Journal of Southeast University (English Edition) Vol. 35, No. 4, pp. 440 − 446 Dec. 2019 ISSN 1003—7985

A virtual delay queue-based backpressure scheduling for multi-cell cellular edge computing systems

Du Peng[1] Zhang Yuan[2]

([1]College of Automation & College of Artificial Intelligence, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)
([2]National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China)

Abstract: To further improve delay performance in multi-cell cellular edge computing systems, a new delay-driven joint communication and computing resource BP (backpressure) scheduling algorithm is proposed. Firstly, the mathematical models of the communication delay and computing delay in multi-cell cellular edge computing systems are established and expressed as virtual delay queues. Then, based on the virtual delay models, a novel joint wireless subcarrier and virtual machine resource scheduling algorithm is proposed to stabilize the virtual delay queues in the framework of the BP scheduling principle. Finally, the delay performance of the proposed virtual queue-based BP scheduling algorithm is evaluated via simulation experiments and compared with the traditional queue length-based BP scheduling algorithm. Results show that under the considered simulation parameters, the total delay of the proposed BP scheduling algorithm is always lower than that of the traditional queue length-based BP scheduling algorithm. The percentage of the reduced total delay can be as high as 51.29% when the computing resources are heterogeneously configured. Therefore, compared with the traditional queue length-based BP scheduling algorithms, the proposed virtual delay queue-based BP scheduling algorithm can further reduce delay in multi-cell cellular edge computing systems.

Key words: multi-cell cellular systems; edge computing; backpressure scheduling; delay

DOI: 10.3969/j.issn.1003 − 7985.2019.04.006

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, in which computing servers are located in the base stations (BSs) and users' applications are offloaded to BSs to execute[1-2]. There are two types of resources to be scheduled: the communication resource used during the offloading and the computing resource used during the execution. Hence, there are two types of delay, that is, communication delay and computing delay, in cellular edge computing systems. This paper studies the resource scheduling problem for cellular edge computing systems which minimizes the communication delay and computing delay.

There are many related studies in the literature. According to the assumptions of delay, they can be classified into three categories. For the first category[3-7], only the transmission delay or execution delay is considered. However, queueing related delay is not considered. The second category of the scheduling algorithm is based on the queuing theory model. For this category[8-12], in addition to transmission delay and execution delay, the queueing related delay is also considered. However, this category assumes that the queue can be modelled as the traditional queue (e.g., the M/M/1 queue) so that the delay formulas of the queueing theory can be readily applied. The third category of the scheduling algorithm is based on the backpressure (BP) scheduling framework. For this category, the queueing related delay is also considered. Furthermore, this category does not make any assumption of the model of the queue and uses the framework of BP scheduling which was established in Refs. [13 − 14] to design scheduling algorithms. A typical resource scheduling algorithm of this category is derived as follows[13-19]: Firstly, consider that the average delay and average queue length are interchangeable by Little's Law; secondly, define the queue length-based Lyapunov function; finally, design the backpressure scheduling algorithm which minimizes the conditional drift of the Lyapunov function by the method in Refs. [13 − 14].

In this work, we focus on the backpressure-based scheduling algorithms. Existing backpressure-based scheduling algorithms have two limitations. Firstly, existing BP scheduling algorithms were all based on the queue length which did not control the delay directly. Actually, due to the lack of delay formulas, the existing backpressure-based scheduling algorithms cannot solve the delay control problem directly but have to solve the queue length control problem instead. Secondly, most existing backpressure-based scheduling algorithms only considered the

---

Received 2019-05-30, **Revised** 2019-10-12.

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

Foundation items: The National Natural Science Foundation of China (No. 61571111), the Incubation Project of the National Natural Science Foundation of China at Nanjing University of Posts and Telecommunications (No. NY219106).

Citation: Du Peng, Zhang Yuan. A virtual delay queue-based backpressure scheduling for multi-cell cellular edge computing systems[J]. Journal of Southeast University (English Edition), 2019, 35(4): 440 − 446. DOI: 10.3969/j.issn.1003 − 7985.2019.04.006.

single-cell scenario. For this scenario, the scheduler only needs to decide how to allocate communication resources between users. However, the multi-cell scenario. is more practical than the single-cell scenario. In fact, for the multi-cell scenario, in addition to deciding how to allocate communication resources between users, the scheduler still needs to decide how to assign BSs to users. Therefore, this work will extend the traditional backpressure-based scheduling algorithms by studying the scheduling algorithm which solves the delay control problem directly for multi-cell cellular edge computing systems.

1 System Models

Consider a multi-cell cellular edge computing system, in which time is slotted and the duration of each slot is T (unit: s). Let I denote the total number of users.

