

# Delay-performance optimization resource scheduling in many-to-one multi-server cellular edge computing systems

Du Peng<sup>1</sup> Ba Teer<sup>2</sup> Zhang Yuan<sup>2</sup>

(<sup>1</sup>College of Automation & College of Artificial Intelligence, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

(<sup>2</sup>National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China)

**Abstract:** To further reduce the delay in cellular edge computing systems, a new type of resource scheduling algorithm is proposed. Without assuming the knowledge of the statistics of user task arrival traffic, the analytical formulae of the communication and computing queueing delays in many-to-one multi-server cellular edge computing systems are derived by using the arriving curve and leaving curve. Based on the analytical formulae, an optimization problem of delay minimization is directly formulated, and then a novel scheduling algorithm is designed. The delay performance of the proposed algorithm is evaluated via simulation experiments. Under the considered simulation parameters, the proposed algorithm can achieve 12% less total delay, as compared to the traditional algorithms. System parameters including the weight, the amount of computing resources provided by servers, and the average user task arrival rate have impact on the percentage of delay reduction. Therefore, compared with the queue length optimization based traditional scheduling algorithms, the proposed delay optimization-based scheduling algorithm can further reduce delay.

**Key words:** cellular system; delay; edge computing; resource scheduling

**DOI:**10.3969/j.issn.1003-7985.2019.03.008

Cellular users tend to run computation-intensive applications which require many computing resources. To meet this requirement, the idea of cellular edge computing is introduced. First, one or more off-the-shelf computing units or servers are located in the base station (BS) to form a small-scale computing resource pool<sup>[1-2]</sup>. Then, users' computing tasks are offloaded to the BS to run. The so-formed system is called the cellular edge computing system<sup>[3-4]</sup>. In such a system, there are two types of resources: the radio resource and computing resource. Therefore, the resource scheduling in the cellular

edge computing system must take both types of resources into consideration. This paper studies the resource scheduling in the cellular edge computing system.

This paper focuses on the delay aspect of the resource scheduling. In cellular edge computing systems, there are four types of delays: transmission delay, communication queueing delay, execution delay and computing queueing delay. According to the way of incorporating delays into resource scheduling, existing works can be classified into three categories. For the first category, only transmission and execution delays are considered, but the communication and computing queueing delays are not considered<sup>[5-8]</sup>. For the second category, in addition to the transmission and execution delays, the queueing related delays are also considered. Furthermore, this type of algorithm assumed that the queue can be modelled as the traditional queue (e.g., the M/M/1 queue) so that the delay formulae of the queueing theory can be readily applied in the problem formulation<sup>[9-12]</sup>. For the third category of the algorithm, the queueing related delays are also considered. However, instead of using queueing theory's delay formulae, Little's law is used in this category. According to Little's law, the average delay is proportional to the average queue length. Therefore, the delay minimization problem was transformed into the queue length minimization problem. Then, the scheduling algorithms were derived<sup>[13-16]</sup>.

Generally, the third category of the algorithm is better than the second category, since it does not need any assumption on traffic's statistics. However, the third category of the algorithm depends heavily on Little's law. Actually, Little's law can be inaccurate. Here is an example. Consider a queue where there are two newly arrived tasks at the beginning of each slot and each task needs half slot time to serve. Let  $Q(n)$  denote the number of backlogged tasks at the end of slot  $n$ . Then,  $Q(n)$  is always 2. On the one hand, according to Little's Law, the average delay is 1 slot. On the other hand, it can be verified that the delay of the odd-numbered and even-numbered tasks is 0.5 and 1 slot, respectively, so the true average delay is 0.75 slot. Therefore, Little's law is inaccurate for this example. The reason is that the queue length is not sampled frequently enough. Actually, for this example, if the queue length is sampled once every

**Received** 2019-01-13, **Revised** 2019-05-10.

**Biographies:** Du Peng (1971—), male, doctor, lecturer; Zhang Yuan (corresponding author), male, doctor, associate professor, y. zhang@seu.edu.cn.

**Foundation item:** The National Natural Science Foundation of China (No. 61571111).

**Citation:** Du Peng, Ba Teer, Zhang Yuan. Delay-performance optimization resource scheduling in many-to-one multi-server cellular edge computing systems[J]. Journal of Southeast University (English Edition), 2019, 35(3): 325 – 331. DOI: 10.3969/j.issn.1003-7985.2019.03.008.

half-slot, the queue length will be 1 or 2 alternately, and then the average delay can be correctly calculated according to Little's law.

Motivated by the above observations, this work tries to improve the third category of the algorithm by dropping the dependence on Little's law. The key is to derive the formulae of the delays when there is no assumption on the traffic's statistics so that the scheduling algorithm can be proposed by solving the delay minimization problem directly without using Little's law.

