# Fog Network Task Scheduling for IoT Applications


**Chongchong Zhang**
Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, 200050, China
University of Chinese Academy of Sciences, Beijing, 101408, China
chongchong.zhang@mail.sim.ac.cn

**Fei Shen**
Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, 200050, China
fei.shen@mail.sim.ac.cn

**Jiong Jin**
School of Software and Electrical Engineering, Faculty of Science, Engineering and Technology, Swinburne University of Technology, VIC 3122, Melbourne, Australia

**Yang Yang**
School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China



## Abstract

In the Internet of Things (IoT) networks, the data traffic would be very bursty and unpredictable. It is therefore very difficult to analyze and guarantee the delay performance for delay-sensitive IoT applications in fog networks, such as emergency monitoring, intelligent manufacturing, and autonomous driving. To address this challenging problem, a Bursty Elastic Task Scheduling (BETS) algorithm is developed to best accommodate bursty task arrivals and various requirements in IoT networks, thus optimizing service experience for delay-sensitive applications with only limited communication resources in time-varying and competing environments. To better describe the stability and consistence of Quality of Service (QoS) in realistic scenarios, a new performance metric "Bursty Service Experience Index (BSEI)" is defined and quantified as delay jitter normalized by the average delay. Finally, the numeral results shows that the performance of BETS is fully evaluated, which can achieve $5-10$ times lower BSEI than traditional task scheduling algorithms, e.g. Proportional Fair (PF) and the Max Carrier-to-Interference ratio (MCI), under bursty traffic conditions. These results demonstrate that BETS can effectively smooth down the bursty characteristics in IoT networks, and provide much predictable and acceptable QoS for delay-sensitive applications.



**2012 ACM Subject Classification** Networks

**Keywords and phrases** Task Scheduling, Internet of Things, fog network, delay sensitive

**Digital Object Identifier** 10.4230/OASIcs.Fog-IoT.2020.10

**Funding** National Natural Science Foundation of China under grant No. 61871370, and Natural Science Foundation of Shanghai under grant 18ZR1437500


## 1 Introduction

In future Internet of Things (IoT) networks, billions or even trillions of heterogeneous machines and devices are connected by multiple advanced technologies including the Wireless Sensor Networks (WSN), Radio Frequency Identification (RFID), cloud-edge/fog caching and computing [3, 10] and etc. With the acceleration of 5G commercial deployment, a large proportion of the IoT applications is delay-sensitive, such as emergency monitoring, intelligent manufacturing, disaster relief, online games and autonomous driving. The internet traffic of these delay-sensitive applications is bursty and unpredictable at different time scales [8]. For example, some delay-sensitive applications like interactive multiplayer online games, the event-driven applications, intelligent manufacturing and autonomous driving require

stable delay performance with low delay jitter. In such cases, multipath signals need to be received simultaneously in order to make the next-step control decisions. It is obvious to observe that the QoS of IoT applications depends greatly on the delay performance of the traffic data, which will reflect in the economic benefits of different service providers [11].

However, the wireless communication and computing resources in mobile networks are highly limited, which make difficult to meet the fast growing demands of the booming IoT applications with heterogeneous delay requirements. Therefore, efficient management of network resources and flexible task scheduling algorithms play important roles in both academic researches and industrial applications, especially with bursty task arrivals, dynamic network topologies [2] and unpredictable terminal behaviors [1], in order to guarantee the performance in both the average delay and the delay jitter.

To overcome the severe contradiction between the high delay requirements of the massive traffic generated in the IoT networks and the scarce communication resources, the first thing is to understand the busty characteristics of the data traffic, wherein terminal tasks arrive randomly at the terminal buffers with different arrival rate and task sizes. To take full advantage of the varying characteristic of the wireless channel state in time domain, frequency domain, code domain, etc., plentiful researches are carried out in 5G and IoT networks focusing on the system indexes including the mobility [12, 5], packet delay, and the high frequency transmission [6]. Traditional scheduling algorithms like Proportional Fair (PF) and Max Carrier-to-Interference ratio (MCI) just try to achieve satisfactory bit-level throughput performance [5, 7]. Therefore, novel task scheduling algorithms need to be proposed in order to guarantee the heterogeneous requirements of various delay-sensitive IoT applications with bursty traffic load and dynamic network environments.

