Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 1324897, 10 pages
https://doi.org/10.1155/2018/1324897

Research Article

Dynamic Service Request Scheduling for Mobile Edge Computing Systems

Ying Chen, Yongchao Zhang, and Xin Chen

Computer School, Beijing Information Science and Technology University (BISTU), Beijing 100101, China

Correspondence should be addressed to Ying Chen; chenying@bistu.edu.cn

Received 20 April 2018; Accepted 3 July 2018; Published 13 September 2018

Academic Editor: Kok-Seng Wong

Nowadays, mobile services (applications) running on terminal devices are becoming more and more computation-intensive. Offloading the service requests from terminal devices to cloud computing can be a good solution, but it would put a high burden on the network. Edge computing is an emerging technology to solve this problem, which places servers at the edge of the network. Dynamic scheduling of offloaded service requests in mobile edge computing systems is a key issue. It faces challenges due to the dynamic nature and uncertainty of service request patterns. In this article, we propose a Dynamic Service Request Scheduling (DSRS) algorithm, which makes request scheduling decisions to optimize scheduling cost while providing performance guarantees. The DSRS algorithm can be implemented in an online and distributed way. We present mathematical analysis which shows that the DSRS algorithm can achieve arbitrary tradeoff between scheduling cost and performance. Experiments are also carried out to show the effectiveness of the DSRS algorithm.

1. Introduction

With the rapid development of Information Technology and increasing promotion of terminal devices [1], the mobile services (applications) running on terminal devices are becoming more and more complex and computation-intensive [2, 3]. However, the computing capacity and battery life of terminal devices are generally limited, and these devices cannot afford to process all these service requests locally on devices. To solve this problem, some researches propose to offload the service requests from terminal devices to cloud computing, which has more computing resources and larger capacity [4–8]. Nevertheless, cloud computing is usually located remotely that is far away from terminal devices. Besides, with the increasing popularity of mobile services running on terminal devices, scheduling all the offloaded service requests to cloud computing can put a significant burden on the networks [9, 10]. To meet this challenge, recent researches have been proposed to put edge servers with computing capacities at the edge of the networks in close proximity to terminal devices. Mobile Edge Computing (MEC) is an emerging technology based on this idea [11–14], and it has drawn extensive attention from both academy and industry [15–18].

One of the key issues in MEC research is how to schedule service requests [2, 19, 20]: when a large number of service requests are offloaded, how to schedule service requests among multiple MEC systems in order to reduce scheduling cost while providing performance guarantees. It is intuitive that there exist tradeoffs between scheduling cost and performance. Besides, the service request scheduling problem among multiple MEC systems is challenging due to several reasons. Firstly, as terminal devices are moving and the service environment varies over time [21, 22], how to make dynamic request scheduling decisions in accordance with the uncertainty of request patterns and changing environment is a great challenge [23]. Secondly, with the increasing promotion of terminal devices and mobile services, both the number of terminal devices and mobile services are rising dramatically, making the service request scheduling problem more complicated.

Some existing researches have studied the service request scheduling problem in MEC systems. Reference [24] modelled the server in the MEC as one $M/M/1$ queue. Reference

[2] assumed the offloaded service requests arrived at the MEC system according to a Poisson process. These works assumed the request arrival followed certain distribution. However, in reality, the request arrival process is highly dynamic, and the statistical information of request arrival can hardly be obtained or precisely predicted [25, 26]. Besides, with the number of terminal devices and mobile services increasing, traditional centralized optimization techniques such as combination optimization and dynamic programming may suffer from high-complexity and result in long execution time.

In this article, we introduce a dynamic online service request scheduling mechanism which requires no prior information of the statistical information of request arrivals. Specifically, the request scheduling among multiple MEC systems is formulated as an optimization problem, and the goal is to minimize request scheduling cost while providing performance guarantees. Based on Lyapunov optimization techniques, we propose a Dynamic Service Request Scheduling (DSRS) algorithm. DSRS uses a parameter V to control the tradeoff between scheduling cost and queue length. Mathematical analysis is presented which proves that DSRS is $O(1/V)$-optimal with respect to the average scheduling cost while still bounding the average queue length by $O(V)$. Experiments are also conducted which demonstrate that DSRS can make dynamic control decisions to adjust to variable environments and achieve the tradeoff between scheduling cost and queue length.

The remainder of this article is organized as follows. In Section 2, we present the system model for dynamic request scheduling among multiple MEC systems and formulate the optimization problem. In Section 3, based on Lyapunov optimization techniques, we propose an online and Dynamic Service Request Scheduling algorithm. Theoretical analysis of the scheduling algorithm is presented in Section 4. Experiments are conducted to evaluate the efficiency and effectiveness of the scheduling algorithm in Section 5. We conclude this article in Section 6.

2. System Model

2.1. Overview. Consider n mobile edge computing (MEC) systems. Each MEC system has an edge server virtualized to m virtual machines to process offloaded requests of m types of services from the terminal devices [25, 27]. More specifically, the i-th virtual machine on each edge server in the MEC system serves the offloaded requests for the i-th type of service. Let I be the collection of indexes for applications and J be the collection of indexes for edge server in the MEC systems. Without loss of generality, the edge servers in different MEC systems are supposed to be heterogeneous. We consider a time-slotted model and the length of time slot is denoted by τ. The main notations in this section are listed in Table 1.

2.2. Problem Formulation

2.2.1. Service Request Scheduling. In each time slot $t \in \{0, 1, \ldots, T-1\}$, a number of service requests for the m types of services are offloaded. Let $A_i(t)$ be the number of requests for service i offloaded to the MEC systems in time slot t. In our article, we require no prior knowledge of the statistics of $A_i(t)$, which is generally hard to obtain or precisely predict in real-life. $a_{ij}(t)$ represents the number of requests for service i that are scheduled to the edge server in the j-th MEC system in time slot t. $a_{ij}(t)$ is the request scheduling control variable. It should be satisfied that

$$\sum_{j \in J} a_{ij}(t) = A_i(t), \quad \forall i \in I. \tag{1}$$

The request scheduling method in our article will make use of the diversity of different MEC systems to provide service in order to reduce scheduling cost while providing performance guarantees.

