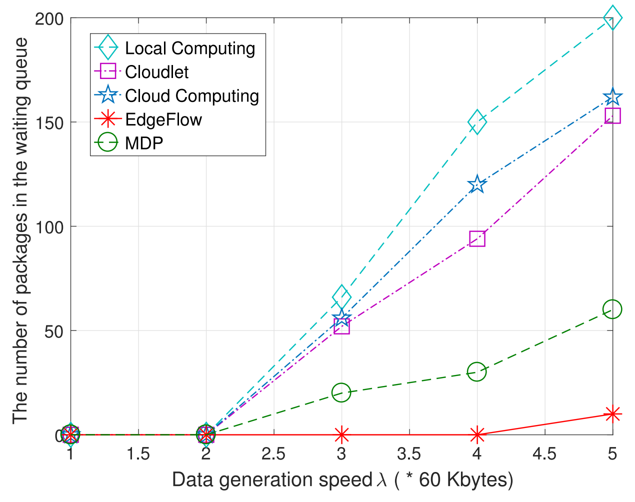


Fig. 9. System robustness with the different burden of tasks on the actual experimental platform.

simulation results and experimental results shows the effect of our EdgeFlow system. It is obvious that our proposed scheme is superior in most cases. Since the EdgeFlow system fully utilizes the computing capacity of the CC, APs, and EDs, the data processing ability is stronger than Cloudlet, local, or cloud computing, which only utilize part of the system computing capacity. However, in other schemes, when the data generation speed $\lambda \geq 3$, the system may start to run out of the resources, and unprocessed data may have accumulated. Compared with other schemes, our EdgeFlow system is more tolerate to the data generation speed, that is, though the data generation speed variate, our EdgeFlow can still provide stable low-latency services.

As shown in Fig. 8, we analyze the processing rate of the system with different data generation speed. When the generation speed $\lambda \leq 2$, the system does not reach the bottleneck of the computing capacity, and thus, all the schemes can process the input task timely. When the generation rate $\lambda \geq 3$, the EdgeFlow system tries its best to guarantee the nonblocking condition by sharing the task overload among multiple layers, which effectively eases the accumulation of the unprocessed tasks and increases the processing rate. In other schemes, however, the data starts to accumulate in the buffer once the processing rate reaches saturation. For example, when $\lambda = 5$, the processing rate of EdgeFlow is 15% higher than the MDP scheme and 43% higher than the local computing scheme.

As shown in Fig. 9, we evaluate the system robustness for the tasks with heavy loads, which can be reflected by the number of unprocessed packages waiting in the buffer. The length of the waiting queue represents the degree of the blocking in the system. When $\lambda \leq 2$, the computing and transmission resources can still handle the input tasks, and thus all schemes do not result in the data accumulation. When $\lambda \geq 3$, the resources of the AP or CC cannot guarantee the stable operation of the system. The EdgeFlow system balances the computing and transmission resources in the multilayer network, which can maintain the stability of the system as much as possible. Other system, e.g., Cloudlet, offloads all computing tasks to the AP, which bring about heavy loads to wireless

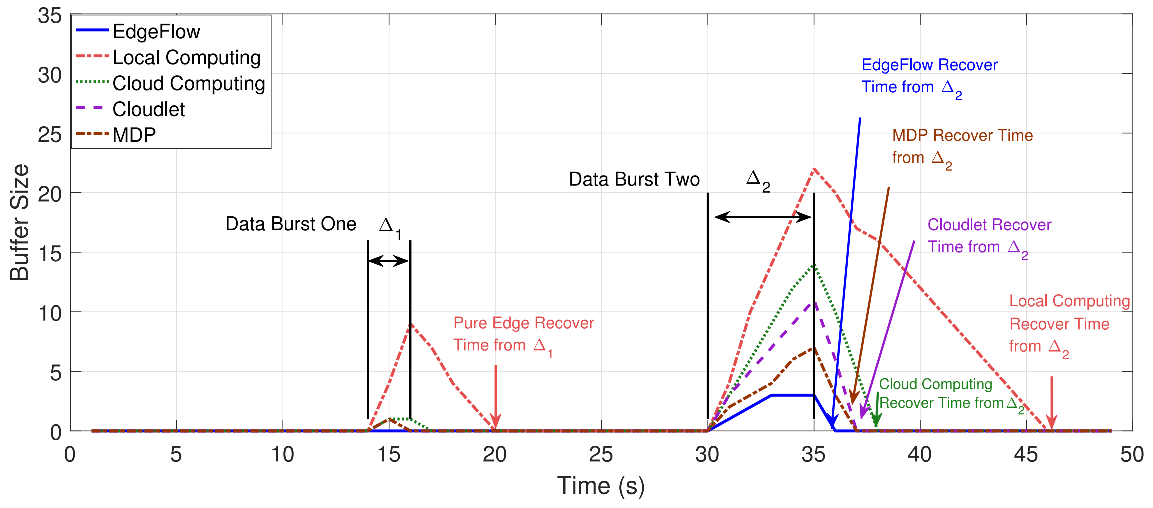


Fig. 10. Recovery time after the different burden of tasks on the actual experimental platform.

links while lots of wired resources may still be idle, resulting in the heavier data accumulation. Hence, the EdgeFlow system performs better than other schemes when processing tasks with heavy loads.