The rest of this paper is organized as follows. The system model for terminal task delay are provided in Section 2. Section 3 gives the solution of probability distribution of task delay. The BETS algorithm is proposed in Section 4. Numerical validations are performed in Section 5. Finally, Section 6 concludes this paper.

# 2 System Model

## 2.1 System Overview

In the traffic layer, an IoT cluster with delay sensitive applications is considered. As shown in Fig. 1, $N$ terminals are randomly distributed in the coverage area of the central server. In the MAC layer, each Time Slot (TS) with duration time of $\Delta t$, and the entire system radio resources are divided into $M$ orthogonal parts, i.e. $M$ Resource Blocks (RBs). At the start of each TS, the central scheduler allocates these $M$ RBs to the terminals according to the metric $H_{jm}$ defined by a certain task scheduling algorithm, and the RB will be allocated to the terminal who has the highest scheduling metric on it, which can be formulated as

$$(j^*, m^*) = \arg\max_{j,m} H_{jm}. \tag{1}$$

The task delay in this layer is mainly reflected by the sensing and allocation of the ratio resources.

In the physical layer, the instantaneous data rate of terminal $j$ on RB $m$, i.e. $r_{jm}$, is set to be Gaussian distributed denoted by $N(E[r_{jm}], \sigma^2_{jm})$, according to the research on capacity approximation in a Rayleigh fading environment. The data rates of the same terminal on different RBs are assumed to be i.i.d distributed, and the distribution parameters are related to the terminal's position in this cluster [4]. The Probability Distribution Function (PDF) of $r_{jm}$ is $f_{r_{jm}}(x)$. The task delay in this layer is mainly reflected by the transmission delay of the terminal data.

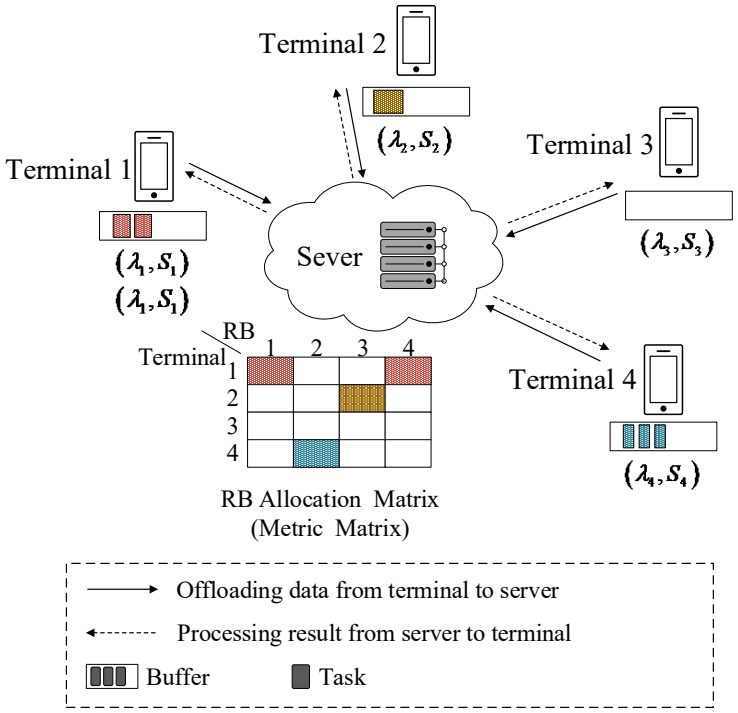



**Figure 1** Task scheduling in an IoT cluster with bursty traffic load. The randomly generated terminal tasks are offloaded to the central server through wireless link.

The tasks generated at terminal $j$ follow a Poisson arrival process with arrival rate $\lambda_j$ and random task size $S_j$, which arrive at the terminal buffer and need to be offloaded to the central server to be processed. processing delays in the terminal device and the cloud server.

