A Queueing Game Based Management Framework for Fog Computing with Strategic Computing Speed Control

Changyan Yi, Member, IEEE, Jun Cai, Senior Member, IEEE, Kun Zhu, Member, IEEE, and Ran Wang, Member, IEEE

Abstract—In this paper, a novel management framework for fog computing with strategic computing speed control at fog nodes (FNs) is studied. In the considered model, mobile users declare requests of offloading resource-hungry computation tasks that are dynamically collected at a dedicated edge server (ES). Upon receiving these requests, the ES can decide to either self-process or delegate some workloads to third-party FNs for maximizing the overall management profit. Unlike the existing work, this paper takes into account strategic behaviors of FNs in computing speed control, i.e., each FN can strategically allocate its computing resource to maximize its utility, which consists of the benefit gained from executing offloaded tasks and the cost incurred by dissatisfied (delayed) service to its own subscribed tasks. To jointly address the long-term system performance and FNs’ strategic interactions, a scheduling mechanism integrating a noncooperative game and a queueing model is formulated. We then investigate two delegation reward settings, i.e., constant and utility-dependent delegation prices, and propose efficient adaptive algorithms to determine the optimal workload distribution at the ES and the computing speed equilibrium among FNs. Both theoretical analyses and simulations are conducted to evaluate the performance of the proposed solutions and demonstrate their superiorities over counterparts.

Index Terms—Fog computing, computing speed control, workload distribution, game, queueing.

1 INTRODUCTION

With the prevalence of smart mobile devices (e.g., smartphones and tablets), a variety of advanced mobile applications, such as pervasive healthcare monitoring, natural language processing and interactive gaming, are newly developed for fulfilling people’s increasing needs of high-quality of living and entertaining [1]. These emerging applications are commonly computation-intensive and energy-consuming, leading to heavy burdens on resource-constrained mobile devices. To resolve this conflict, fog computing [2] has been proposed as a promising paradigm for Internet of Things (IoT) in providing prompt and flexible complementary computing services to support mobile end users. By enabling fog computing, in addition to dedicated edge servers deployed at wireless base stations, third-party fog nodes (e.g., private servers or devices) can also be recruited to help mobile users in expanding their computing capacities if necessary [3].

Due to the great potential, fog computing has gained substantial research attentions [4]. Recent studies in this area include energy-efficient computation offloading [5], [6], transmission scheduling with the low-latency requirements [7], [8], and joint radio-and-computational resource allo-

- C. Yi, K. Zhu and R. Wang are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, and the Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, Jiangsu, 211106, China. (E-mail: {changyan.yi, zhukun, wangran}@nuaa.edu.cn).
- J. Cai is with the Department of Electrical and Computer Engineering, Concordia University, Montréal, QC, H3G 1M8, Canada. (E-mail: jun.cai@concordia.ca).

Corresponding authors: Jun Cai and Kun Zhu.

cations [9], [10]. However, most of existing work focused on the optimal management at computation-requester ends (i.e., mobile users), while ignoring the computing service management at server ends (i.e., fog nodes). In fact, because of the following reasons, a careful study of the computing service management (particularly the workload distribution among fog nodes) is also imperative and crucial.

i) In practice, third-party servers are normally implemented for specific application purposes and are required to provide computing services to their own subscribers. For instance, road-side units (RSUs) are ordinarily built for supporting Internet of Vehicular (IoV) applications [11]. Thus, even though RSUs may be recruited as fog nodes for other computation applications, they have to deal with their own subscribed computation tasks simultaneously. This motivates us to investigate how fog nodes would allocate their computing resources (or control their computing speeds) for different uses in the system management.

ii) For real-time applications, such as live streaming and medical surveillance, their computation tasks may be delay-sensitive and generated randomly over the time [12]–[15]. These necessitate a fast-responsive system control with long-term performance guarantees.

iii) Owing to high device intelligence, all server ends (both edge and fog servers) in fog computing systems may behave strategically and selfishly [16], [17]. As a result, the dedicated edge server (acting as the central controller) will manage the computation workload distribution for benefiting itself, and each fog node will pursue its self-interest by strategically determining its
computing speed to potentially affect the workload distribution among all servers and the overall system performance. Obviously, these lead to noncooperative interactions/competitions among different individual nodes, which requests the exploration of a game-theoretic decision-making process.