2.2.2. Scheduling Cost. Let $\gamma_{ij}(t)$ be the unit cost of scheduling requests for service i to the j-th MEC system. $\gamma_{ij}(t)$ can be different among different services i and different MEC systems j. It can also vary across time for other factors such as traffic, wireless fading, the available resources, etc. The request scheduling cost of service i in time slot t can be calculated as $\sum_{j \in J} \gamma_{ij}(t) a_{ij}(t)$. The total scheduling cost for all the services can be expressed as

$$g(t) = \sum_{i \in I} \sum_{j \in J} \gamma_{ij}(t)\, a_{ij}(t). \tag{2}$$

Instead of studying the instantaneous scheduling cost, we focus on the long-term average cost. The time-average scheduling cost across time slots $t \in \{0, 1, \ldots, T-1\}$ can be expressed as

$$g = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{E}\{g(t)\}. \tag{3}$$

g is the minimization objective of the request scheduling problem in this article.

2.2.3. Performance. Queueing delay is one of the most important performance metrics. According to *Little's Law*, queueing delay is in proportion to the number of requests waiting in the queue. Thus, we seek to reduce queue length and maintain low congestion states. Let $Q_{ij}(t)$ represent the queue length of service i in the j-th MEC system in time slot t. $b_{ij}(t)$ denotes the number of requests for service i that can be served by the j-th MEC system. Thus, the queue length $Q_{ij}(t)$ evolves as

$$Q_{ij}(t+1) = \max\left[Q_{ij}(t) - b_{ij}(t), 0\right] + a_{ij}(t). \tag{4}$$

To reduce queueing delay and maintain system stability, we seek to bound the average queue length. Let the time-average queue length across the $t \in \{0, 1, \ldots, T-1\}$ slots represented by q_{ij}. The service request scheduling method in this article bounds the average queue length as

$$q_{ij} = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{E}\left\{Q_{ij}(t)\right\} < \zeta, \quad \exists \zeta \in \mathbb{R}^+. \tag{5}$$

Table 1: Notations and definitions.

Notation	Definition
I	Services set.
J	MEC systems set.
$A_i(t)$	Number of requests for service i in time slot t.
$a_{ij}(t)$	Number of requests for service i that are scheduled to the j-th MEC system in time slot t.
$b_{ij}(t)$	Number of requests for service i that can be served by the j-th MEC system in time slot t.
$\gamma_{ij}(t)$	Unit cost of scheduling requests for service i to the j-th MEC system.
$Q_{ij}(t)$	Queue length of service i on the j-th MEC system in time slot t.
$g(t)$	Scheduling cost for all the services in time slot t.

2.2.4. Unified Framework. To combine scheduling cost and performance, the request scheduling problem in this article is formulated as

$$\text{minimize} \quad g = \lim_{T\longrightarrow\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{E}\left\{g\left(t\right)\right\}; \tag{6}$$

subject to constraints (1), (5).

Solving problem (6) offline requires the future information (such as requests arrival information, scheduling cost information) which is generally hard to obtain or precisely predict in practice. Thus, we propose an online Dynamic Service Request Scheduling algorithm to solve the problem, which will be shown in Section 3.

3. Dynamic Request Scheduling Algorithm Design

In this section, based on the Lyapunov optimization framework [28], we decompose the original optimization problem into a series of independent subproblems. Then, we design a Dynamic Service Request Scheduling algorithm to solve these subproblems in a distribute way.

3.1. Problem Transformation Using Lyapunov techniques. Based on Lyanuov optimization techniques, we define $\Theta(t) = (Q_{ij}(t))$ as the queue length matrix of the MEC systems. Then, we denote $L(\Theta(t))$ as the Lyanuov function as follows, which is a scalar measure of the queue congestion state in the system,

$$L\left(\Theta\left(t\right)\right) = \frac{1}{2} \sum_{i\in I} \sum_{j\in J} Q_{ij}^2\left(t\right). \tag{7}$$

A small value of $L(\Theta(t))$ indicates that the queue lengths of all MECs are small, which represents a low congestion state of the MEC systems according to the *Little's Law*. In order to reduce queue length and maintain system stability, we seek to keep the Lyanuov function at a small value. Then, we define the *conditional Lyapunov drift* $\Delta(\Theta(t))$,

$$\Delta\left(\Theta\left(t\right)\right) = \mathbf{E}\left\{L\left(\Theta\left(t+1\right)\right) - L\left(\Theta\left(t\right)\right) \mid \Theta\left(t\right)\right\}. \tag{8}$$

By reducing the value of $\Delta(\Theta(t))$, we can push the Lyanuov function towards to a small value. To integrate scheduling cost and queue length in the MEC systems, we define the *drift plus cost* according to Lyapunov optimization framework, which is expressed as

$$\Delta\left(\Theta\left(t\right)\right) + V\mathbf{E}\left\{g\left(t\right) \mid \Theta\left(t\right)\right\}. \tag{9}$$

The parameter V can be considered as the tradeoff parameter between the scheduling cost and queue length, which can be determined by service providers or users according to their requirements in real applications. Next in Theorem 1, we show that the *drift plus cost* is upper bounded if the service arrival rate can be upper bounded.