In order to obtain the statistical result of terminal task delay, the offloading processes of the two consecutive tasks of the same terminal and the related parameters are modeled in Fig. 2, where the blue and red lines represent the bustsy timelines of the previous and current tasks respectively. The current task has to wait in the buffer before the delivery of the previous task. The busty nature of the traffic and the varying channel state bring challenge to the analysis, which has to take all the traffic layer, MAC layer and physical layer into consideration.

Taking the discreteness of the scheduling process in each TS into consideration and omitting the terminal mark, the time durations in Fig. 2 are defined as follows:

- $t_{\mathrm{a}} = n_{\mathrm{a}} \cdot \Delta t$: the arrival interval, i.e. the time duration between the arrivals of this two tasks;
- $t_{\mathrm{w}} = n_{\mathrm{w}} \cdot \Delta t$: the waiting time, i.e. the time duration between the current task's arrival and its start of transmission;
- $t_{\mathrm{t}} = n_{\mathrm{t}} \cdot \Delta t$: the communication delay, i.e. the time duration between the current task's start of transmission and its delivery;
- $t_{\mathrm{d}} = n_{\mathrm{d}} \cdot \Delta t$: the task delay of the current task, i.e. the time duration between the arrival and the delivery of the task;
- $t'_{\mathrm{d}} = n'_{\mathrm{d}} \cdot \Delta t$: the task delay of the previous task.

The time durations described above are all discrete, and the parameters with the form of "$n_\star$" are nonnegative integers which represent the TS amounts of corresponding time durations. In real systems, the task delay cannot be infinitely large. Thus a threshold $n_{\mathrm{d,max}}$ is defined

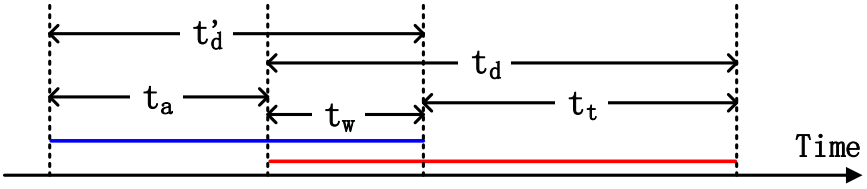



**Figure 2** The offloading processes of two consecutive tasks of a same terminal.

for $n_{\mathrm{d}}$, and the probability $\Pr(n_{\mathrm{d}} > n_{\mathrm{d,max}})$ is small enough to be neglected. Therefore, the ranges of these integers are $1 \le n_{\mathrm{d}}, n'_{\mathrm{d}} \le n_{\mathrm{d,max}}$, $0 \le n_{\mathrm{w}} \le (n_{\mathrm{d,max}} - 1)$, $1 \le n_{\mathrm{t}} \le n_{\mathrm{d,max}}$, and $n_{\mathrm{a}} \ge 0$. It is obvious that the probability of the time duration $t_\star$ is equivalent to the probability distribution of the corresponding TS amount $n_\star$. Thus we concentrate on the PDF of the delay TS amount $n_{\mathrm{d}}$ when carrying out latter analyses.

## 3 Problem Solution

To obtain the probability distribution of task delay is equivalent to solving the equation set proposed in the following theorem.

▶ **Theorem 1.** *The theoretical task offloading delay distribution can be calculated through the following equation set.*

$$V_{\mathrm{d}}^{n_{\mathrm{d,max}}\times 1} = A_{\mathrm{t}}^{n_{\mathrm{d,max}}\times n_{\mathrm{d,max}}} A_{\mathrm{a}}^{n_{\mathrm{d,max}}\times n_{\mathrm{d,max}}} V_{\mathrm{d}}^{n_{\mathrm{d,max}}\times 1} \tag{2a}$$

$$[1 \quad 1 \quad \cdots \quad 1]^{n_{\mathrm{d,max}}\times 1} V_{\mathrm{d}}^{n_{\mathrm{d,max}}\times 1} = 1. \tag{2b}$$