For the communication resource, assume that there are a total of R subcarriers and J BSs in the system. At the beginning of each slot, the communication scheduler allocates subcarriers and BS for the applications of users. For each subcarrier allocation, let $s_{i,e}$ and $d_{i,e}$ denote the number of subcarriers and the index of BS allocated to the e-th application of user i, respectively. Let $m_{i,j}(\cdot)$ denote the subcarrier-slot mapping function which maps the number of allocated subcarriers to the number of slots needed to offload an application from user i to BS j. Assume that the subcarrier-slot mapping function $m_{i,j}(\cdot)$ is a monotonic decreasing function of the variable. For BS allocation, let Ψ_i denote the set of BSs accessible by user i (i.e., BS j can receive the signal from user i with a signal-to-interference-plus-noise-ratio exceeding a threshold), then we have $d_{i,e} \in \Psi_i$.

For the computing resource, assume that there are a total of F_j virtual machines (VMs) in BS j. At the beginning of each slot, the computing scheduler of each BS j allocates VMs for the applications offloaded to itself. Let $g_{j,i,e}$ denote the number of VMs allocated to the e-th application of user i by BS j where $d_{i,e} = j$. Assuming that at most $g_{\max}$ VMs can be allocated to an application, then we have $g_{j,i,e} \leqslant g_{\max}$. Let $u_i(\cdot)$ denote the VM-slot mapping function which maps the number of allocated VMs to the number of slots needed to execute an application of user i, which is a monotonic decreasing function of the variable.

1.1 Communication delay model

There are I communication queues, one for each user. For the i-th communication queue, let $U_i(n-1)$ denote the number of applications in the queue at the beginning of slot n, $X_{i,j}(n)$ denote the number of applications offloaded to BS j (i.e., leaving the queue) during slot n, and $A_i(n)$ denote the number of applications arriving to the queue at the end of slot n. The value of $X_{i,j}(n)$ can be written as

$$X_{i,j}(n) = \sum_{e:d_{i,e}=j} 1\{n_{i,e} + m_{i,j}(s_{i,e}) - 1 = n\} \qquad (1)$$

where $1\{\cdot\}$ equals one if the condition is true and zero otherwise and $n_{i,e}$ represents the index of slot at the beginning of which the values of $s_{i,e}$ and $d_{i,e}$ are given. Hence, the evolution of the i-th communication queue is described as

$$U_i(n) = \Big(U_i(n-1) - \sum_{j\in\Psi_i} X_{i,j}(n)\Big)^+ + A_i(n) \qquad (2)$$

where $(\cdot)^+ = \max(\cdot, 0)$.

The constraints on the scheduling variables $(n_{i,e}, s_{i,e}, d_{i,e})$ are stated as follows. Let $r_{i,e}(n)$ denote the number of subcarriers allocated to the e-th application of user i during slot n, that is

$$r_{i,e}(n) = \begin{cases} s_{i,e} & n \in [n_{i,e}, n_{i,e} + m_{i,j}(s_{i,e}) - 1] \\ 0 & \text{else} \end{cases} \qquad (3)$$

Then, let $R_{i,j}(n)$ denote the total number of subcarriers allocated to the link from user i to BS j during slot n, that is

$$R_{i,j}(n) = \sum_{e:d_{i,e}=j} r_{i,e}(n) \qquad (4)$$

For any two users i and k, if there is a BS which is accessible to both i and k, we say that user k is a competitor user of user i. Let Ω_i denote the set of the competitor users of user i. Then for each user i and BS $j \in \Psi_i$, we have the following constraint:

$$\sum_{j\in\Psi_i} R_{i,j}(n) + \sum_{k\in\Omega_i} \sum_{h\in\Psi_k} R_{k,h}(n) \leqslant R \qquad (5)$$

Finally, the formula of communication delay is derived as follows. Let $\Gamma_{i,\text{tot}}(n)$ denote the total delay experienced by all applications in the i-th communication queue until slot $n+1$. Then, we have

$$\Gamma_{i,\text{tot}}(n) = \sum_{k=1}^{n} U_i(k)T \qquad (6)$$

Let $\Gamma_i(n)$ denote the time-average of $\Gamma_{i,\text{tot}}(n)$, that is, $\Gamma_i(n) = \Gamma_{i,\text{tot}}(n)/n$, as the measure of the communication delay experienced by the applications of user i. Then, we can express $\Gamma_i(n)$ as a virtual queue:

$$\Gamma_i(n) = \Gamma_i(n-1) - \varepsilon\Gamma_i(n-1) + \varepsilon U_i(n)T \qquad (7)$$

where ε is the smoothing coefficient and takes the value of $1/n$. Since $1/n$ approaches zero with the increase in n, we can alternatively set ε to be a positive constant. Finally, let $W_i(n)$ denote the normalized version of $\Gamma_i(n)$, that is, $W_i(n) = \Gamma_i(n)/T$. Then, we have

$$W_i(n) = W_i(n-1) - \varepsilon W_i(n-1) + \varepsilon U_i(n) \qquad (8)$$

1.2 Computing delay model

For each BS j, there are I computing queues, one for each user. For the i-th computing queue in BS j, let $Q_{j,i}(n)$ denote the number of applications in the queue at the end of slot $n+1$, and $Y_{j,i}(n)$ denote the number of applications finishing the execution (i.e., leaving the queue) during slot $n+1$. The value of $Y_{j,i}(n)$ can be written as

$$Y_{j,i}(n) = \sum_{e:d_{i,e}=j} 1\{v_{j,i,e} + u_i(g_{j,i,e}) - 1 = n + 1\} \quad (9)$$