Theorem 1 (bounding drift plus cost). *In each time slot t, under any algorithm, for all possible values of $\Theta(t)$ and any parameter value of V, if there exits a peak value A_i^{max} that upper bounds the number of requests arrived in each time slot, the drift plus cost can be upper bounded by*

$$\begin{aligned}
&\Delta\left(\Theta\left(t\right)\right) + V\mathbf{E}\left\{g\left(t\right) \mid \Theta\left(t\right)\right\} \\
&\quad\le B + \sum_{i\in I} \sum_{j\in J} Q_{ij}\left(t\right) \mathbf{E}\left\{a_{ij}\left(t\right) - b_{ij}\left(t\right) \mid \Theta\left(t\right)\right\} \\
&\qquad + V \sum_{i\in I} \sum_{j\in J} \mathbf{E}\left\{a_{ij}\left(t\right)\gamma_{ij}\left(t\right) \mid \Theta\left(t\right)\right\},
\end{aligned} \tag{10}$$

where $B = (1/2)[\sum_{i\in I}(A_i^{max})^2 + \sum_{i\in I}\sum_{j\in J}\widehat{b}_{ij}^2]$ is a constant.

Proof. By squaring the both sides of (4) and applying the inequality that $(\max[Q_{ij}(t) - b_{ij}(t), 0])^2 \le (Q_{ij}(t) - b_{ij}(t))^2$, we have

$$\begin{aligned}
Q_{ij}^2\left(t+1\right) \le{}& \left(Q_{ij}\left(t\right) - b_{ij}\left(t\right)\right)^2 + a_{ij}^2\left(t\right) \\
&+ 2a_{ij}\left(t\right) \max\left[Q_{ij}\left(t\right) - b_{ij}\left(t\right), 0\right].
\end{aligned} \tag{11}$$

Then, we define $\overline{b}_{ij}(t)$ as the actual number of requests for service i served by the j-th MEC system in time slot t,

$$\overline{b}_{ij}\left(t\right) = \begin{cases} b_{ij}\left(t\right), & b_{ij}\left(t\right) \le Q_{ij}\left(t\right) \\ Q_{ij}\left(t\right), & otherwise. \end{cases} \tag{12}$$

We can obtain that $\max[Q_{ij}(t) - b_{ij}(t), 0] = Q_{ij}(t) - \overline{b}_{ij}(t)$ and rewrite (11) as follows:

$$\begin{aligned} Q_{ij}^2(t+1) \le{}& Q_{ij}^2(t) + a_{ij}^2(t) + b_{ij}^2(t) \\ &+ 2Q_{ij}(t)\left(a_{ij}(t) - b_{ij}(t)\right) \\ &- 2a_{ij}(t)\,\overline{b}_{ij}(t). \end{aligned} \tag{13}$$

Because $a_{ij}(t)\overline{b}_{ij}(t) \ge 0$, we have

$$\begin{aligned} &\frac{1}{2}\left[Q_{ij}^2(t+1) - Q_{ij}^2(t)\right] \\ &\quad\le \frac{1}{2}\left[a_{ij}^2(t) + b_{ij}^2(t)\right] + Q_{ij}(t)\left[a_{ij}(t) - b_{ij}(t)\right]. \end{aligned} \tag{14}$$

Taking the expectations on the condition of $\Theta(t)$ to both sides in (14) and summing over $i \in I$ and $j \in J$, it can be obtained that

$$\begin{aligned} \Delta(\Theta(t)) \le{}& \frac{1}{2}\sum_{i\in I}\sum_{j\in J}\mathbf{E}\left\{a_{ij}^2(t) + b_{ij}^2(t) \mid \Theta(t)\right\} \\ &+ \sum_{i\in I}\sum_{j\in J}Q_{ij}(t)\,\mathbf{E}\left\{a_{ij}(t) - b_{ij}(t) \mid \Theta(t)\right\}. \end{aligned} \tag{15}$$

Since it holds that $\sum_{j\in J} a_{ij}(t) = A_i(t)$ and $A_i(t) \le A_i^{max}$, we have

$$\sum_{j\in J}\mathbf{E}\left\{a_{ij}^2(t) \mid \Theta(t)\right\} \le \mathbf{E}\left\{A_i^2(t) \mid \Theta(t)\right\} \le \left(A_i^{max}\right)^2. \tag{16}$$

In addition, we define $\widehat{b}_{ij}$ as the upper bound of $b_{ij}(t)$ over all the time slots. We can obtain that

$$\begin{aligned} &\sum_{i\in I}\sum_{j\in J}\mathbf{E}\left\{a_{ij}^2(t) + b_{ij}^2(t) \mid \Theta(t)\right\} \\ &\quad\le \sum_{i\in I}\left(A_i^{max}\right)^2 + \sum_{i\in I}\sum_{j\in J}\widehat{b}_{ij}^2. \end{aligned} \tag{17}$$

By adding $V\mathbf{E}\{g(t) \mid \Theta(t)\}$ to both sides and letting B take the value of $(1/2)[\sum_{i\in I}(A_i^{max})^2] + \sum_{i\in I}\sum_{j\in J}\widehat{b}_{ij}^2$, it can be obtained that

$$\begin{aligned} &\Delta(\Theta(t)) + V\mathbf{E}\left\{g(t) \mid \Theta(t)\right\} \\ &\quad\le B + V\mathbf{E}\left\{g(t) \mid \Theta(t)\right\} \\ &\qquad+ \sum_{i\in I}\sum_{j\in J}Q_{ij}(t)\,\mathbf{E}\left\{a_{ij}(t) - b_{ij}(t) \mid \Theta(t)\right\}. \end{aligned} \tag{18}$$

Substituting (2) into the right-hand-side (R.H.S.) of (18), we can obtain (10). □

3.2. Dynamic Request Scheduling Algorithm. Following the design principles of Lyapunov optimization techniques, we design an efficient Dynamic Service Request Scheduling (DSRS) algorithm to minimize the upper bound of *drift plus cost* in each time slot t. By decomposing the minimization of upper bound problem into a series of independent subproblems, our DSRS algorithm optimizes the average scheduling cost concurrently in a distributed way. In addition, it will be proven that DSRS algorithm can achieve a long-term time-average scheduling cost that is arbitrarily close to the optimal value while maintaining the stability of the MEC systems.