*In the above expressions, $V_{\mathrm{d}}$ with dimension $n_{\mathrm{d,max}} \times 1$ is the probability distribution vector for the task delay TS amount $n_{\mathrm{d}}$. $A_{\mathrm{t}}$ and $A_{\mathrm{a}}$ with dimension $n_{\mathrm{d,max}} \times n_{\mathrm{d,max}}$ are the parameter arrays, whose elements are the probability that the task delay is $t_{\mathrm{d}} = n_{\mathrm{d}} \cdot \Delta t$ on condition that the task waiting time is $t_{\mathrm{w}} = n_{\mathrm{w}} \cdot \Delta t$, i.e. $\Pr(n_{\mathrm{d}}|n_{\mathrm{w}})$, and the probability that the task waiting time is $t_{\mathrm{w}} = n_{\mathrm{w}} \cdot \Delta t$ on condition that the task delay of the previous task is $t'_{\mathrm{d}} = n'_{\mathrm{d}} \cdot \Delta t$, i.e. $\Pr(n_{\mathrm{w}}|n'_{\mathrm{d}})$, respectively.*

**Proof.** By using the law of total probability twice, the probability for terminal task delay to be $n_{\mathrm{d}}$ TSs, i.e. $\Pr(n_{\mathrm{d}})$, can be expanded as

$$\begin{aligned}\Pr(n_{\mathrm{d}}) &= \sum_{n_{\mathrm{w}}=0}^{n_{\mathrm{d,max}}-1} \Pr(n_{\mathrm{d}}|n_{\mathrm{w}}) \cdot \Pr(n_{\mathrm{w}}) \\ &= \sum_{n_{\mathrm{w}}=0}^{n_{\mathrm{d,max}}-1} \Pr(n_{\mathrm{d}}|n_{\mathrm{w}}) \cdot \sum_{n'_{\mathrm{d}}=1}^{n_{\mathrm{d,max}}} \Pr(n_{\mathrm{w}}|n'_{\mathrm{d}}) \cdot \Pr(n'_{\mathrm{d}}), \\ &\qquad n_{\mathrm{d}} = 1, 2, \cdots, n_{\mathrm{d,max}}. \end{aligned} \tag{3}$$

We also have

$$\Pr(n_{\mathrm{d}}) = \Pr(n'_{\mathrm{d}}), \quad n_{\mathrm{d}} = n'_{\mathrm{d}} = 1, 2, \cdots, n_{\mathrm{d,max}}, \tag{4}$$

which comes from the fact that for the same terminal, the task delays of all the tasks follow the same statistical probability distribution. For the probability distribution of task delay, i.e. $\Pr(n_{\mathrm{d}})$, the normalized constraint shown below also needs to be satisfied.

$$\sum_{n_{\mathrm{d}}=1}^{n_{\mathrm{d,max}}} \Pr(n_{\mathrm{d}}) = 1. \tag{5}$$

The vectorial and array representation of the formulas (3) (4) is the formula (2a), and the vectorial representation of the formula (5) is the formula (2b). This completes the proof of the Theorem 1. ◀

## 4 BETS Algorithm

The traffic load of the terminal in IoT networks is normally a series of tasks, which have dynamic task sizes and randomly generated at the transmitter [9]. The traditional algorithm like PF scheduling tries to pursue satisfactory bit-level throughput, while the QoS of delay sensitive IoT applications draws more concern. All these new features call for newly designed task scheduling policies, which should take into the following considerations.

- Task delay rather than the terminal throughput should become the main scheduling purpose.
- The bursty and heterogeneous characteristics of different applications should be smoothed to provide consistent terminal experience.

Therefore, we introduce the Bursty Elastic Task Scheduling (BETS) algorithm to cope with the bursty nature of IoT traffic.