## 1 System Model

Consider a cell consisting of one BS and  $I$  users, in which time is slotted and the duration of each slot is  $T_{\text{slot}}$  (unit:s). Let  $E_i$  denote the number of cycles needed by the task of user  $i$ . For simplicity and without loss of generality, assume that  $E_1 \leq E_2 \leq \dots \leq E_I$ . Assume that there are  $J$  computing servers in the BS. Each server  $j$  can provide  $F_j$  cycles per second. Then, it will take  $E_i/F_j$  of server  $j$  to execute one task of user  $i$ . For convenience, let  $\alpha_{ij} = E_i/F_j/T_{\text{slot}}$  denote the normalized execution time of one task of user  $i$  in server  $j$ .

A task experiences two delays: the communication delay which is the sum of communication queue waiting time and transmission time, and the computing delay which is the sum of computing queue waiting time and execution time. The formulae of these two delays are derived as follows.

### 1.1 Communication delay

First, we derive expressions describing the evolution of communication queues. Let  $A_i(n)$  denote the number of newly arrived computing tasks of user  $i$  during slot  $n$ . Let  $N_i(n)$  denote the number of tasks of user  $i$  selected by the resource scheduler during slot  $n$ . Furthermore, let  $U_i(n)$  denote the number of backlogged tasks of user  $i$  at the end of slot  $n$ , which can be expressed as

$$U_i(n) = U_i(n-1) - N_i(n) + A_i(n) \quad (1)$$

where  $U_i(0) = 0$ . There are three constraints on  $N_i(n)$ . First,  $N_i(n)$  must be the integer. Secondly,  $N_i(n)$  must not exceed the backlogged tasks in the queue, that is

$$0 \leq N_i(n) \leq U_i(n-1) \quad (2)$$

Thirdly,  $X_i(n)$  must be limited by the communication capability of the air interface. For this constraint, some notations are needed. Let  $R_i$  denote the number of subcarriers needed by user  $i$  to offload a task request to the BS in one slot. The value of  $R_i$  depends on both the number of bits of each task request of user  $i$  and the channel condition of user  $i$  (e. g., path loss, fading, etc.). Specifically, the better the channel condition of user  $i$ , the smaller the value of  $R_i$ . Assume that there are a total of  $R$  subcarriers in the cellular edge computing system, and then this constraint can be expressed as

$$\sum_{i=1}^I R_i N_i(n) \leq R \quad (3)$$

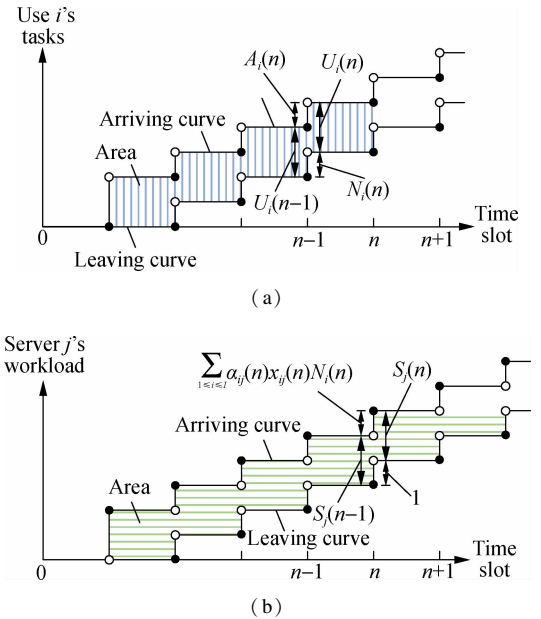
Next, we derive the formula of communication delay. For user  $i$ , let  $\Omega_i(n) = \sum_{1 \leq k \leq n} A_i(k)$  denote the total number of arrivals until slot  $n$ . Similarly, let  $\Xi_i(n) = \sum_{1 \leq k \leq n} N_i(k)$  denote the total number of departures until slot  $n$ . In the literature,  $\Omega_i(n)$  and  $\Xi_i(n)$  are also known as the arriving curve and leaving curve, respectively, as illustrated in Fig. 1(a). The total area between these two curves up to slot  $n$  represents the total time all tasks have spent in the queue during the interval  $(0, nT_{\text{slot}})$ . Let  $\hat{W}_i(n)$  denote this area until slot  $n$ . We can calculate  $\hat{W}_i(n) = \sum_{1 \leq k \leq n} U_i(k) T_{\text{slot}}$ , corresponding to the shadowed area in Fig. 1(a). For convenience, let  $W_i(n)$  denote the normalized version of  $\hat{W}_i(n)$ , i. e.,  $W_i(n) = \hat{W}_i(n)/T_{\text{slot}}$ . Then, we can have the formula of  $W_i(n)$  as

$$W_i(n) = W_i(n-1) + U_i(n-1) \quad (4)$$

Finally, we define

$$W_{\text{avg},i} = \lim_{n \rightarrow +\infty} \frac{W_i(n)}{n} \quad (5)$$

as the measure of the communication delay experienced by user  $i$ 's tasks.