In each time slot t, based on the current queue length matrix $\Theta(t)$ of the MEC systems, the DSRS algorithm makes request scheduling decisions $a_{ij}(t)$ to minimize the upper bound of R.H.S. of (10). Since B and b_{ij} can be considered as constant in the optimization problem, we can rewrite the minimization of upper bound as

$$\min_{a_{ij}(t)} \quad \sum_{i\in I}\sum_{j\in J}a_{ij}(t)\,Q_{ij}(t) + Va_{ij}(t)\,\gamma_{ij}(t). \tag{19}$$

subject to

$$\sum_{j\in J}a_{ij}(t) = A_i(t), \quad \forall i \in I. \tag{20}$$

As the request scheduling decisions $a_{ij}(t)$ are independent among different services, the above centralized minimization problem (19) can be decomposed into the following subproblem (21) for each service $i \in I$, i.e.,

$$\min_{a_{ij}(t)} \quad \sum_{j\in J}a_{ij}(t)\left(Q_{ij}(t) + V\gamma_{ij}(t)\right). \tag{21}$$

subject to

$$\sum_{j\in J}a_{ij}(t) = A_i(t). \tag{22}$$

Problem (21) can be regarded as a generalized min-weight problem, where the number of requests scheduled to the MEC systems is weighted by the value of $Q_{ij}(t) + V\gamma_{ij}(t)$. Therefore, for each service $i \in I$, the optimal solution is to schedule all the requests to the MEC system with the minimum value of $Q_{ij}(t) + V\gamma_{ij}(t)$; i.e.,

$$a_{ij}(t) = \begin{cases} A_i(t), & j = j^* \\ 0, & otherwise \end{cases} \tag{23}$$

where $j^* \in \operatorname{argmin}(Q_{ij}(t) + V\gamma_{ij}(t))$ for all $j \in J$.

Remark. There exist tradeoffs between the scheduling cost and queue length of the MEC systems. Scheduling all the service requests to the MEC system with low cost can reduce the overall scheduling cost; however, the queue length of the MEC system can be very large. The DSRS algorithm combines scheduling cost and queue length, and $Q_{ij}(t) + V\gamma_{ij}(t)$ can be regarded as the penalty factor for each MEC system. Recall that V represents the tradeoff between scheduling cost and queue length. The intuition of the optimal scheduling policy obtained by the DSRS algorithm is to minimize the penalty function of the MEC systems in each time slot. In this way, the DSRS algorithm can reduce both the scheduling cost and the queue length. In addition, by changing the value of V, the

1: In the beginning of each time slot t, observe the current queue length $Q_{ij}(t)$.
2: **for all** $i \in I$ **do**
3: Set auxiliary variable $MinWeight = -\infty$;
4: Set $j^* = 1$;
5: **for all** $j \in J$ **do**
6: Calculate the penalty factor $pf_j = Q_{ij}(t) + V\gamma_{ij}(t)$;
7: **if** $pf_j < MinWeight$ **then**
8: $j^* = j$;
9: **end if**
10: **end for**
11: **for all** $j \in J$ **do**
12: **if** $j == j^*$ **then**
13: Set $a_{ij}(t) = A_i(t)$;
14: **else**
15: Set $a_{ij}(t) = 0$;
16: **end if**
17: **end for**
18: **end for**

Algorithm 1: Dynamic Service Request Scheduling (DSRS).

DSRS algorithm can achieve the arbitrary tradeoff between scheduling cost and queue length.

After the scheduling decisions $a_{ij}(t)$ are determined, the queue length $Q_{ij}(t)$ updates according to (4). The detailed algorithm is shown in Algorithm 1.

4. Algorithm Analysis

In this section, we present mathematical analysis of the boundary of the time-average queue length and scheduling cost of our DSRS algorithm. It can be proven that our algorithm can achieve the scheduling cost arbitrarily close to the optimal value while maintaining the stability of the MEC systems. Let $\overline{Q}$ denote the long-term time-average queue length,

$$\overline{Q} = \lim_{T \longrightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i \in I} \sum_{j \in J} \mathbf{E}\left\{Q_{ij}(t)\right\}. \tag{24}$$

We present in Lemma 2 that if the arrival $A_i(t)$ is independent and identically distributed (i.i.d.) over time slots, there exists a randomized policy π^* which can achieve the minimum cost g^* defined in (3), where the control decision $a_{ij}(t)$ follows certain fixed probability distribution independent of the queue length matrix $\Theta(t)$.

Lemma 2. *For any service request arrival rate $\lambda \in \Lambda$, where Λ is the capacity region of the system, if the arrival $A_i(t)$ is i.i.d. over time slots, there exists a randomized policy π^* that determines the control decision $a_{ij}(t)$ in each time slot t and achieves the following:*

$$\begin{aligned} &\mathbf{E}\left\{g^{\pi^*}(t)\right\} = g^*(\lambda); \\ &\mathbf{E}\left\{\sum_{j \in J} a_{ij}^{\pi^*}(t)\right\} \le \mathbf{E}\left\{\sum_{j \in J} b_{ij}(t)\right\}. \end{aligned} \tag{25}$$

where $g^*(\lambda)$ denotes the minimum time-average cost under the arrival rate λ.

Proof. Lemma 2 can be proven by Caratheodory's theorem in [28], we omit the detailed proof here for simplicity and brevity. □

Since it is assumed that there exists upper bound A_i^{max} of the service request arrival rate, there also exists upper bound $\hat{g}$ and lower bound $\check{g}$ of the objective g. Then, we derive the boundary of queue length and scheduling cost of the DSRS algorithm based on Lemma 2.