In a system adopting the BETS algorithm, scheduling decisions are made for all the RBs at the start of TS $n$. The scheduling metric of terminal $j$ on RB $m$ is defined as

$$H_{jm} = \frac{r_{jm}}{R_j / S_j}, \tag{6}$$

where $R_j$ is the historical average throughput of terminal $j$. The detailed execution steps of BETS are described in Algorithm 1, and (9) is the updating formula of $R_j$. The parameter $k$ in the update formula (9) is the average window length, and $R'_j$ is the updated historical average throughput of terminal $j$. $I_{jm}$ is the indicator variable, the function of $I_{jm}$ is defined as

$$I_{jm} = \begin{cases} 1, & \text{if RB } m \text{ is allocated to terminal } j, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

In the scheduling metric of BETS algorithm 1, the terminal with larger task size will have a higher scheduling metric and vice versa, thus a smaller delay jitter can be obtained through BETS. Besides, the BETS is equivalent to the PF scheduling algorithm in the cases that all terminals have the same task size. For comparison, the scheduling metric of PF scheduling algorithm is defined as

$$H_{jm} = \frac{r_{jm}}{R_j}. \tag{10}$$

As for the MCI scheduling algorithm, it directly takes the instantaneous data rate $r_{jm}$ as the scheduling metric and always tries to maximize the system throughput. The execution steps of PF and MCI scheduling algorithms are similar to that of BETS algorithm except for the calculation of the scheduling metric matrix $\mathbf{H}$.

**Algorithm 1** BETS Algorithm.

At the start of each TS, calculate the scheduling metric matrix $\mathbf{H}$ with element $H_{jm}$ in row $j$ and column $m$ as

$$H_{jm} = \frac{r_{jm}}{R_j / S_j};$$

The rows corresponding to the terminals with empty buffers are set to be 0;
**while** there are nonzero elements in $\mathbf{H}$, **do**
    Find the maximum value in the scheduling metric matrix $\mathbf{H}$ as

$$(j^*, m^*) = \arg\max_{j,m} H_{jm}; \tag{8}$$

    Allocate RB $m^*$ to terminal $j^*$;
    Set the elements in column $m^*$ to be 0;
    **if** Terminal $j^*$ has got enough radio resource to clear its buffer, **then**
        Set the elements in row $j^*$ to be 0;
    **end if**
    Update the historical average throughput of each terminal as

$$R'_j = \left(1 - \frac{1}{k}\right) R_j + \frac{1}{k} \sum_{m=1}^{M} I_{jm} \times r_{jm}; \tag{9}$$

**end while**
**return** the scheduling results;

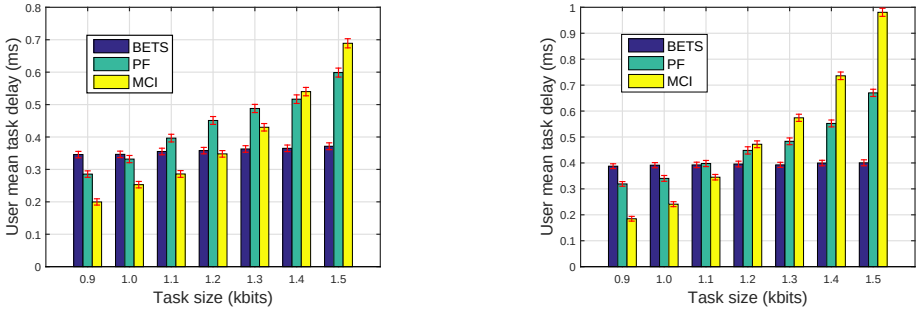
## 5 Numerical Validations

In this section, the delay performances of BETS, PF, and MCI scheduling algorithms with varying task size among terminals are investigated in an IoT cluster. In each TS with duration time of 0.1 ms, there are 50 orthogonal RBs to be allocated to multiple terminals according to certain task scheduling algorithms including not only the proposed BETS algorithm, but also the traditional PF and MCI scheduling algorithms. The mean values of the Gaussian distributed instantaneous data rates of the system terminals range from 500 kbps to 1500 kbps as results of the terminals' different positions in the IoT cluster. The average window length of the BETS and PF scheduling algorithms is 500. A total duration of 4 s is set for the simulation process.

In the cases that the task size of the terminals follows Pareto or exponential distributions with mean value of 1 bits, the mean task delays for different task sizes are provided in Fig. 3.