As depicted in Fig. 10, we analyze the system robustness in the perspective of the time for the system to recover from the blocking time after the data burst. At the time $t = 14$ s, the first burst lasting 2 s only causes the data accumulation for the local computing scheme, while the others are barely affected. After that, at the time $t = 30$ s, a bigger burst lasting 5 s arrives and affects all schemes. When dealing with data burst, the system suddenly turns from the nonblocking to the blocking state. Other schemes, i.e., the cloud computing, local computing, and Cloudlet, remain the same task assignment strategy, which is not suitable for the blocking state since clearing the accumulated data in the buffer becomes the primary mission. The MDP scheme adjust the task assignment strategy based on the queue state, which is hysteretic than the change of data generation speed. Compared with other schemes, EdgeFlow determines the optimal task assignment strategy based on the data generation speed and system state, and thus guarantees the smallest volume of accumulated data and the shortest recovery time, which reflects that the EdgeFlow system is most robust among these schemes, especially for the tasks with heavy loads.

VIII. Conclusion

In this paper, we have proposed a multilayer data flow processing system EdgeFlow, which consists of the CC, APs, and the EDs. The EdgeFlow system can provide the low latency services for the IoT real-time applications via the integrated utilization of the computing capacity and transmission resource of both CC and edge nodes. The blocking and nonblocking states have been investigated and the quantitive boundary between the two states has been derived in Proposition 1. In the nonblocking state, the system latency is minimized, while in the blocking state, the latency is meaningless for the accumulated data and the recovery time of the system is minimized. The multilayer collaborative task assignment and resource allocation strategies have been proposed in Algorithms 1 and 2 for both states to achieve the optimal solutions. The implementation of the EdgeFlow system is based on the USRPs, the Intel NUCs, and the Linux system for the typical IoT applications, face recognition. Experimental results have shown that our EdgeFlow system can obviously reduce the system latency and increase the data processing rate, especially in the case of high data generation speed. The system is able to stay in the nonblocking state by the dynamic task assignment strategy and the resources allocation when the data generation speed increases, and thus the volume of accumulated data in the buffer remains small.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[2] H. Zhang *et al.*, "Cellular Internet-of-Things (IoT) communications over unlicensed band," in *Proc. IEEE DySPAN*, Seoul, South Korea, Oct. 2018, pp. 1–10.

[3] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, "Internet of Things: Vision, applications and research challenges," *Ad Hoc Netw.*, vol. 10, no. 7, pp. 1497–1516, Sep. 2012.

[4] N. C. Luong *et al.*, "Data collection and wireless communication in Internet of Things (IoT) using economic analysis and pricing models: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2546–2590, 4th Quart., 2016.

[5] A. M. Vilamovska *et al.*, *RFID Application in HealthCare—Scoping and Identifying Areas for RFID Deployment in HealthCare Delivery*, RAND Europe, Cambridge, U.K., Feb. 2009.

[6] C. Buckl *et al.*, "Services to the field: An approach for resource constrained sensor/actor networks," in *Proc. WAINA*, Bradford, U.K., May 2009, pp. 476–481.

[7] Y. Meng *et al.*, "WiVo: Enhancing the security of voice control system via wireless signal in IoT environment," in *Proc. ACM Mobihoc*, Los Angeles, CA, USA, Jun. 2018, pp. 81–90.

[8] F. Tang, Z. M. Fadlullah, B. Mao, and N. Kato, "An intelligent traffic load prediction based adaptive channel assignment algorithm in SDN-IoT: A deep learning approach," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2018.2838574.

[9] D. Evans, "The Internet of Things: How the next evolution of the Internet is changing everything," San Jose, CA, USA, CISCO, White Paper, Apr. 2011.

[10] A. Papageorgiou, B. Cheng, and E. Kovacs, "Real-time data reduction at the network edge of Internet-of-Things systems," in *Proc. CNSM*, Barcelona, Spain, Nov. 2015, pp. 284–291.

[11] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato, "A survey on network methodologies for real-time analytics of massive IoT data and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1457–1477, 3rd Quart., 2017.

[12] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[13] *Self-Driving Cars Will Create 2 Petabytes of Data, What Are the Big Data Opportunities for the Car Industry?* Accessed: Dec. 7, 2016. [Online]. Available: https://datafloq.com/read/self-driving-cars-create-2-petabytes-data-annually/172

[14] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," Sophia Antipolis, France, ETSI, White Paper, Sep. 2015.

[15] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.

[16] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[17] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.

[18] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[19] B. Di, L. Song, Y. Li, and G. Y. Li, "Non-orthogonal multiple access for high-reliable and low-latency V2X communications in 5G systems," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 10, pp. 2383–2397, Oct. 2017.