Theorem 3. *Assume that there exists ε satisfying $\lambda + \varepsilon \in \Lambda$, then, under our DSRS algorithm, for any value of the parameter V, the time-average queue length defined in (24) is bounded as*

$$\overline{Q} \le \frac{B + V(\hat{g} - \check{g})}{\varepsilon}. \tag{26}$$

Furthermore, the time-average system scheduling cost can be bounded by (27), which shows the cost derived by our DSRS algorithm can approach the optimal value by increasing the parameter V. Here, B is the constant defined in Theorem 1.

$$g^{DSRS} \le g^* + \frac{B}{V}. \tag{27}$$

Proof. Since it holds that $\lambda + \varepsilon \in \Lambda$, we can obtain that there exists a randomized policy π' which satisfies (28) and (29) according to Lemma 2.

$$\mathbf{E}\left\{g^{\pi'}(t)\right\} = g^*(\lambda + \varepsilon); \tag{28}$$

$$\mathbf{E}\left\{\sum_{j \in J} a_{ij}^{\pi'}(t)\right\} \le \mathbf{E}\left\{\sum_{j \in J} b_{ij}(t)\right\} - \varepsilon. \tag{29}$$

As our DSRS algorithm can achieve the minimum value of the R.H.S of (10) among all feasible policies (including policy π'), it can be obtained that

$$\begin{aligned}&\Delta(\Theta(t)) + V\mathbf{E}\{g(t) \mid \Theta(t)\}\\ &\le B + V\mathbf{E}\left\{g^{\pi'(t)} \mid \Theta(t)\right\}\\ &\quad + \sum_{i\in I}\sum_{j\in J} Q_{ij}(t)\,\mathbf{E}\left\{a_{ij}^{\pi'}(t) - b_{ij}(t) \mid \Theta(t)\right\}.\end{aligned} \tag{30}$$

Substituting (28) and (29) into the R.H.S. of (30), taking expectations on both sides, and then using iterated expectations, we can yield

$$\begin{aligned}&\mathbf{E}\{L(\Theta(t+1)) - L(\Theta(t))\} + V\mathbf{E}\{g(t)\}\\ &\le B + Vg^*(\lambda+\varepsilon) - \varepsilon\sum_{i\in I}\sum_{j\in J}\mathbf{E}\left\{Q_{ij}(t)\right\}.\end{aligned} \tag{31}$$

Moving $V\mathbf{E}\{g(t)\}$ to the R.H.S. of (31), it can be obtained that

$$\begin{aligned}&\mathbf{E}\{L(\Theta(t+1)) - L(\Theta(t))\}\\ &\le B + V\left(g^*(\lambda+\varepsilon) - \mathbf{E}\{g(t)\}\right)\\ &\quad - \varepsilon\sum_{i\in I}\sum_{j\in J}\mathbf{E}\left\{Q_{ij}(t)\right\}\\ &\le B + V\left(\widehat{g} - \check{g}\right) - \varepsilon\sum_{i\in I}\sum_{j\in J}\mathbf{E}\left\{Q_{ij}(t)\right\}.\end{aligned} \tag{32}$$

To be general, we assume the queue length is empty when $t = 0$. By summing both sides of (32) over $t \in \{0, 1, \ldots T-1\}$ and applying the fact that $L(\Theta(t)) \ge 0$, we can obtain

$$\begin{aligned}\varepsilon\sum_{t=0}^{T-1}\sum_{i\in I}\sum_{j\in J}\mathbf{E}\left\{Q_{ij}(t)\right\} &\le \left(B + V\left(\widehat{g} - \check{g}\right)\right)T\\ &\quad - \mathbf{E}\{L(\Theta(T))\}\\ &\le \left(B + V\left(\widehat{g} - \check{g}\right)\right)T.\end{aligned} \tag{33}$$

Dividing both sides of (33) by εT and taking a lim as $T \longrightarrow \infty$ yield (26).

By summing both sides of (31) over $t \in \{0, 1, \ldots T-1\}$ and applying the fact that $\mathbf{E}\{Q_{ij}(t)\} \ge 0$, it can be obtained

$$V\sum_{t=0}^{T-1}\mathbf{E}\{g(t)\} \le \left(Vg^*(\lambda+\varepsilon) + B\right)T. \tag{34}$$

Dividing both sides of (34) by VT, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{E}\{g(t)\} \le g^*(\lambda+\varepsilon) + \frac{B}{V}. \tag{35}$$

Taking a lim of (35) as $T \longrightarrow \infty$, applying Lebesgue's dominated convergence theorem, and letting $\varepsilon \longrightarrow 0$ yield (27). □

Remark. Theorem 3 shows that our DSRS algorithm can achieve a $[O(1/V), O(V)]$ tradeoff between the time-average scheduling cost and queue length. According to (27), the gap between the time-average scheduling cost obtained by our DSRS algorithm and the optimal value is within $O(1/V)$. By setting the value of V sufficiently large, the DSRS algorithm can approach the optimal scheduling cost. However, a large V will cause a large queue backlog of the MEC systems. Nevertheless, the queue length obtained by our DSRS algorithm is also bounded according to (26). And constraint (5) can be satisfied by letting ζ take the value of $(B + V(\widehat{g} - \check{g}))/\varepsilon$.

Then, we analyze the time complexity of the DSRS algorithm. According to Algorithm 1, for the two inner loops (line 5-10 and line 11-17), DSRS algorithm traverses each edge server once. Therefore, each loop terminates in $O(n)$ operations, where n is the number of edge servers. For the outer loop (line 1-18), since the request scheduling of different service applications is independent, it terminates in $O(n)$ operations. Thus, the time complexity of the DSRS algorithm is $O(n)$.

5. Evaluation

In this section, we conduct experiments to evaluate our DSRS algorithm. First, we analyze the impact of parameters. Then, we present comparison experiments which show the effectiveness of our DSRS algorithm.

In the experiments, we consider 4 MEC systems, each with an edge server providing services for the offloaded requests. There are two types of heterogeneous services. For each service $i \in I$, the request arrival process is generated according to Poisson distribution with arrival rate λ_i [29]. Note that the DSRS algorithm actually requires no knowledge of the statistical information of request arrivals. The computing capacity of the MEC systems is set as $\beta_i \cdot \lambda_i$ where $\beta_i > 1$. Without loss of generality, we assume the MEC systems are heterogeneous with different computing capacities. And the unit scheduling costs of different MEC systems are set to be positively related to its computing capacity.