As shown from the numerical results, the BETS achieves a higher fairness for delay performance with varying task size among terminals than those achieved by PF and MCI scheduling algorithms.

For IoT applications with varying task size, it's important to achieve an equalizing delay performance for different task sizes. In the following definition, the the bursty service experience index (BSEI) is introduced to evaluate the delay experience among the system terminals.




**(a)** Pareto distributed task size with mean value 1 kbits.

**(b)** Exponential distributed task size with mean value 1 kbits.

**Figure 3** Terminal mean task delay for varying task size. The theoretical estimation errors are also provided.

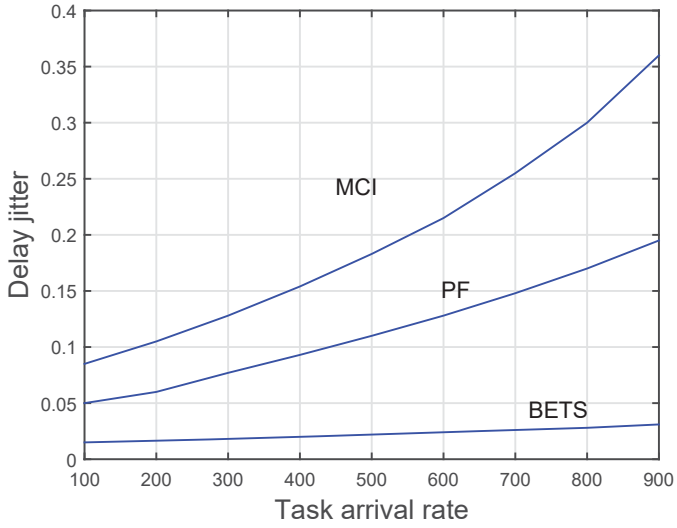



**Figure 4** Delay jitter comparisons of the three task scheduling algorithms.

▶ **Definition 2.** *The BSEI of task delay is represented by the following performance index.*

$$Q_{\mathrm{d}} = \frac{\sigma(t_{\mathrm{d}})}{E(t_{\mathrm{d}})}, \tag{11}$$

*where $\sigma(t_{\mathrm{d}})$ and $E(t_{\mathrm{d}})$ are the standard deviation and average value of the task delay respectively.*

The standard deviation $\sigma(t_{\mathrm{d}})$ and the BSEI $Q_{\mathrm{d}}$ represent the absolute and relative jitters of the task delay respectively. A smaller $Q_{\mathrm{d}}$ indicates lower jitter and better BSEI for task delay in the system.

The jitter and the BSEI of terminal task delay are shown in Fig. 4 and Fig. 5, respectively, and the BSEI is the ratio of the delay jitter and the average delay as shown in (11). The results in these two figures further validate the superiority of BETS algorithm for achieving better SE and a more consistent performance for terminal task delay, while the other two algorithms achieves much large delay jitter and thus a poor experience for delay-sensitive

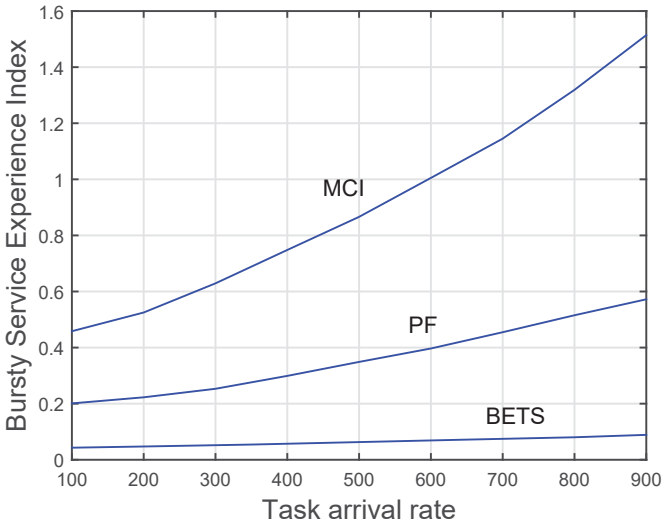



**Figure 5** BSEI comparisons of the three task scheduling algorithms.

tasks. As shown in Fig. 5, the proposed BETS algorithm can significantly reduce the BSEI, e.g. at a typical task arrival rate of 500, BETS can achieve about 5 times and 10 times more consistant service experience than MCI and PF, respectively, for delay sensitive IoT applications. It's because the introducing of the task size $S_j$ to the scheduling metric of the BETS algorithm. Terminals with higher traffic load have higher scheduling metric as shown in (6).