**Fig. 1** The delay models. (a) The communication delay; (b) The computing delay

### 1.2 Computing delay

First, we derive expressions describing the evolution of computing queues. There are  $J$  computing queues. For each  $j$ , let  $S_j(n)$  denote the workload of backlogged tasks, that is, the number of slots needed to execute all these tasks by server  $j$  at the end of slot  $n$ , which can be expressed as

$$S_j(n) = (S_j(n-1) + \sum_{i=1}^I \alpha_{ij} x_{ij}(n) N_i(n) - 1)^+ \quad (6)$$

where  $(\cdot)^+ = \max\{\cdot, 0\}$  and  $x_{ij}(n)$  is 1 if user  $i$ 's tasks are allocated to server  $j$  and 0 otherwise. There are two constraints on  $x_{ij}(n)$ . Firstly,  $x_{ij}(n)$  must be 0 or 1. Secondly, since a user can select one and only one server while a server can serve many users (i. e. , “many-to-one”), we have

$$\sum_{j=1}^J x_{ij}(n) = 1 \quad (7)$$

Next, we derive the formula of computing delay. Let  $Z_i(n)$  denote the normalized total computing delay experienced by user  $i$  until slot  $n$ , which can be written as

$$Z_i(n) = \sum_{j=1}^J Z_{ij}(n) \quad (8)$$

where  $Z_{ij}(n)$  is the normalized total computing delay experienced by user  $i$ 's tasks in server  $j$  until slot  $n$ . For this case, we use different methods to calculate the area between the arriving curve and leaving curve. Let  $D_{ijh}(n)$  denote the normalized delay (including execution time) experienced by the  $h$ -th task ( $1 \leq h \leq N_i(n)$ ). Then, we calculate the normalized area between the arriving curve and leaving curve as

$$Z_{ij}(n) = Z_{ij}(n-1) + \sum_{h=1}^{N_i(n)} D_{ijh}(n) \quad (9)$$

corresponding to the shadowed area in Fig. 1(b). Actually, task  $h$  experiences four types of delays: 1) It experiences delay  $S_j(n-1)^+$  which indicates that the task will not be instantly executed if the queue is not empty at the end of slot  $n-1$ ; 2) It experiences delay  $\sum_{l=1}^{i-1} \alpha_{lj} x_{lj}(n) N_l(n)$  which indicates that user  $i$ 's task has to wait the execution of user  $l$ 's tasks for  $l < i$ ; 3) It experiences delay  $(h-1)\alpha_{ij}$  which indicates that user  $i$ 's tasks are executed in sequence so that task  $h$  has to wait the execution of previous  $(h-1)$  tasks; 4) It experiences delay  $\alpha_{ij}$  which is the execution time of task  $h$  itself. Integrating these together, we obtain the expression of  $D_{ijh}(n)$  as

$$D_{ijh}(n) = (S_j(n-1) + \sum_{l=1}^{i-1} \alpha_{lj} x_{lj}(n) N_l(n) + (h-1)\alpha_{ij} + \alpha_{ij}) x_{ij}(n) \quad (10)$$

For convenience, let

$$\begin{aligned} D_{ij}(n) &= \sum_{h=1}^{N_i(n)} D_{ijh}(n) = \\ &= (S_j(n-1) + \sum_{l=1}^{i-1} \alpha_{lj} x_{lj}(n) N_l(n)) x_{ij}(n) N_i(n) + \\ &+ \alpha_{ij} x_{ij}(n) \frac{N_i^2(n) + N_i(n)}{2} \end{aligned} \quad (11)$$

Then, we can have the formula of  $Z_i(n)$  as

$$Z_i(n) = Z_i(n-1) + \sum_{j=1}^J D_{ij}(n) \quad (12)$$

Finally, we define

$$Z_{\text{avg},i} = \lim_{n \rightarrow +\infty} \frac{Z_i(n)}{n} \quad (13)$$

as the measure of the computing delay experienced by user  $i$ 's tasks.

## 2 Resource Scheduling

### 2.1 Problem formulation

Let  $W_{\text{avg}} = \sum_{1 \leq i \leq I} W_{\text{avg},i}/I$  and  $Z_{\text{avg}} = \sum_{1 \leq i \leq I} Z_{\text{avg},i}/I$  denote the average communication and computing delay for all users, respectively. Since communication queues are concatenated with computing queues, there are tradeoffs between  $W_{\text{avg}}$  and  $Z_{\text{avg}}$ . If  $W_{\text{avg}}$  is small,  $Z_{\text{avg}}$  will be large; otherwise, if  $W_{\text{avg}}$  is large, then  $Z_{\text{avg}}$  will be small. Therefore, we need to optimize these two delays jointly. To qualify the tradeoff, a weight factor  $b \geq 0$  is introduced to reflect the relative importance of these two delays. Therefore, the multiobjective optimization can be formulated as

$$\min_{\{N_i(n), \{x_{ij}(n)\}\}} W_{\text{avg}} + bZ_{\text{avg}} \quad (14)$$

subject to the constraints in Eqs. (2), (3), (7) and that  $N_i(n)$  are integers and  $x_{ij}(n)$  are binaries. This optimization problem must be transformed into a solvable form. Actually, this optimization problem consists of two subproblems. The first subproblem is  $\min W_{\text{avg}}$  and the second subproblem is  $\min Z_{\text{avg}}$ . The detailed forms of these two subproblems are derived as follows.