Recall that a total of $X_{i,j}(n)$ applications leave the i-th communication queue at the end of slot n, as described in Eq. (1). We assume that these applications enter the i-th computing queue in BS j at the end of slot $n+1$. Hence, the evolution of the i-th computing queue in BS j is described as

$$Q_{j,i}(n) = (Q_{j,i}(n-1) - Y_{j,i}(n))^+ + X_{i,j}(n) \quad (10)$$

The constraints on the scheduling variables ($v_{j,i,e}$, $g_{j,i,e}$) are stated as follows. Let $f_{j,i,e}(n)$ denote the number of VMs allocated to the e-th application of user i in BS j during slot $n+1$, that is

$$f_{j,i,e}(n) = \begin{cases} g_{j,i,e} & n+1 \in [v_{j,i,e}, v_{j,i,e} + u_i(g_{j,i,e}) - 1] \\ 0 & \text{else} \end{cases} \quad (11)$$

Then, let $F_{j,i}(n)$ denote the total number of VMs allocated to user i in BS j during slot $n+1$, that is

$$F_{j,i}(n) = \sum_{e:d_{i,e}=j} f_{j,i,e}(n) \quad (12)$$

Then for each BS j and user i, we have the constraint:

$$\sum_{i=1}^{I} F_{j,i}(n) \leqslant F_j \quad (13)$$

Finally, the formula of computing delay is derived as follows. Let $\Xi_{j,i,\mathrm{tot}}(n)$ denote the total delay experienced by all applications in the i-th computing queue in BS j until slot $n+1$. Then, we have

$$\Xi_{j,i,\mathrm{tot}}(n) = \sum_{k=1}^{n} Q_{j,i}(k)T \quad (14)$$

Let $\Xi_{j,i}(n)$ denote the time-average of $\Xi_{j,i,\mathrm{tot}}(n)$, that is, $\Xi_{j,i}(n) = \Xi_{j,i,\mathrm{tot}}(n)/n$, as the measure of the computing delay experienced by the applications of user i in BS j. Then, we can express $\Xi_{j,i}(n)$ as a virtual queue:

$$\Xi_{j,i}(n) = \Xi_{j,i}(n-1) - \varepsilon\Xi_{j,i}(n-1) + \varepsilon Q_{j,i}(n)T \quad (15)$$

Let $Z_{j,i}(n)$ denote the normalized version of $\Xi_{j,i}(n)$, that is, $Z_{j,i}(n) = \Xi_{j,i}(n)/T$. Then, we have

$$Z_{j,i}(n) = Z_{j,i}(n-1) - \varepsilon Z_{j,i}(n-1) + \varepsilon Q_{j,i}(n) \quad (16)$$

2 Virtual Delay Queue-Based Backpressure Scheduling

2.1 The proposed scheduling algorithm

Define the Lyapunov function as

$$L(n) = \sum_{i=1}^{I} W_i(n)^2 + \sum_{i=1}^{I}\sum_{j=1}^{J} Z_{j,i}(n)^2 \quad (17)$$

According to the Lyapunov optimization technique, the conditional Lyapunov drift can be calculated as $\Delta(n) = E\{L(n) - L(n-1) \mid \boldsymbol{W}(n-1), \boldsymbol{Z}(n-1)\}$, where $E\{\cdot\}$ is the expectation operation, $\boldsymbol{W}(n-1) = [W_1(n-1), \ldots, W_I(n-1)]$, and $\boldsymbol{Z}(n-1) = [Z_{1,1}(n-1), \ldots, Z_{J,I}(n-1)]$. Substituting Eq. (17) into the above equation, we have

$$\Delta(n) = E\Big\{\sum_{1\leqslant i\leqslant I} \varepsilon_n^2 W_i(n-1)^2 + \sum_{1\leqslant i\leqslant I}\sum_{1\leqslant j\leqslant J} \varepsilon_n^2 Z_{j,i}(n-1)^2 + \sum_{1\leqslant i\leqslant I} \varepsilon_n^2 U_i(n)^2 + \sum_{1\leqslant i\leqslant I}\sum_{1\leqslant j\leqslant J} \varepsilon_n^2 Q_{j,i}(n)^2 - \sum_{1\leqslant i\leqslant I} 2\varepsilon_n W_i(n-1)^2 - \sum_{1\leqslant i\leqslant I}\sum_{1\leqslant j\leqslant J} 2\varepsilon_n Z_{j,i}(n-1)^2 + \sum_{1\leqslant i\leqslant I} 2\varepsilon_n(1-\varepsilon_n) W_i(n-1) U_i(n) + \sum_{1\leqslant i\leqslant I}\sum_{1\leqslant j\leqslant J} 2\varepsilon_n(1-\varepsilon_n) Z_{j,i}(n-1) Q_{j,i}(n) \,\Big|\, \boldsymbol{W}(n-1), \boldsymbol{Z}(n-1)\Big\}$$