5.1. Parameter Analysis

5.1.1. Effect of Tradeoff Parameter. Figures 1 and 2 show the time-average scheduling cost and queue length of the MEC systems with different values of V. In Figure 1, it can be seen that the scheduling cost decreases as the value of V increases, which is in accordance with (27) in Theorem 3. This is because as V increases, more weight is put on scheduling cost, and the DSRS algorithm would schedule more service requests to the MEC system with lower unit cost in order to reduce the overall scheduling cost. However, Figure 2 shows that the queue length also rises with the increase of V, which is consistent with (26) in Theorem 3. Nevertheless, the queue length would stabilize gradually with far more increase of V. Together with Figures 1 and 2, we can see that the DSRS algorithm can make a tradeoff between scheduling cost and queue length by adjusting the value of V.

5.1.2. Effect of Service Request Arrival Rate. We analyze the effect of service request arrival rate on the scheduling cost and

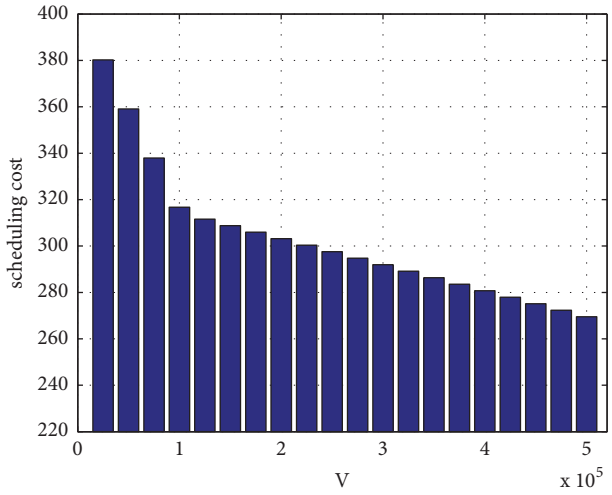

Figure 1: Scheduling cost with different values of V.

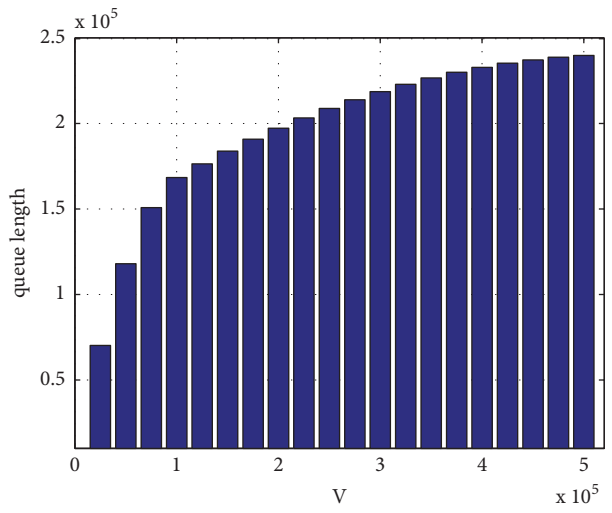

Figure 2: Queue length with different values of V.

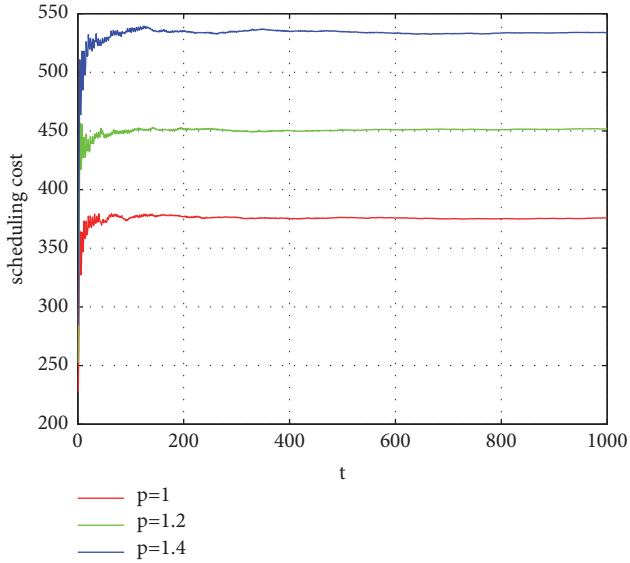

Figure 3: Scheduling cost with different arrival rates.

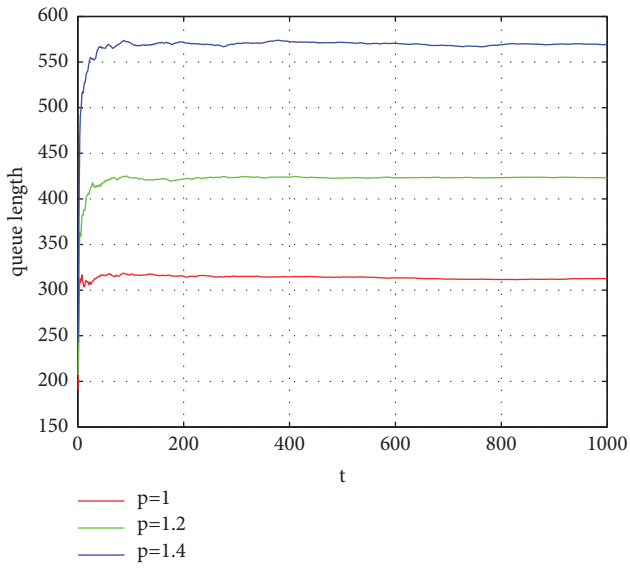

Figure 4: Queue length with different arrival rates.

queue length. In the experiments, for each application $i \in I$, we scale the service request arrival rate up or down to $p \cdot \lambda_i$. We consider three different cases, where $p = 1$, 1.2 and 1.4, respectively. Figures 3 and 4 show that both of the scheduling cost and queue length increase as the request arrival rate increases. Nevertheless, the queue length can stabilize quickly with the increase of service request arrival rate. This shows that our DSRS algorithm can dynamically adjust the request scheduling decisions according to different service request arrivals and maintain the stability of the MEC systems.