First, we consider the subproblem  $\min W_{\text{avg}}$ . As indicated in Eq. (4), minimizing  $W_{\text{avg},i}$  is equivalent to minimizing the average communication queue length. Due to the Lyapunov optimization framework established in Ref. [17], define the Lyapunov function  $L(n) = \sum_{1 \leq i \leq I} (U_i(n))^2$ . Then, the conditional Lyapunov drift is  $\Delta(n) = E\{L(n) - L(n-1) \mid U(n-1)\}$ , where  $U(n-1) = [U_1(n-1), \dots, U_I(n-1)]$  and  $E\{\cdot\}$  is the expectation operation. Exploiting that  $((a-b)^+ + c)^2 \leq a^2 + b^2 + c^2 + 2a(c-b)$  for any positive  $a, b$ , and  $c$ , we can have  $(U_i(n))^2 - (U_i(n-1))^2 \leq (A_i(n))^2 + (N_i(n))^2 + 2U_i(n-1)(A_i(n) - N_i(n))$ . According to the derivations in Ref. [17], we can estimate that  $\Delta(n) \leq B - 2E\{\sum_{1 \leq i \leq I} U_i(n-1)N_i(n) \mid U(n-1)\}$ , where  $B = \sum_{1 \leq i \leq I} (E\{(A_i(n))^2\} + (U_i(n-1))^2 + 2U_i(n-1)E\{A_i(n)\})$  is a constant. The Lyapunov-based algorithm is to make a transmission decision  $N_i(n)$  to minimize the right-hand-side of the drift bound. Note that the transmission decision on slot  $n$  only affects the final term on the right-hand-side. Thus, we need to design an algorithm minimi-

zing  $-E\left\{\sum_{1 \leq i \leq I} U_i(n-1)N_i(n) \mid U(n-1)\right\}$ . We now use the concept of opportunistically maximizing an expectation. That is, the above expression is maximized by the algorithm that observes the current queues  $U(n-1)$  and chooses  $N_i(n)$  to minimize  $-\sum_{1 \leq i \leq I} U_i(n-1)N_i(n)$ . Therefore, the subproblem  $\min W_{\text{avg}}$  can be transformed into the following optimization subproblem for each slot  $n$ :

$$\min_{\{N_i(n)\}} - \sum_{i=1}^I U_i(n-1)N_i(n) \quad (15)$$

subject to the constraints in Eqs. (2), (3) and that  $N_i(n)$  are integers, where  $U_i(n-1)$  is the communication queue length measured for the previous slot.

Secondly, we consider the subproblem  $\min Z_{\text{avg}}$ . In this case, a different method is taken. First, according to the definition of  $Z_{\text{avg}}$ , this subproblem is equivalent to  $\min - \sum_{1 \leq i \leq I} Z_{\text{avg},i}$ . Then, according to Eq. (13), this subproblem is equivalent to  $\min E\left\{-\sum_{1 \leq i \leq I} Z_i(n)\right\}$ . Substituting Eq. (12), this subproblem is equivalent to  $\min E\left\{-\sum_{1 \leq i \leq I} (Z_i(n-1) + - \sum_{1 \leq j \leq J} D_{ij}(n))\right\}$ . On slot  $n$ , since  $Z_i(n-1)$  has been given, the scheduling decision  $x_{ij}(n)$  only affects the term  $D_{ij}(n)$ . Therefore, we need to design an algorithm that minimizes  $E\left\{-\sum_{1 \leq i \leq I} - \sum_{1 \leq j \leq J} D_{ij}(n) \mid S(n-1)\right\}$ , where  $S(n-1) = [S_1(n-1), \dots, S_J(n-1)]$ . Using the concept of opportunistically maximizing an expectation, the above expression is minimized by the algorithm that observes the current queue  $S(n-1)$  and chooses  $x_{ij}(n)$  to minimize  $-\sum_{1 \leq i \leq I} - \sum_{1 \leq j \leq J} D_{ij}(n)$ . Therefore, the subproblem  $\min Z_{\text{avg}}$  can be transformed into the following optimization subproblem for each slot  $n$ :

$$\min_{\{x_{ij}(n)\}} \sum_{i=1}^I \sum_{j=1}^J D_{ij}(n) \quad (16)$$

subject to the constraints in Eq. (7) and that  $x_{ij}(n)$  are binaries, where  $S_j(n-1)$  in the expression of  $D_{ij}(n)$  is the computing queue length measured for the previous slot.

According to the above derivations, the subproblem  $\min W_{\text{avg}}$  and  $\min Z_{\text{avg}}$  can be transformed into the optimization problem in Eq. (15) and Eq. (16), respectively. Therefore, the problem in Eq. (14) can be transformed into the following optimization problem for each slot  $n$ :

$$\min_{\{N_i(n)\}, \{x_{ij}(n)\}} - \sum_{i=1}^I U_i(n-1)N_i(n) + b \sum_{i=1}^I \sum_{j=1}^J D_{ij}(n) \quad (17)$$

subject to the constraints in Eqs. (2), (3), (7) and that  $N_i(n)$  are integers and  $x_{ij}(n)$  are binaries.