where the first six terms can be upper bounded by a constant under the expectation operation. Using the concept of opportunistically maximizing an expectation, the above expression is minimized by the algorithm that obtains the values of $\boldsymbol{W}(n-1)$ and $\boldsymbol{Z}(n-1)$ and minimize $\sum_{1\leqslant i\leqslant I} W_i(n-1) U_i(n) + \sum_{1\leqslant i\leqslant I}\sum_{1\leqslant j\leqslant J} Z_{j,i}(n-1) Q_{j,i}(n)$. Furthermore, substituting Eqs. (2) and (10) into the above equation, the objective can be upper bounded as

$$\sum_{1\leqslant i\leqslant I} W_i(n-1) U_i(n-1) + \sum_{1\leqslant i\leqslant I} W_i(n-1) A_i(n) + \sum_{1\leqslant i\leqslant I}\sum_{1\leqslant j\leqslant J} Z_{j,i}(n-1) Q_{j,i}(n-1) - \sum_{1\leqslant i\leqslant I}\sum_{1\leqslant j\leqslant J} W_i(n-1) X_{i,j}(n) + \sum_{1\leqslant i\leqslant I}\sum_{1\leqslant j\leqslant J} Z_{j,i}(n-1) X_{i,j}(n) - \sum_{1\leqslant i\leqslant I}\sum_{1\leqslant j\leqslant J} Z_{j,i}(n-1) Y_{j,i}(n)$$

where the first three terms can also be upper bounded by a constant. Therefore, the final form of the optimization problem for each n is

$$\min \sum_{i=1}^{I}\sum_{j=1}^{J} (Z_{j,i}(n-1) - W_i(n-1)) X_{i,j}(n) - \sum_{i=1}^{I}\sum_{j=1}^{J} Z_{j,i}(n-1) Y_{j,i}(n) \quad (18)$$

where $X_{i,j}(n)$ and $Y_{j,i}(n)$ are determined by scheduling variables ($n_{i,e}$, $s_{i,e}$, $d_{i,e}$) and ($v_{j,i,e}$, $g_{j,i,e}$) which must satisfy the constraints in Eqs. (3) to (5) and (11) to (13). Note that $X_{i,j}(n)$ and $Y_{j,i}(n)$ are decoupled in both the objective and constraints in (18), and therefore can be optimized separately as follows.

The subproblem of optimizing communication resource scheduling variables can be written as

$$\min \sum_{i=1}^{I}\sum_{j=1}^{J} (Z_{j,i}(n-1) - W_i(n-1)) X_{i,j}(n) \quad (19)$$

where $X_{i,j}(n)$ is determined by scheduling variables ($n_{i,e}$, $s_{i,e}$, $d_{i,e}$) which must satisfy the constraints in Eqs. (3) to (5). This is a typical backpressure scheduling problem[13-14]. The proposed communication resource scheduling algorithm is presented as follows. We initially set $\mathrm{tmpW}_i = (1-\varepsilon) W_i(n-1) + \varepsilon U_i(n-1)$ and $\mathrm{tmpZ}_{j,i} = (1-\varepsilon) Z_{j,i}(n-1) + \varepsilon Q_{j,i}(n-1)$ for each i and j. For each application e of user i satisfying $d_{i,e} = j$ and $n_{i,e} \leqslant n \leqslant n_{i,e} + m_{i,j}(s_{i,e}) - 1$, update $R_i \leftarrow R_i - s_{i,e}$ and $R_k \leftarrow R_k - s_{i,e}$ for each $k \in \Omega_i$. Let L_i contain the index of applications of user

i which has not been allocated any subcarrier yet. For each n, the algorithm performs the following steps.

1) Determine the pair $(i^*, j^*) = \arg\min(\mathrm{tmpZ}_{j,i} - \mathrm{tmpW}_i)$ over all feasible user i's (i. e., $R_i > 0$ and L_i is not empty) and BS $j \in \Psi_i$. If there is no feasible user, the algorithm stops.

2) If there is an $e^* \in L_{i*}(n)$ with $d_{i*,e*} = j^*$ and $s_{i*,e*} > 0$, then update $s_{i*,e*} \leftarrow s_{i*,e*} + 1$; otherwise, pick the application which arrives the earliest and denoted as e^*, then set $n_{i*,e*} = n$, $s_{i*,e*} = 1$, and $d_{i*,e*} = j^*$. Update $R_{i*} \leftarrow R_{i*} - 1$ and $R_k \leftarrow R_k - 1$ for each $k \in \Omega_{i*}$. If $m_{i*,j*}(s_{i*,e*}) = 1$, remove this e^* from $L_{i*}(n)$.