Although some preliminary efforts [18]–[20] have been recently made on workload distributions for fog computing systems, all of them imposed a common assumption that fog nodes are willing to fully contribute their computing capacities in serving offloaded computation tasks regardless of their own subscribers, which may not always hold in reality. Moreover, none of them addressed all aforementioned features, as taking all of them into account significantly increases the complexity of the involved workload distribution problem from the following aspects.

a. The fog computing management includes dynamic control of delay-sensitive computation tasks and operations of all servers. To characterize the implicit end-to-end (communication and computation) performance, a task-level queueing analysis is desired.

b. Since utility functions of fog nodes depend on not only their strategically determined computing speeds, but also the outcome of the queueing management, their strategic interactions and game behaviors have to be analyzed with a joint consideration of the induced queueing effects, which results in a complicated queueing game problem.

c. For efficiency and robustness, it is required that the workload distribution should be optimized with the objective of achieving a certain goal (e.g., maximizing the overall management profit), while at the same time guaranteeing that no third-party fog node will unilaterally change its strategy in the game for more benefits. This implies the need of an optimal queueing scheduling integrated with the equilibrium solution of the game, which makes the system management even more difficult.

To tackle these challenges and fill the gap in the literature, in this paper, we propose a queueing game-based management framework to describe both the dynamic nature of the fog computing system and the inherent strategic interactions among third-party fog nodes. The proposed management framework aims to produce the long-term optimal workload distribution and achieve the equilibrium of fog nodes’ computing speed control. In the considered model, computation offloading requests declared by mobile users are first collected at a dedicated edge server deployed at the base station. Upon receiving the offloading request of each computation task, the edge server can then decide to either self-process the task or delegate it to a third-party fog node. As the central controller, the edge server determines the workload distribution among different servers (edge server and fog nodes) according to the tradeoff between its self-processing costs (e.g., the energy consumption and the delay costs) and delegation costs (i.e., the reward for incentivizing third-party fog nodes). The task-level management is further modeled as a queueing game, where each fog node acting as an individual player can strategically adjust its computing speed for maximizing its own utility consisting of the reward gained from executing offloaded tasks and the cost incurred by dissatisfied (delayed) service to its own subscribers. After that, we jointly analyze the queueing and game performances, and design efficient algorithms to derive the corresponding solution, i.e., the system management decisions of the dedicated edge server and the strategic computing speed of each third-party fog node, for different delegation reward settings.

The main contributions of this paper are summarized in the following.

- With the objective of balancing the tradeoff between the edge server’s processing and delegation costs, a long-term optimization problem for the computation workload distribution in the dynamic fog computing system is formulated.
- By considering the strategic computing speed control of third-party fog nodes and the resulted competitions/interactions, a noncooperative queueing game is defined.
- An efficient approach for the problem with a constant delegation price is first proposed, which consists of a novel load-balancing rule and an adaptive algorithm, to determine the optimal workload distribution at the ES and find the computing speed equilibrium among FNs.
- We then extend the proposed approach to a more general form, in which the utility-dependent delegation price setting can be adopted.
- Numerical simulations are conducted to examine all theoretical analyses and show the superiorities of the proposed algorithms over counterparts.

The rest of this paper is organized as follows: Section 2 provides a brief review of related work and highlights the novelties of this paper. Section 3 presents the system model and problem formulation of the management framework for the workload distribution with strategic computing speed control in fog computing. In Section 4, efficient approaches based on both constant and utility-dependent delegation price settings are proposed and analyzed. Simulation results are given in Section 5, followed by conclusions in Section 6.

2 RELATED WORK

As one of the key enabling technologies for IoT, fog computing has drawn a lot of research attentions from both academia and industry in recent years. For instance, Yang et al. in [6] proposed an energy-efficient scheduling algorithm for fog computing with user collaborations. Gu et al. in [9] developed a matching-game framework for joint radio-and-computational resource allocations in IoT fog computing. Both papers were restricted to quasi-static network scenarios, where the arrival, transmission and processing of computation tasks were all assumed to be time-invariant. To further support real-time mobile applications, the latest work has been dedicated in studying fog computing with dynamic network settings. For example, Pu et al. in [5] leveraged a Lyapunov-based algorithm in designing an optimal task offloading policy for fog computing with D2D communications. In [7], Lee et al. presented an online method for minimizing the maximum communication and computation latency in fog networks. However, most of
them focused on the resource management and scheduling at computation-requester ends (i.e., mobile users) only.