## 2.2 Scheduling algorithm

Let  $g_{ij}$  denote the partial derivative of the objective function in Eq. (17) with respect to  $x_{ij}(n)$ , which can be calculated as

$$g_{ij}(N, \mathbf{x}) = bS_j(n-1)N_i(n) + \frac{b}{2}\alpha_{ij}(N_i^2(n) + N_i(n)) + b\left(\sum_{l=1}^{i-1} \alpha_{lj}x_{lj}(n)N_l(n)N_i(n) + \sum_{l=i+1}^I \alpha_{lj}x_{lj}(n)N_l(n)N_i(n)\right) \quad (18)$$

where  $N = [N_1(n), N_2(n), \dots, N_I(n)]$  and  $\mathbf{x} = [x_{11}(n), x_{12}(n), \dots, x_{IJ}(n)]$ , respectively. Let  $f_i$  denote the partial derivative of the objective function in Eq. (17) with respect to  $N_i(n)$ , which can be calculated as

$$f_i(N, U, \mathbf{x}) = -U_i(n-1) + b \sum_{j=1}^J S_j(n-1)x_{ij}(n) + b \sum_{j=1}^J \alpha_{ij}x_{ij}(n)\left(N_i(n) + \frac{1}{2}\right) + b \sum_{j=1}^J \left(\sum_{l=1}^{i-1} \alpha_{lj}x_{lj}(n)N_l(n)x_{ij}(n) + \sum_{l=i+1}^I \alpha_{lj}x_{lj}(n)N_l(n)x_{ij}(n)\right) \quad (19)$$

where  $U = [U_1(n-1), U_2(n-1), \dots, U_I(n-1)]$ . Additionally, at any  $(N, \mathbf{x})$ , if  $f_i \geq 0$ , user  $i$  does not offload any more task to any server; if  $N_i(n) = U_i(n-1)$ , user  $i$  does not offload any more task to any server; if  $R - \sum_{1 \leq i \leq I} R_i N_i(n) < R_i$ , user  $i$  does not offload any more task to any server. Therefore, we define the feasible user set at  $(\mathbf{x}, N)$  as

$$C(N, U, \mathbf{f}) = \{i: f_i < 0, N_i(n) < U_i(n-1), R - \sum_{i=1}^I R_i N_i(n) \geq R_i\} \quad (20)$$

where  $\mathbf{f} = [f_1, f_2, \dots, f_I]$ .

The proposed scheduling algorithm is as follows. First, initialize  $N_i(n) = 0$  for each  $i$  and  $U'_i(n-1) = U_i(n-1)$  for each  $i$ . The algorithm executes the following steps.

**Step 1** The values of  $\mathbf{x}$  are determined. Initially,  $x_{ij} = 0$  for each  $(i, j)$ . For each user  $i$ , calculate the gradient  $g_{ij}$  for each  $j$  by substituting  $N$  and  $\mathbf{x}$  into Eq. (18), then select the server  $j^* = \arg \min g_{ij}$  over all  $j$ , and assign user  $i$  to this server (i.e., set  $x_{ij^*} = 1$ ).

**Step 2** Update the values of  $N$ . First, calculate the values of  $\mathbf{f}(N, U, \mathbf{x})$ ; secondly, select the user  $i^* = \arg \min f_i$  over all feasible  $i \in C$ ; thirdly, increase  $N_{i^*}$  by one (i.e., let  $N_{i^*} \leftarrow N_{i^*} + 1$ ); fourthly, update  $U_{i^*}(n-1) = U_{i^*}(n-1) - 1$  and  $C(N, U, \mathbf{f})$ . If  $C$  is not empty, go back to Step 1; otherwise, the algorithm stops.

The computation complexity of the proposed algorithm is analyzed as follows. The proposed scheduling algorithm consists of two parts. For the first part, the gradient  $g_{ij}$  is calculated for each user  $i$  and each server  $j$ . Thus,

the computational complexity of this part is  $O(IJ)$ . For the second part, the gradient  $f_i$  is calculated for each user  $i$ . Thus, the computational complexity of this part is  $O(I)$ . Hence, the computational complexity of these two parts is  $O(IJ)$ . According to the proposed algorithm, these two parts are executed until set  $C$  becomes empty. According to the definition of  $C$  in Eq. (20), the times that  $C$  is not empty will be no more than  $\max_i(R/R_i)$ , which is a constant. Thus, the times that these two parts are executed is no more than a constant. Therefore, the overall computational complexity of the proposed algorithm is  $O(IJ)$ .