## 6 Conclusions

In this paper, the problem of theoretical performance analysis of terminal task delay in IoT networks was investigated. In order to cope with the bursty nature of the traffic statistics in various IoT applications, a novel traffic scheduling algorithm named BETS was introduced, which takes the terminal task size into consideration when making scheduling decisions. Moreover, a new performance metric "Bursty Service Experience Index (BSEI)" is defined and quantified as delay jitter normalized by the average delay to better describe the stability and consistence of Quality of Service (QoS) in realistic scenarios. The numeral results show that the task delay performance of BETS is better than PF and MCI scheduling algorithms.

## References

1 Najah Abu-Ali, Abd Elhamid M. Taha, Mohamed Salah, and Hossam Hassanein. Uplink scheduling in LTE and LTE-Advanced: Tutorial, survey and evaluation framework. *IEEE Communications Surveys & Tutorials*, 16(3):1239–1265, 3rd quarter 2014.

2 Samaresh Bera, Sudip Misra, Sanku Kumar Roy, and Mohammad S. Obaidat. Soft-WSN: Software-defined WSN management system for IoT applications. *IEEE Systems Journal*, PP(99):1–8, 2016.

3 Mung Chiang, Sangtae Ha, I Chih-Lin, Fulvio Risso, and Tao Zhang. Clarifying fog computing and networking: 10 questions and answers. *IEEE Communications Magazine*, 55(4):18–20, April 2017.

4 E. Liu and K. K. Leung. Proportional fair scheduling: Analytical insight under Rayleigh fading environment. In *2008 IEEE Wireless Communications and Networking Conference*, pages 1883–1888, March 2008.

5 R. Margolies, A. Sridharan, V. Aggarwal, R. Jana, N. K. Shankaranarayanan, V. A. Vaishampayan, and G. Zussman. Exploiting mobility in proportional fair cellular scheduling: Measurements and algorithms. *IEEE/ACM Transactions on Networking*, 24(1):355–367, February 2016.

6 Solmaz Niknam, Ali Arshad Nasir, Hani Mehrpouyan, and Balasubramaniam Natarajan. A multiband OFDMA heterogeneous network for millimeter wave 5G wireless applications. *IEEE Access*, 4(99):5640–5648, 2017.

7 Wiroonsak Santipach, Kritsada Mamat, and Chalie Charoenlarpnopparut. Outage bound for max-based downlink scheduling with imperfect CSIT and delay constraint. *IEEE Communications Letters*, 20(8):1675–1678, August 2016.

8 Philipp Schulz, Maximilian Matthe, Henrik Klessig, Meryem Simsek, Gerhard Fettweis, Junaid Ansari, Shehzad Ali Ashraf, Bjoern Almeroth, Jens Voigt, and Ines Riedel. Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture. *IEEE Communications Magazine*, 55(2):70–78, February 2017.

9 A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman. Characterizing and classifying IoT traffic in smart cities and campuses. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 559–564, May 2017.

10 Yang Yang. Multi-tier computing networks for intelligent IoT. *Nature Electronics*, 2(1):4–5, January 2019.

11 Yang Yang, Jing Xu, Guang Shi, and Cheng-Xiang Wang. *5G Wireless Systems*. Springer, 2018.

12 Z. Zhang, C. Jiao, and C. Zhong. Impact of mobility on the uplink sum rate of MIMO-OFDMA cellular systems. *IEEE Transactions on Communications*, 65(10):4218–4231, October 2017.