It is worth noting that, in fog computing systems, the computing service management (particularly the workload distribution) at server ends (i.e., fog nodes) is also of great importance, yet has rarely been discussed in the literature. This motivated some preliminary research efforts [18]–[20]. Specifically, Deng et al. in [18] solved a workload assignment problem for fog-cloud computing with a balanced tradeoff between power consumption and service delay. In [19], a dynamic workload scheduling algorithm for single-hop fog architecture was introduced. Besides, Fan et al. in [20] proposed a workload allocation scheme for minimizing delays in hierarchical cloudlet networks. However, none of them considered that fog nodes could strategically determine their computing speeds and might reserve computing capacities for their own subscribers.

In fact, the workload distribution among servers with the speed scaling capability has been recognized and investigated in the conventional cloud computing [21]–[23]. Unfortunately, all solutions and algorithms developed in these papers cannot be applied for fog computing due to the unmatched system settings: i) fog computing involves not only computing processes of computational tasks, but also their wireless transmissions, so that the end-to-end performance has to be characterized; ii) servers in fog computing frameworks are hierarchical rather than parallel, meaning that tasks can be either executed at the dedicated edge server or dispatched to third-party fog nodes; and iii) unlike cloud servers, fog nodes may belong to different parties that have to simultaneously deal with their own subscribed tasks, and thus both gains and costs caused by computing resource allocations (or computing speed control) have to be taken into account.

To sum up, different from the existing work, this paper studies a queueing game based management framework to obtain the long-term optimal workload distribution for the dynamic fog computing system with strategic computing speed control.

### 3 System Model and Problem Description

In this section, the network model of the fog computing system is first described. Then, a queueing game based management framework for workload distribution and strategic computing speed control is formulated. For convenience, Table 1 lists some important notations used in this paper.

#### 3.1 Network Model

Consider a fog computing system, as illustrated in Fig. 1(a), which consists of a number of mobile users running computation-intensive applications, a dedicated edge server (ES) deployed at the base station, and a set of third-party fog nodes (FNs) \( \mathcal{N} \) with cardinality of \( |\mathcal{N}| = N \). Through the ES (acting as the central controller), all mobile users can declare requests of offloading resource-hungry computation tasks for alleviating their own computational burdens. These offloaded tasks may be either executed by the processing unit of the ES or delegated to third-party FNs in set \( \mathcal{N} \), depending on the workload distribution decision. The wireless data transmissions between computation offloading requesters (i.e., mobile users) and servers (i.e., FNs/ES) are achieved by D2D/cellular overlaying communications, where each server occupies an independent pre-allocated channel. In this paper, since our emphasis is on the management of fog computing with strategic computing speed control, the computation offloading and transmission scheduling at requester ends are omitted, and we refer interested readers to our previous work [13] for details of this process.

Let the aggregate arrival of computation offloading requests (initiated by all mobile users) at the ES be approximated as a Poisson process with an average rate \( \lambda_{\text{total}} \) [18], [19], [24]. For each offloading request, the ES can decide to either self-process it or delegate it to one of the third-party FNs through the dispatcher. Denote the admission rates of offloaded computation tasks (workloads) to the processing unit of the ES and the dispatcher by \( \lambda_E \) and \( \lambda_F \), respectively. Obviously, we have \( \lambda_{\text{total}} = \lambda_E + \lambda_F \). Furthermore, the dispatcher distributes \( \lambda_F \) to FNs in set \( \mathcal{N} \), i.e., \( \lambda_F = \lambda_1 + \lambda_2 + \ldots + \lambda_N \), where \( \lambda_i \) indicates the admission rate of the offloaded computation workload delegated to FN \( i \in \mathcal{N} \). Meanwhile, each third-party FN

![Fig. 1](image_url)