3) For each j and i, update $\mathrm{tmpW}_i = (1-\varepsilon)W_i(n-1) + \varepsilon(U_i(n-1) + \sum_e \gamma_{i,e})$ and $\mathrm{tmpZ}_{j,i} = (1-\varepsilon)Z_{j,i}(n-1) + \varepsilon(Q_{j,i}(n-1) + \sum_e \delta_{i,e})$, where if $n_{i,e} = 0$, $\gamma_{i,e} = 1$ and $\delta_{i,e} = 0$; if $n_{i,e} = n$, $\gamma_{i,e} = 1 - 1/m_{i,j}(s_{i,e})$ and $\delta_{i,e} = 1$. Go back to step 1).

The subproblem of optimizing computing resource scheduling variables can be written as

$$\max \sum_{i=1}^{I} \sum_{j=1}^{J} Z_{j,i}(n-1)\, Y_{j,i}(n) \tag{20}$$

where $Y_{j,i}(n)$ is determined by scheduling variables $(v_{j,i,e}, g_{j,i,e})$ which must satisfy the constraints in Eqs. (11) to (13). This is a typical max-weight scheduling problem. The proposed computing resource scheduling algorithm is presented as follows. We initially set $\mathrm{tmpZ}_{j,i} = (1-\varepsilon)Z_{j,i}(n-1) + \varepsilon Q_{j,i}(n-1)$ for each j and i. For each application e of user i satisfying $d_{i,e} = j$ and $v_{j,i,e} \leqslant n + 1 \leqslant v_{j,i,e} + u_i(g_{j,i,e}) - 1$, update $F_{j,i} \leftarrow F_{j,i} - g_{j,i,e}$ and $F_{j,k} \leftarrow F_{j,k} - g_{j,i,e}$ for each $k \neq i$. Let $P_{j,i}$ contain the index of applications of user i in BS j which have not been allocated any VM yet. For each BS j on each n, the algorithm performs the following steps.

1) Determine user $i^* = \arg\max \mathrm{tmpZ}_{j,i}$ over all feasible user i's(i. e., $F_{j,i} > 0$ and $P_{j,i}$ is not empty). If there is no feasible user, the algorithm stops.

2) If there is an $e^* \in P_{j,i*}$ with $g_{j,i*,e*} > 0$, then update $g_{j,i*,e*} \leftarrow g_{j,i*,e*} + 1$; otherwise, pick the application arriving earliest as e^*, then set $v_{j,i*,e*} = n + 1$ and $g_{j,i*,e*} = 1$. Update $F_{j,i*} \leftarrow F_{j,i*} - 1$ and $F_{j,k} \leftarrow F_{j,k} - 1$ for each $k \neq i^*$. If $g_{j,i*,e*} = g_{\max}$, remove this e^* from $P_{j,i*}$.

3) Update $\mathrm{tmpZ}_{j,i} = (1-\varepsilon)Z_{j,i}(n-1) + \varepsilon(Q_{j,i}(n-1) + \sum_e \lambda_{i,e})$ for each i, where $\lambda_{i,e} = 1$ if $v_{j,i,e} = 0$, or $\lambda_{i,e} = 1 - 1/u_i(g_{j,i,e})$ if $v_{j,i,e} = n + 1$. Go back to Step 1.

The computation complexity of the proposed algorithm is analyzed as follows. The proposed scheduling algorithm consists of two parts. For the first part, the value of $(\mathrm{tmpZ}_{j,i} - \mathrm{tmpW}_i)$ is calculated for each user i and each BS j. Thus, the computational complexity of this calculation is $O(IJ)$. Furthermore, according to the algorithm, such calculation is executed until there is no feasible user. Since the number of subcarriers is limited, the number of times of calculating the value of $(\mathrm{tmpZ}_{j,i} - \mathrm{tmpW}_i)$ will be no more than a constant. Therefore, the computational complexity of the first part of the proposed algorithm is $O(IJ)$. For the second part, for each BS j, the value of $\mathrm{tmpZ}_{j,i}$ is calculated for each user i. Thus, the computational complexity of this calculation is $O(I)$. Furthermore, according to the algorithm, such calculation is executed until there is no feasible user. Since the number of VMs in each BS is limited, the number of times of calculating the value of $\mathrm{tmpZ}_{j,i}$ will be no more than a constant. Therefore, the computational complexity of the second part of the proposed algorithm is $O(I)$. The overall computational complexity of the proposed algorithm is $O(IJ)$.

2.2 Traditional queue length-based backpressure scheduling

To benchmark the performance of the proposed scheduling algorithm, the traditional queue length-based scheduling algorithm is introduced as follows. According to Refs. [13 – 19], a typical traditional queue length-based backpressure scheduling is derived as follows: Firstly, define the queue length-based Lyapunov function; then, design the backpressure scheduling algorithm which minimizes the conditional drift of the Lyapunov function.