5.1.3. Effect of Unit Scheduling Cost. To analyze the effect of unit scheduling cost on the MEC systems, we scale the unit scheduling cost up or down to $q \cdot \gamma_{ij}$. We consider three different cases, where $q = 1$, 1.2 and 1.4, respectively. We can see from Figure 5 that the overall scheduling cost rises as the unit scheduling cost increases, since the scheduling cost of each request increases. In Figure 6, we can see that the queue length of the MEC systems also increases with the increase of unit scheduling cost. The reason is that our DSRS algorithm tries to achieve low scheduling cost by scheduling more requests to the MEC with smaller unit scheduling cost. However, this will lead to the larger queue backlog in some MEC systems.

5.2. Comparison Experiment. We conduct comparison experiment and compare our DSRS algorithm with Randomized algorithm to evaluate the effectiveness of the DSRS algorithm. The Randomized algorithm schedules all the service requests to each MEC system randomly. The scheduling costs and queue lengths of the two algorithms are shown in Figures 7 and 8, respectively.

We can see from Figure 7 that the scheduling cost of our DSRS algorithm is smaller than that of Randomized

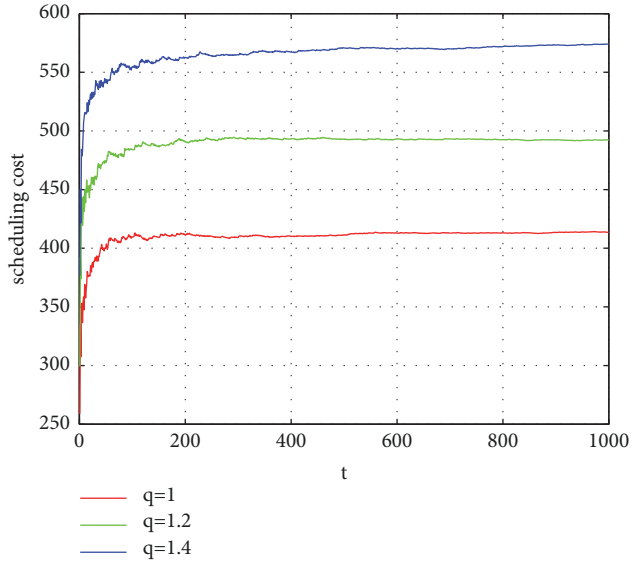

Figure 5: Scheduling cost with different unit scheduling costs.

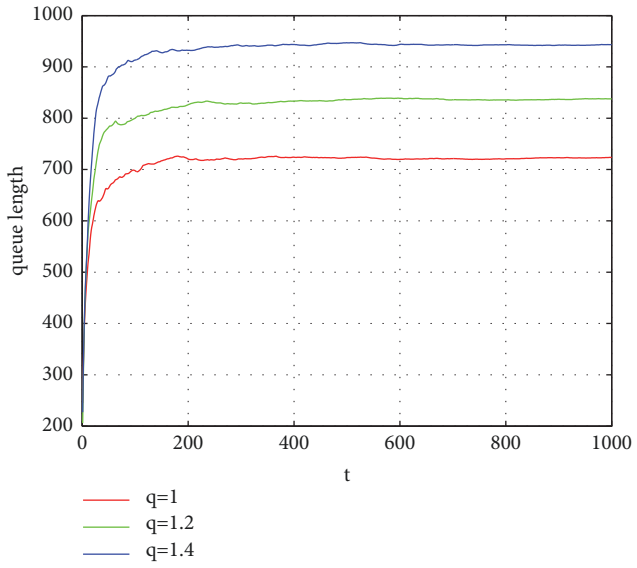

Figure 6: Queue length with different unit scheduling costs.

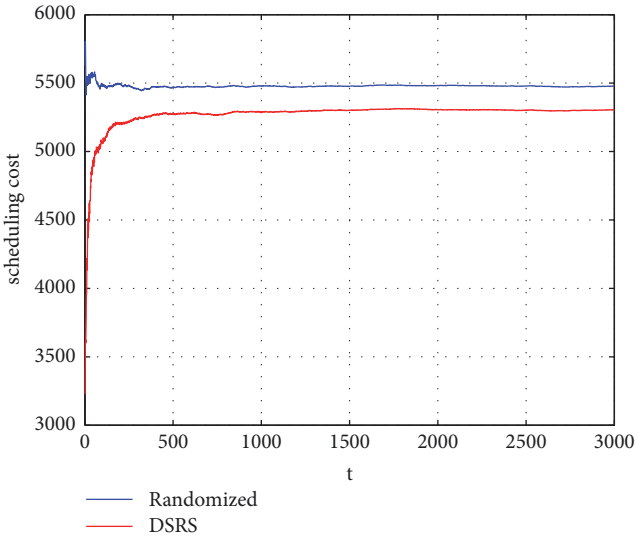

Figure 7: Scheduling cost under different algorithms.

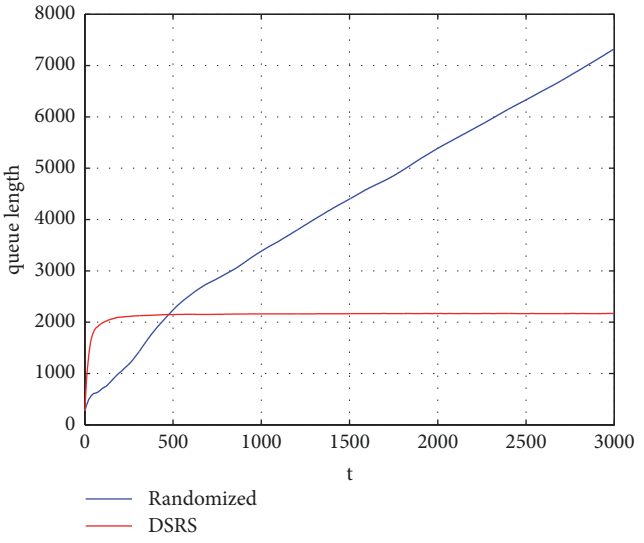

Figure 8: Queue length under different algorithms.

algorithm, which shows the effectiveness of our DSRS algorithm in reducing cost. In Figure 8, we can observe that the queue length of the Randomized algorithm is slightly smaller than our DSRS algorithm at the very beginning. However, as time goes by, the queue length of the Randomized algorithm increases continuously along with the time. The queue length of our DSRS algorithm stabilizes quickly and maintains at a small level. The reason is that our DSRS algorithm can adjust scheduling decisions dynamically according to the current queue backlog and maintain low congestion state in the MEC systems. Together with Figures 7 and 8, we can see the effectiveness of our DSRS algorithm in optimizing both scheduling cost and queue length.