**Fig. 1:** A queueing model of the fog computing system.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ES )</td>
<td>dedicated edge server deployed at the base station</td>
</tr>
<tr>
<td>( \mathcal{N} )</td>
<td>set of third-party fog nodes</td>
</tr>
<tr>
<td>( \lambda_{\text{total}} )</td>
<td>aggregate arrival rate of offloading requests</td>
</tr>
<tr>
<td>( \lambda_E )</td>
<td>admission rate of computation workload on the ES</td>
</tr>
<tr>
<td>( \lambda_F )</td>
<td>admission rate of computation workload for delegation</td>
</tr>
<tr>
<td>( \lambda_{FN} )</td>
<td>admission rate of offloaded tasks on FN ( i )</td>
</tr>
<tr>
<td>( \lambda_i )</td>
<td>admission rate of FN ( i )'s own subscribed tasks</td>
</tr>
<tr>
<td>( \mu_i^i )</td>
<td>normalized rate of transmitting tasks to server ( i )</td>
</tr>
<tr>
<td>( \bar{\mu}_i^i )</td>
<td>normalized computing capacity of server ( i )</td>
</tr>
<tr>
<td>( \mu^F )</td>
<td>normalized computing speed of FN on offloaded tasks</td>
</tr>
<tr>
<td>( V )</td>
<td>service reward for handing each offloading request</td>
</tr>
<tr>
<td>( T_{\text{tran}} )</td>
<td>transmission delay of each task assigned to server ( i )</td>
</tr>
<tr>
<td>( T_{\text{proc}} )</td>
<td>processing delay of each task assigned to server ( i )</td>
</tr>
<tr>
<td>( \beta )</td>
<td>average cost per unit of delay for each offloaded task</td>
</tr>
<tr>
<td>( \theta_i )</td>
<td>average cost per unit of delay for each FN ( i )'s own task delegation price to FN ( i ) for processing each task</td>
</tr>
<tr>
<td>( \xi_i, \alpha_i )</td>
<td>scaling parameters of the power model of server ( i )</td>
</tr>
<tr>
<td>( U_i )</td>
<td>expected utility of the ES</td>
</tr>
<tr>
<td>( C_{\text{del}}^i )</td>
<td>expected utility of each FN ( i )</td>
</tr>
<tr>
<td>( C_{\text{total}}^i )</td>
<td>delegation cost suffered by the ES</td>
</tr>
<tr>
<td>( \mathcal{G} )</td>
<td>total cost of FN ( i ) in providing computation services</td>
</tr>
</tbody>
</table>

**Table 1:** Important notations in this paper
$i \in \mathcal{N}$ is also responsible for dealing with its subscribed computation tasks that arrive following a Poisson process with an average admission rate $\Lambda_i$ (to fulfill its own application purpose). For any computation task assigned to either the ES or a FN that cannot be transmitted or processed immediately due to limitations of communication or computing capabilities, this task will be temporarily stored in the associated buffer (the buffer of the mobile user if it is waiting to be transmitted or the buffer of the server if it is waiting to be processed). Therefore, the impact of dynamics (in task arrivals, transmission and processing) can be characterized/measured by the buffering delay which will be analyzed in Section 3.2. Here, we do not consider buffer overflow. Then, the studied fog computing framework can be modeled as a queueing system, as shown in Fig. 1(b).

By considering heterogeneities of different computation tasks in communication and computing requirements, we define that each task has a random packet size of $Z^t$ with a finite mean $E[Z^t]$ and demands a random number of CPU cycles $Z^c$ with a finite mean $E[Z^c]$. It is worth noting that our proposed analytical framework allows $Z^t$ and $Z^c$ to be correlated in nature, while their inherent relationship is not explicitly described for conciseness. In addition, we define the expected transmission rate2 (measured by bits per second) of mobile users in offloading computation tasks to server $i \in \mathcal{ES} \cup \mathcal{N}$ as $r_i$, and the computing speed (measured by CPU cycles per second) of server $i \in \mathcal{ES} \cup \mathcal{N}$ as $s_i$. Then, the average transmission time of one task offloaded to any server $i \in \mathcal{ES} \cup \mathcal{N}$ can be calculated as $E[Z^t]/r_i = 1/\mu^t_i$, where $\mu^t_i$ denotes the normalized transmission rate of offloading any task to server $i$. Following the similar process, we can define the average computation time of one task on any server $i \in \mathcal{ES} \cup \mathcal{N}$ as $E[Z^c]/s_i = 1/\mu^c_i$, where $\mu^c_i$ stands for the the normalized computing speed of server $i$, and further characterize the computing capacity (i.e., the maximum computing speed) of each server $i$ in a normalized form as $\mu^t_i \in \mathcal{ES} \cup \mathcal{N}$. For the ES, since its computing resource (characterized by $\mu^c_E$) can be exclusively utilized for serving offloaded tasks with admission rate $\lambda_E$, we have $\mu^t_E = \mu^c_E$. In contrast, the computing resource of each FN $i \in \mathcal{N}$ (characterized by $\mu^c_i$) has to be divided into two parts, i.e., $\mu^c_i = \mu^t_i + \mu^c_i$, which are the normalized computing speeds reserved by FN $i \in \mathcal{N}$ for serving its own subscribed computation tasks and the offloaded ones with admission rates $\lambda_i$ and $\lambda_i$, respectively.