Therefore, for the considered multi-cell cellular edge computing system, we firstly define the Lyapunov function as

$$L[n] = \sum_{i=1}^{I} U_i[n]^2 + \sum_{j=1}^{J} \sum_{j=1}^{J} Q_{j,i}[n]^2 \tag{21}$$

Then we design the backpressure scheduling algorithm which minimizes the conditional drift of the Lyapunov function as follows. The conditional Lyapunov drift is $\Delta(n) = \mathrm{E}\{L(n) - L(n-1) \mid \boldsymbol{U}(n-1), \boldsymbol{Q}(n-1)\}$, where $\boldsymbol{U}(n-1) = [U_1(n-1), \ldots, U_I(n-1)]$, and $\boldsymbol{Q}(n-1) = [Q_{1,1}(n-1), \ldots, Q_{J,I}(n-1)]$. After similar derivations[14], the following optimization problem should be solved for each n:

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} (Q_{j,i}[n-1] - U_i[n-1]) X_{i,j}[n] - \sum_{i=1}^{I} \sum_{j=1}^{J} Q_{j,i}[n-1]\, Y_{j,i}[n] \tag{22}$$

where $X_{i,j}(n)$ and $Y_{j,i}(n)$ are determined by scheduling variables $(n_{i,e}, s_{i,e}, d_{i,e})$ and $(v_{j,i,e}, g_{j,i,e})$ which must satisfy the constraints in Eqs. (3) to (5) and (11) to (13). Note that $X_{i,j}(n)$ and $Y_{j,i}(n)$ are also decoupled in (22). Therefore, the optimization problem in (22) can also be decomposed into two subproblems, where the communication resource scheduling subproblem is

$$\min \sum_{i=1}^{I} \sum_{j=1}^{J} (Q_{j,i}[n-1] - U_i[n-1]) X_{i,j}[n] \tag{23}$$

and the computing resource scheduling subproblem is

$$\max \sum_{i=1}^{I} \sum_{j=1}^{J} Q_{j,i}[n-1] Y_{j,i}[n] \tag{24}$$

These optimization subproblems and those in (19) and (20) are very similar. Therefore, according to Refs. [13 − 19], we use the same procedure to solve these optimization subproblems except that the value of delay is replaced by the value of queue length. Finally, the computational complexity of the traditional algorithm is analyzed as follows. The traditional scheduling algorithm also consists of two parts. For the first part, the core step is to calculate the value of queue length difference for each user i and each BS j. Thus, the computational complexity of the first part of the traditional algorithm is $O(IJ)$. For the second part, for each BS j, the core step is to calculate the value of queue length for each user i. Thus, the computational complexity of the second part of the traditional algorithm is $O(I)$. The overall computational complexity of the traditional queue length-based backpressure scheduling algorithm is also $O(IJ)$. Therefore, the proposed delay-based scheduling algorithm and the traditional queue length-based scheduling algorithm have the same computational complexity.

3 Performance Evaluation

Consider a time-slotted multi-cell cellular edge computing system. Assume that there are $J = 3$ BSs. As illustrated in Fig. 1, the geographical positions of these BSs are: (0, 1), (0.866, −0.5), and (−0.866, −0.5) in kilometer. A total of $I = 20$ users are uniformly distributed over the area covered by BSs. For each run, we generate a snapshot of the distribution of the users. For each user i, assume that his/her applications arrive according to a Poisson process with the average inter-arrival time of T_i (unit: s). Unless otherwise stated, the value of T_i is set to be 5 slots. Let d_{ij} represent the distance between user i and BS j. If $d_{ij} < 1.5B$, user i is considered to be a neighbor of BS j. Then sets Ψ_i and Ω_i can be determined for each user i. For any user i and BS j, the function $m_{i,j}(\cdot)$ is set as follows. If $d_{ij} \leqslant 0.2$ km, then set the function as $m_{i,j}(1) = 2$ and $m_{i,j}(2) = 1$, as shown in Fig. 2(a); if $0.2\text{ km} < d_{ij} < 0.6$ km, then set the function as $m_{i,j}(1) = 3$, $m_{i,j}(2) = 2$ and $m_{i,j}(3) = 1$, as shown in Fig. 2(b); if $d_{ij} \geqslant 0.6$ km, then set the function as $m_{i,j}(1) = 4$, $m_{i,j}(2) = 3$, $m_{i,j}(3) = 2$ and $m_{i,j}(4) = 1$, as shown in Fig. 2(c). The value of $g_{\max}$ is set to be 3, and the function $u_i(g)$ is set as $u_i(1) = 6$, $u_i(2) = 2$, and $u_i(3) = 1$. The value of R will be changed for different simulations. The value of F_j is set as follows. For odd-numbered j, we set $F_j = 10$; for even-numbered j, we set $F_j = \rho_j \times 10$ where the value of ρ_j will be changed for different simulations.