6. Conclusion

In this article, we study dynamic request scheduling for MEC systems. We formulate it as an optimization problem, and the goal is to optimize scheduling cost while providing performance guarantee. We propose the DSRS algorithm to solve the optimization problem, which transforms it to a series of subproblems and solves each one efficiently in a distributed way. Mathematical analysis is presented which demonstrates that the DSRS algorithm can approach the optimal scheduling cost while bounding the queue length. Parameter analysis experiments and comparison experiments are both conducted to verify the effectiveness of the DSRS algorithm.

Data Availability

Most of the simulation experimental data used for supporting the study of this article are included within the article. Further additional information about the data is available from the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (no. 61370065 and no. 61502040), the Key Research and Cultivation Projects at Beijing Information Science and Technology University (no. 5211823411), Beijing Municipal Program for Excellent Teacher Promotion (no. PXM2017 014224 000028), and the Supplementary and Supportive Project for Teachers at Beijing Information Science and Technology University (no. 5111823401).

References

[1] Q. Ye and W. Zhuang, "Distributed and Adaptive Medium Access Control for Internet-of-Things-Enabled Mobile Networks," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 446–460, 2017.

[2] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 755-737, 2017.

[3] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications, IEEE INFOCOM 2016*, pp. 1–9, April 2016.

[4] B.-G. Chun, S. Ihm, P. Maniatis, and eta, "Clonecloud: Elastic execution between mobile device and cloud," in *Proceedings of the 6th ACM EuroSys Conference on Computer Systems (EuroSys '11)*, pp. 301–314, ACM, New York, NY, USA, April 2011.

[5] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proceedings of the IEEE INFOCOM*, pp. 945–953, March 2012.

[6] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen, "Comet: Code offload by migrating execution transparently," in *Proceedings of the Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, pp. 93–106, Berkeley, Calif, USA, 2012.

[7] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic Resource and Task Allocation for Energy Minimization in Mobile Cloud Systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, 2015.

[8] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3317–3329, 2015.

[9] X. Lyu, W. Ni, H. Tian et al., "Optimal schedule of mobile edge computing for internet of things using partial information," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2606–2615, 2017.

[10] J. Liu, S. Wang, A. Zhou, F. Yang, and R. Buyya, "Availability-aware Virtual Cluster Allocation in Bandwidth-Constrained Datacenters," in *Proceedings of the IEEE Transactions on Services Computing*, pp. 1-1, 2017.

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

[12] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and CPU time allocation for mobile edge computing," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, December 2016.

[13] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[14] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A Survey on Service Migration in Mobile Edge Computing," *IEEE Access*, vol. 6, pp. 23511–23528, 2018.

[15] Y. Kim, J. Kwak, and S. Chong, "Dual-Side Optimization for Cost-Delay Tradeoff in Mobile Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 2, pp. 1765–1781, 2018.

[16] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[17] S. Wang, Y. Zhao, L. Huang, J. Xu, and C.-H. Hsu, "QoS prediction for service recommendations in mobile edge computing," *Journal of Parallel and Distributed Computing*, 2017.

[18] J. Zheng, Y. Cai, Y. Wu, and X. S. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," in *Proceedings of the IEEE Transactions on Mobile Computing*, pp. 1-1, 2018.

[19] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[20] H. Tan, Z. Han, X. Li, and F. C. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proceedings of the IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, Atlanta, Ga, USA, May 2017.

[21] S. Deng, L. Huang, D. Hu, J. L. Zhao, and Z. Wu, "Mobility-Enabled Service Selection for Composite Services," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 394–407, 2016.

[22] Q. Ye and W. Zhuang, "Token-Based Adaptive MAC for a Two-Hop Internet-of-Things Enabled MANET," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1739–1753, 2017.

[23] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Performance Evaluation*, vol. 91, pp. 205–228, 2015.

[24] Y. Nan, W. Li, W. Bao et al., "Adaptive Energy-Aware Computation Offloading for Cloud of Things Systems," *IEEE Access*, vol. 5, pp. 23947–23957, 2017.

[25] Z. Zhou, F. Liu, H. Jin, B. Li, B. Li, and H. Jiang, "On arbitrating the power-performance tradeoff in SaaS clouds," in *Proceedings of the 32nd IEEE Conference on Computer Communications, IEEE INFOCOM 2013*, pp. 872–880, Italy, April 2013.

[26] S. Ren, Y. He, and F. Xu, "Provably-efficient job scheduling for energy and fairness in geographically distributed data centers," in *Proceedings of the 32nd IEEE International Conference on*

Distributed Computing Systems, ICDCS 2012, pp. 22–31, China, June 2012.

[27] K. Ha, P. Pillai, W. Richter et al., "Just-in-time provisioning for cyber foraging," in *Proceedings of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '13)*, pp. 153–166, New York, NY, USA, 2013.

[28] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*, Morgan and Claypool, 2010.

[29] E. Chlebus and J. Brazier, "Nonstationary Poisson modeling of web browsing session arrivals," *Information Processing Letters*, vol. 102, no. 5, pp. 187–190, 2007.