### 3 Simulation Results

Consider a time-slotted cellular communications system, in which the duration  $T_{\text{slot}}$  of each slot is 10 ms. Assume that the total number of subcarriers provided by the cell is  $R = 40$ . Assume that there are  $J = 2$  servers. Unless otherwise stated, the values of  $F_j$  are set to be  $2 \times 10^5 \times [1, 1.5]$ . For the users in the cell, the parameters are set as follows. Assume that there are  $I = 10$  users. For each user  $i$ , assume that his/her tasks arrive according to a Poisson process with an average inter-arrival time of  $T_i$ . Unless otherwise stated, the value of  $T_i$  is set to be 5 slots; the values of  $R_i$  are set to be  $[2, 2, 2, 2, 3, 3, 3, 3, 4, 4]$ ; and the value of  $E_i$  is set to be  $1.5 \times 10^3 \times (1 + i/10)$  for user  $i$ .

The main performance metric considered in this paper is the delay, which consists of two parts: the communication delay and computing delay. During the simulations, since it is not convenient to plot the delay of each user individually, we first obtain the average delays  $W_{\text{avg},i}$  and  $Z_{\text{avg},i}$  for each user  $i$ , then obtain the average delays  $W_{\text{avg}} = \sum_{i=1}^I W_{\text{avg},i}/I$  and  $Z_{\text{avg}} = \sum_{i=1}^I Z_{\text{avg},i}/I$  over all users, and then plot the values of  $W_{\text{avg}}$  and  $Z_{\text{avg}}$  as the final results for each parameter setting. The simulation results are presented as follows.

#### 3.1 Comparison with traditional algorithm

First, the traditional algorithm is briefly introduced as follows. Define the Lyapunov function as  $L(n) = \sum_{i=1}^I (U_i(n))^2 + b \sum_{j=1}^J (S_j(n))^2$ , where  $b$  is also a weight factor. The traditional algorithm is to make transmission decisions to minimize the conditional drift of the Lyapunov function<sup>[17]</sup>. Thus, the scheduling algorithm minimizes  $\sum_i U_i(n-1)N_i(n) + b \sum_i \sum_j \alpha_{ij} x_{ij}(n) \cdot N_i(n)S_j(n-1)$  for each  $n$  subject to the constraints in Eqs. (2), (3), (7) and  $N_i(n)$  are integers and  $x_{ij}(n)$  are binaries. We use the same algorithm in Section 3.2 to solve this problem, except that the gradients in Eqs. (18) and (19) are replaced by  $g_{ij}(N, \mathbf{x}) = b\alpha_{ij}N_i(n)S_j(n-1)$

$$\text{and } f_i(N, \mathbf{U}, \mathbf{x}) = -U_i(n-1) + b \sum_{j=1}^J \alpha_{ij} x_{ij}(n) S_j(n-1).$$

Fig. 2 shows the simulation results of  $W_{\text{avg}}$ ,  $Z_{\text{avg}}$ , and  $W_{\text{avg}} + Z_{\text{avg}}$  with different  $b$  for the proposed algorithm (i. e., Alg1) and the traditional algorithm (i. e., Alg2). For the curves in Fig. 2, we have the following observations. First, both algorithms can trade-off between the communication delay and computing delay. With the increase in  $b$ , the computing delay decreases, while the communication delay increases. For example, when  $b$  is increased from  $10^{-2}$  to  $10^1$ , the computing delay of the proposed algorithm decreases from 13.638 6 to 0.570 8 slots, while the communication delay of the proposed algorithm increases from 0.502 8 to 32.674 5 slots, and the total delay increases from 14.141 4 to 33.245 3 slots. Secondly, the total delay of the proposed algorithm (i. e., the line with circle mark) is always better than that of the traditional algorithm (i. e., the line with cross mark). For example, when  $b = 10^{-1}$ , the total delay of the traditional algorithm is 15.291 0 slot, while the total delay of the proposed algorithm is 13.426 6 slot, with a 12% off. This is due to the fact that the proposed algorithm directly minimizes the delay, while the traditional algorithm does not. Therefore, we can conclude that the delay performance of the proposed scheduling algorithm is better than that of the traditional algorithm.

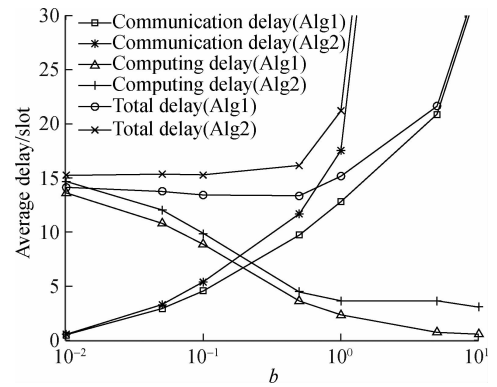
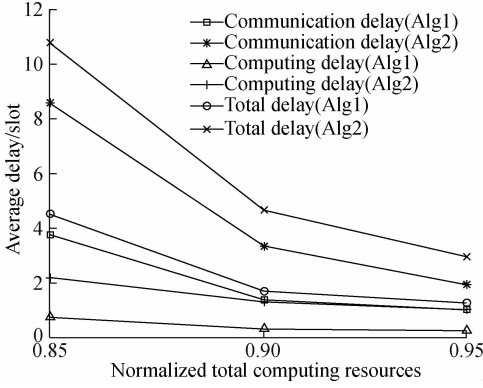