The main performance metric considered in this paper is delay, which consists of two parts: the communication delay and the computing delay. For each parameter setting, we firstly collect the communication delay and computing delay experienced by each application of each user, then average these values over all applications and all users, and finally calculate the average total delay as the summation of the average communication delay and average computing delay. For each parameter setting, we run the simulation experiment independently one hundred times and average the results. Additionally, we consider a total of four different scheduling algorithms: The first is the proposed delay-based scheduling algorithm (i. e., (19) and (20)), abbreviated as the Dly algorithm; the second is the joint delay-based communication resource scheduling (i. e., (19)) and the queue length-based computing resource scheduling (i. e., (24)), abbreviated as the DlyQue algorithm; the third is the joint queue length-based communication resource scheduling (i. e., (23)) and the delay-based computing resource scheduling (i. e., (20)), abbreviated as the QueDly algorithm; and the final is the traditional queue length-based scheduling (i. e., (23) and (24)), abbreviated as the Que algorithm.

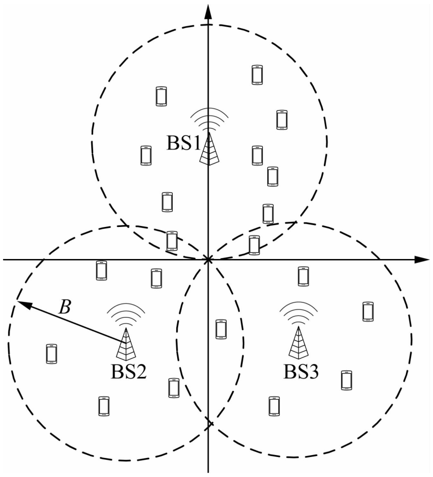
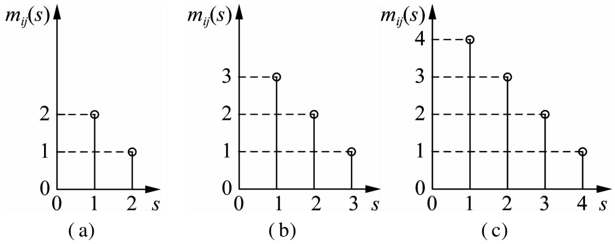


Fig. 1 The simulated multi-cell cellular edge computing system



(a) (b) (c)

Fig. 2 The function $m_{i,j}(s)$ used in the simulation. (a) The distance is less than or equal to 0.2 km; (b) The distance is between 0.2 and 0.6 km; (c) The distance is greater than or equal to 0.6 km

Fig. 3 shows the simulation results of the normalized value of the total delay (i. e., measured in slot) with different R (i. e., the total number of subcarriers in the system) for different resource scheduling algorithms. In this experiment, we set $F_j = 5$ for even-numbered j and change R from 10 to 20. According to the results in the figure, we can observe that the delay of the proposed algorithm (i. e., the Dly algorithm) is always better than that of the other three queue length-based algorithms. For example, when $R=16$, the delay of the Que, DlyQue, and QueDly algorithm is 3.5, 3.0, and 2.7 slots, respectively, while the delay of the proposed Dly algorithm is 2.5 slots, with a 28%, 17%, and 7% off, respectively. This is due to the fact that the proposed Dly algorithm controls delay directly, while the other queue length-based algorithms do not. Therefore, we can conclude that the delay performance of the proposed delay-based algorithm is better than that of the queue length-based algorithm.

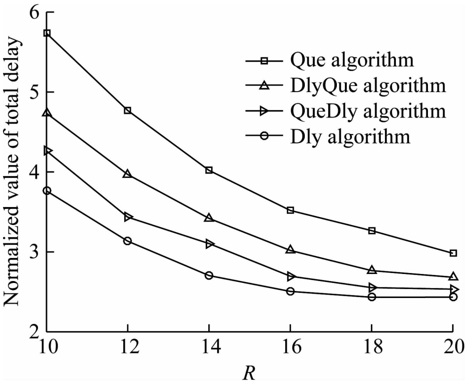


Fig. 3 Impact of the total number of subcarriers on the delay performance

Fig. 4 shows the simulation results of the normalized value of the total delay for different values of the total number of VMs in even-numbered BSs. In this experiment, we set $R=16$ and decrease the value of F_j from 10 to 5 for even-numbered j. According to the curves in the figure, we can observe that, with the decrease of F_j for even-numbered j, the heterogeneity of BSs increases, then the difference between the proposed Dly algorithm and the other three queue length-based algorithm increases. For example, when F_j for even-numbered j is 4 (i. e., the computing resource configuration is very heterogeneous), the delay of the Que, DlyQue, and QueDly algorithms is 5.8, 5.0, and 3.2 slots, respectively, while the delay of the proposed Dly algorithm is 2.6 slots, with a 55%, 48%, and 18% improvement, respectively. Therefore, we can conclude that the proposed Dly algorithm is more capable of evenly distributing traffic load across BSs with different computing capabilities than the queue length-based algorithm.