[20] P. Wang, B. Di, H. Zhang, K. Bian, and L. Song, "Cellular V2X communications in unlicensed spectrum: Harmonious coexistence with VANET in 5G systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5212–5224, Aug. 2018.

[21] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. ACM Mobile Cloud Comput.*, Helsinki, Finland, Aug. 2012, pp. 13–16.

[22] H. Li, H. Zhu, S. Du, X. Liang, and X. Shen, "Privacy leakage of location sharing in mobile social networks: Attacks and defense," *IEEE Trans. Depend. Secure Comput.*, vol. 15, no. 4, pp. 646–660, Jul./Aug. 2018.

[23] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, "Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration," *IEEE Trans. Comput.*, vol. 67, no. 9, pp. 1287–1300, Sep. 2018.

[24] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017.

[25] S. Yi *et al.*, "LAVEA: Latency-aware video analytics on edge computing platform," in *Proc. ACM/IEEE SEC*, San Jose, CA, USA, Oct. 2017, pp. 183–196.

[26] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4514–4526, May 2018.

[27] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE ISIT*, Barcelona, Spain, Jul. 2016, pp. 1451–1455.

[28] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[29] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[30] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultra-dense IoT networks," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2018.2838584.

[31] S.-W. Ko, K. Han, and K. Huang, "Wireless networks for mobile edge computing: Spatial modeling and latency analysis," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5225–5240, Aug. 2018.

[32] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[33] J. M. Steele, *The Cauchy-Schwarz Master Class: An Introduction to the Art of Mathematical Inequalities*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[34] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[35] C. Yao, X. Wang, Z. Zheng, G. Sun, and L. Song, "EdgeFlow: Open-source multi-layer data flow processing in edge computing for 5G and beyond," *arXiv preprint arXiv: 1801.02206v2*.

[36] *The EdgeFlow Framework*. Accessed: Apr. 19, 2018. [Online]. Available: https://github.com/sirius93123/EdgeFlow

[37] M. Ettus and M. Braun, "The universal software radio peripheral (USRP) family of low-cost SDRs," in *Opportunistic Spectrum Sharing and White Space Access: The Practical Reality*. Hoboken, NJ, USA: Wiley, Jul. 2015

[38] E. Blossom, "GNU radio: Tools for exploring the radio frequency spectrum," *Linux J.*, vol. 2004, no. 122, p. 4, Jun. 2004.

[39] H. Zhu *et al.*, "You can jam but you cannot hide: Defending against jamming attacks for geo-location database driven spectrum sharing," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 10, pp. 2723–2737, Oct. 2016.

[40] L. Tang and S. He, "Multi-user computation offloading in mobile edge computing: A behavioral perspective," *IEEE Netw.*, vol. 32, no. 1, pp. 48–53, Jan./Feb. 2018.

Pengfei Wang (S'17) received the B.S. degree in electronic engineering from Peking University, Beijing, China, in 2017, where he is currently pursuing the master's degree with the School of Electrical Engineering and Computer Science.

His current research interest includes wireless communications, vehicular networks, and edge computing.

Chao Yao (S'15–M'18) received the B.S. and M.S. degrees in electronic engineering from Peking University, Beijing, China, in 2015 and 2018, respectively.

He is currently an Engineer with Bitmain Company, Beijing. His current research interests include edge computing and full-duplex.

Zijie Zheng (S'14) received the B.S. degree in electronic engineering from Peking University, Beijing China, in 2014, where he is currently pursuing the Ph.D. degree with the School of Electrical Engineering and Computer Science.

His current research interests include game theory and optimization in 5G networks, wireless powered networks, mobile social networks, and wireless big data.

Guangyu Sun (M'14) received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 2003 and 2006, respectively, and the Ph.D. degree in computer science from Pennsylvania State University, State College, PA, USA, in 2011.

He is an Associate Professor with the Center for Energy-Efficient Computing and Applications, Peking University, Beijing. His current research interests include computer architecture, electronic design automation, and acceleration system for modern applications.

Dr. Sun is a member ACM and CCF. He is currently serving as an associate editor of the *ACM Journal on Emerging Technologies in Computing Systems* and the *ACM Transactions on Embedded Computing Systems*.

Lingyang Song (S'03–M'06–SM'12) received the Ph.D. degree from the University of York, York, U.K., in 2007.

He was a Research Fellow with the University of Oslo, Oslo, Norway. He joined Philips Research, Cambridge, U.K., in 2008. In 2009, he joined the School of Electronics Engineering and Computer Science, Peking University, Beijing, China, where he is currently a Boya Distinguished Professor. His current research interests include wireless communication and networks, signal processing, and machine learning.

Dr. Song was a recipient of the IEEE Leonard G. Abraham Prize in 2016, the IEEE Asia–Pacific Young Researcher Award in 2012, and the K. M. Stott Prize for Excellent Research. He has been an IEEE Distinguished Lecturer since 2015.