Fig. 2 Impact of the parameter  $b$  and comparison with traditional scheduling algorithm

#### 3.2 Impact of the amount of computing resources provided by the computing server

Fig. 3 shows the simulation results of  $W_{\text{avg}}$ ,  $Z_{\text{avg}}$ , and  $W_{\text{avg}} + Z_{\text{avg}}$  for different total computing resources. Specifically, in this figure, the horizontal axis represents the value of  $F'_j/F_j$ , where  $F_j$  is the baseline value of the total number of cycles provided by server  $j$  in one second, while  $F'_j$  is the actual value used in simulations. In this experiment, we set  $F'_j/F_j < 1$  and  $b = 1$ . From the curves in Fig. 3, we can observe that, with the decrease in the total computing resources, both the communication and computing delays increase. For example, for the proposed algorithm, when  $F'_j$  decreases from  $0.95 F_j$  to

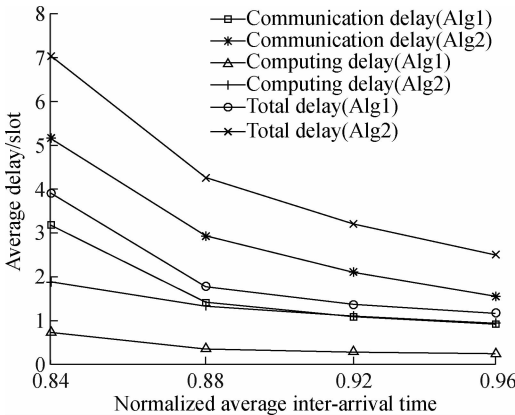
$0.85F_j$ , the communication delay increases from 1.0148 to 3.7693 slots, the computing delay increases from 0.2541 to 0.7445 slots, and the total delay increases from 1.2689 to 4.5138 slots. Furthermore, we can observe that, the total delay of the proposed algorithm (i.e., the blue line with circle mark) is always less than that of the traditional algorithm (i.e., the red line with circle mark). This is due to the fact that the proposed algorithm directly minimizes the delay, while the traditional algorithm does not.



**Fig. 3** Impact of the amount of computing resources provided by the computing server

### 3.3 Impact of the average task arrival rate of users

Fig. 4 shows the simulation results of  $W_{avg}$ ,  $Z_{avg}$ , and  $W_{avg} + Z_{avg}$  for different task arrival rates. Specifically, in this figure, the horizontal axis represents the value of  $T'_i/T_i$ , where  $T'_i$  is the baseline value of the average inter-arrival time of user  $i$ , while  $T_i$  is the actual value used in simulations. In this experiment, we set  $T'_i/T_i < 1$  and  $b = 1$ . From the curves in Fig. 4, we can observe that, with the decrease in the average inter-arrival time, both the communication and computing delays increase. For example, for the proposed algorithm, with the decrease of  $T'_i$  from  $0.96T_i$  to  $0.84T_i$ , the communication delay increases from 0.9204 to 3.1795 slots, the computing delay increases from 0.2468 to 0.7319 slots, and the total delay increases from 1.1672 to 3.9114 slots. Furthermore, we can observe that, the total delay of the propo-



**Fig. 4** Impact of the average task arrival rate of users

sed algorithm (i.e., the blue line with circle mark) is always better than that of the traditional algorithm (i.e., the red line with circle mark). This is due to the fact that the proposed algorithm directly minimizes the delay, while the traditional algorithm does not.

## 4 Conclusions

1) The analytical formulae of the communication and computing queueing delays in the many-to-one multi-server cellular edge computing systems are derived without assuming the knowledge of the statistics of user task arrival traffic.

2) A novel resource scheduling algorithm which solves the delay optimization problem directly is proposed for the many-to-one multi-server cellular edge computing systems.

3) Under the considered simulation parameters, the proposed algorithm can achieve 12% less total delay, as compared to the traditional queue length optimization based algorithm. System parameters including the weight, the amount of computing resources provided by servers, and the average user task arrival rate have impact on the percentage of delay reduction.