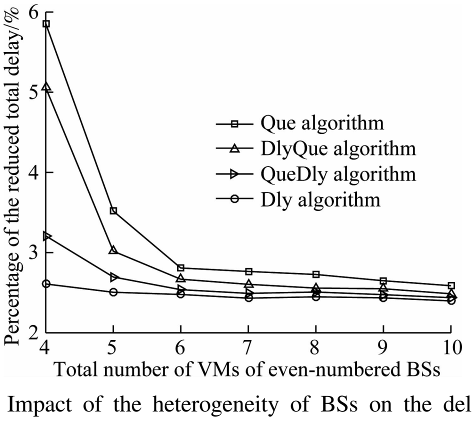


Fig. 4 Impact of the heterogeneity of BSs on the delay performance

Fig. 5 shows the simulation results of the percentage of the reduced total delay (i. e., $(1-d_{\mathrm{Dly}}/d_{\mathrm{Que}})\times 100\%$, where d_{Dly} and d_{Que} represent the total delay of the proposed Dly algorithm and traditional Que algorithm, respectively.) for different values of the smoothing coefficient ε. In this experiment, we set $R=16$ and $F_j=5$ for even-number j and change ε from 10^{-3} to 1. According to the curves in the figure, we can observe that, with the increase in ε, the difference between the proposed Dly algorithm and the traditional Que algorithm decreases. For example, when $\varepsilon=10^{-3}$, the percentage of the reduced total delay is 27%; when $\varepsilon=1$, the percentage of the reduced total delay is 2%. Therefore, we can conclude that the traditional Que algorithm can be considered to be a special case of the proposed Dly algorithm when the smoothing coefficient $\varepsilon=1$.

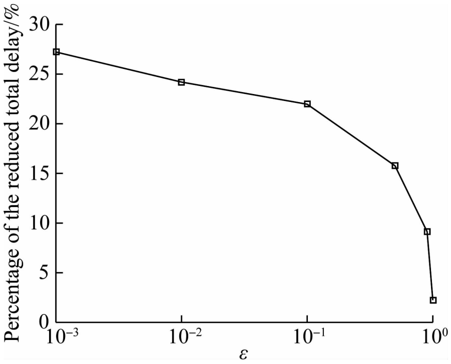


Fig. 5 Impact of the smoothing coefficient on the delay performance

4 Conclusions

1) The analytical formulas of the communication delay and computing delay in multi-cell cellular edge computing systems were derived and expressed as virtual queues.

2) A novel backpressure resource scheduling algorithm was proposed to stabilize the virtual delay queues.

3) Simulation results show that the delay performance of the proposed algorithm is better than that of the queue length-based traditional algorithm, and the proposed algorithm is more capable of evenly distributing traffic load across BSs than the traditional one.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] Wu Y, Ni K J, Zhang C, et al. NOMA-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation [J]. *IEEE Transactions on Vehicular Technology*, 2018, **67**(12): 12244－12258. DOI: 10.1109/tvt.2018.2875337.

[6] 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.

[7] 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.

[8] Guo S S, Wu D L, Zhang H X, et al. Resource modeling and scheduling for mobile edge computing: A service provider's perspective[J]. *IEEE Access*, 2018, **6**: 35611－35623. DOI: 10.1109/access.2018.2851392.

[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.

[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/jiot.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] Tassiulas L, Ephremides A. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks[J]. *IEEE Transactions on Automatic Control*, 1992, **37**(12): 1936－1948. DOI: 10.1109/9.182479.

[14] Neely M J. Stochastic network optimization with application to communication and queueing systems[J]. *Synthesis Lectures on Communication Networks*, 2010, **3**(1): 1－211. DOI: 10.2200/s00271ed1v01y201006cnt007.

[15] Lü X, 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.

[16] 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.

[17] 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.

[18] Du J B, Yu F R, Chu X L, et al. Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization[J]. *IEEE Transactions on Vehicular Technology*, 2019, **68**(2): 1079－1092. DOI: 10.1109/tvt.2018.2883156.

[19] 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.

多小区蜂窝边缘计算中基于虚拟延时队列的 BP 调度

杜鹏[1]　张源[2]

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

摘要：为了进一步改善多小区蜂窝边缘计算系统中的延时性能，提出了一种新的基于延时驱动的联合通信与计算资源 BP（backpressure）调度算法．首先，为多小区蜂窝边缘计算系统中的通信与计算延时建立数学模型并表达为虚拟延时队列．然后，基于该虚拟延时队列模型，以 BP 调度算法为框架，以稳定虚拟延时队列为优化目标，设计了一种新型的联合无线子载波与计算虚拟机资源调度算法．最后，对所提出基于虚拟延时队列的 BP 调度算法的延时性能进行了仿真评估，并与传统基于队列长度的 BP 调度算法进行了对比．结果表明，在所采用的仿真条件下，所提出 BP 调度算法的总延时总是低于传统基于队列长度的 BP 调度算法．特别地，当计算资源异构配置时，总延时减少的百分比可以达到 51.29%．因此，与传统基于队列长度的 BP 调度算法相比，所提出的基于虚拟延时队列的 BP 调度算法可以进一步降低延时．

关键词：多小区蜂窝系统；边缘计算；backpressure 调度；延时

中图分类号：TN929.5