## References

- [1] Checko A, Christiansen H L, Yan Y, et al. Cloud RAN for mobile networks: A technology overview [J]. *IEEE Communications Surveys & Tutorials*, 2015, **17**(1): 405 – 426. DOI:10.1109/comst.2014.2355255.
- [2] Peng M G, Sun Y H, Li X L, et al. Recent advances in cloud radio access networks: System architectures, key techniques, and open issues [J]. *IEEE Communications Surveys & Tutorials*, 2016, **18**(3): 2282 – 2308. DOI: 10.1109/comst.2016.2548658.
- [3] Tran T X, Hajisami A, Pandey P, et al. Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges [J]. *IEEE Communications Magazine*, 2017, **55**(4): 54 – 61. DOI: 10.1109/mcom.2017.1600863.
- [4] Guo H Z, Liu J J, Zhang J. Computation offloading for multi-access mobile edge computing in ultra-dense networks [J]. *IEEE Communications Magazine*, 2018, **56**(8): 14 – 19. DOI:10.1109/mcom.2018.1701069.
- [5] Dinh T Q, La Q D, Quek T Q S, et al. Learning for computation offloading in mobile edge computing [J]. *IEEE Transactions on Communications*, 2018, **66**(12): 6353 – 6367. DOI:10.1109/tcomm.2018.2866572.
- [6] Wang F, Xu J, Wang X, et al. Joint offloading and computing optimization in wireless powered mobile-edge computing systems [J]. *IEEE Transactions on Wireless Communications*, 2018, **17**(3): 1784 – 1797. DOI: 10.1109/TWC.2017.2785305.
- [7] Zhou J Z, Zhang X, Wang W B. Joint resource allocation and user association for heterogeneous services in multi-access edge computing networks [J]. *IEEE Access*, 2019, **7**: 12272 – 12282. DOI:10.1109/access.2019.2892466.

[8] Du J B, Zhao L Q, Chu X L, et al. Enabling low-latency applications in LTE-A based mixed fog/cloud computing systems[J]. *IEEE Transactions on Vehicular Technology*, 2019, **68**(2): 1757 – 1771. DOI:10.1109/tvt.2018.2882991.

[9] Chen L X, Zhou S, Xu J. Computation peer offloading for energy-constrained mobile edge computing in small-cell networks[J]. *ACM Transactions on Networking*, 2018, **26**(4): 1619 – 1632. DOI:10.1109/tnet.2018.2841758.

[10] Ko S W, Han K F, Huang K B. Wireless networks for mobile edge computing: Spatial modeling and latency analysis[J]. *IEEE Transactions on Wireless Communications*, 2018, **17**(8): 5225 – 5240. DOI:10.1109/twc.2018.2840120.

[11] Fan Q, Ansari N. Application aware workload allocation for edge computing-based IoT[J]. *IEEE Internet of Things Journal*, 2018, **5**(3): 2146 – 2153. DOI:10.1109/ijot.2018.2826006.

[12] Yang L C, Zhang H L, Li M, et al. Mobile edge computing empowered energy efficient task offloading in 5G[J]. *IEEE Transactions on Vehicular Technology*, 2018, **67**(7): 6398 – 6409. DOI:10.1109/tvt.2018.2799620.

[13] Lyu X C, Ni W, Tian H, et al. Optimal schedule of mobile edge computing for Internet of Things using partial information[J]. *IEEE Journal on Selected Areas in Communications*, 2017, **35**(11): 2606 – 2615. DOI:10.1109/jsac.2017.2760186.

[14] Kim Y, Kwak J, Chong S. Dual-side optimization for cost-delay tradeoff in mobile edge computing[J]. *IEEE Transactions on Vehicular Technology*, 2018, **67**(2): 1765 – 1781. DOI:10.1109/tvt.2017.2762423.

[15] Feng H, Llorca J, Tulino A M, et al. Optimal control of wireless computing networks[J]. *IEEE Transactions on Wireless Communications*, 2018, **17**(12): 8283 – 8298. DOI:10.1109/twc.2018.2875740.

[16] Kim Y, Lee H W, Chong S. Mobile computation offloading for application throughput fairness and energy efficiency[J]. *IEEE Transactions on Wireless Communications*, 2019, **18**(1): 3 – 19. DOI:10.1109/twc.2018.2868679.

[17] Neely M J. Stochastic network optimization with application to communication and queueing systems[M]// *Synthesis Lectures on Communication Networks*. Morgan & Claypool, 2010: 1 – 211. DOI:10.2200/s00271ed1v01y201006cnt007.

# 多对一多服务器蜂窝边缘计算延时优化资源调度

杜 鹏<sup>1</sup> 巴特尔<sup>2</sup> 张 源<sup>2</sup>

(<sup>1</sup>南京邮电大学自动化学院、人工智能学院,南京 210023)  
(<sup>2</sup>东南大学移动通信国家重点实验室,南京 210096)

**摘要:**为了进一步降低蜂窝边缘计算系统中的延时,提出一种新的方法设计资源调度算法.在不需计算任务到达流量统计特性假设的情况下,以到达曲线和离开曲线为工具,推导了多对一多服务器蜂窝边缘计算系统中的通信与计算延时的解析表达式.以该延时表达式为基础,直接形成最小化延时的优化问题,并设计了相应的新型资源调度算法.对所提出的调度算法的延时性能进行了仿真评估.在所采用的仿真条件下,所提出调度算法的总延时比传统调度算法减少了12%.权重、服务器计算资源数量、用户任务平均达到速率等参数是影响延时降低百分比的重要因素.因此,与基于队列长度优化的传统调度算法相比,所提出的基于延时优化的调度算法可以进一步降低延时.

**关键词:**蜂窝系统; 延时; 边缘计算; 资源调度

**中图分类号:** TN929.5