Note that similar to [5], [10], this paper particularly focuses on the transmission and processing of computation tasks, and ignore potential overheads in feeding back computing outcomes and those required in control signalling. This is because the data sizes of computing outcomes and control signals are commonly negligible compared to those of computation tasks (each of which may consist of various system settings and programs).

3.2 Problem Formulation

In this subsection, we study the utility functions and strategic decisions of the ES and third-party FNs in the considered fog computing system.

As the central controller, the ES is required to collect all computation offloading requests from mobile users (with an average rate $\lambda_{total}$), self-process the computation workload $\lambda_E$ out of $\lambda_{total}$ on its own processing unit, and delegate the rest of the computation workload $\lambda_F = \lambda_{total} - \lambda_E$ to all FNs in set $\mathcal{N}$ with a workload distribution $\lambda_i, \forall i \in \mathcal{N}$, such that $\lambda_F = \sum_{i \in \mathcal{N}} \lambda_i$. Taking into account all costs and gains, the net utility of the ES can be expressed as

$$U_{ES} = V \lambda_{total} - C_{delay} - C_{proc} - C_{dele}$$

where $V$ is the average service reward (charged from mobile users) for handing each computation offloading request, so that $V\lambda_{total}$ indicates the long-term expected reward gained by the ES; $C_{delay}$, $C_{proc}$ and $C_{dele}$ represent the delay cost incurred by temporarily storing computation tasks in associated buffers (i.e., the penalty for service delay), the processing cost consumed in executing computation tasks (i.e., the energy cost in edge computing) and the delegation cost for seeking help from third-party FNs (i.e., the delegation price/reward paid to all FNs), respectively.

Specifically, the delay cost $C_{delay}$ depends on the computation workload $\lambda_E$ that the ES decides to undertake by itself and the expected end-to-end system delay, which consists of transmission delay $T_{trans}^E(\lambda_E)$ and processing delay $T_{proc}^E(\lambda_E)$, that each offloaded computation task would experience if it is assigned to the ES. Mathematically, $C_{delay}^{ES}$ can be defined as

$$C_{delay}^{ES} = \lambda_E \beta \left( E[T_{trans}^E(\lambda_E) + T_{proc}^E(\lambda_E)] \right)$$

where $\beta$ denotes the average cost per unit of delay for each offloaded task. It is worth noting that offloading tasks to any FN (rather than the ES) may also induce a similar delay cost term. However, such cost can be counted as a reward reduction on this chosen FN (which will be discussed later), and thus is cancelled out in $U_{ES}$.

The processing cost $C_{proc}$ refers to the energy consumption cost of the ES in edge computing. According to the widely adopted power dissipation model for CMOS circuits [28], $C_{proc}^{ES}$ can be calculated as

$$C_{proc}^{ES} = \frac{\lambda_E}{\mu^c_E} \xi_E \cdot (\mu^t_E)^{\alpha_E}$$

where $\lambda_E/\mu^c_E$ is the busy time4 (or the utilization ratio) of the ES in processing offloaded computation tasks; $\mu^t_E$ is the

1. In practical fog computing systems, buffer overflow barely happens because i) the storages of current mobile devices are over hundred gigabytes so that their buffer sizes can be considerably large; and ii) the data traffic may be well balanced among many FNs, leading to a relatively small individual workload.

2. Since each server has been pre-allocated with an independent channel for receiving computation tasks from mobile users via D2D communications, transmission rates of different mobile users in offloading tasks to a server may be heterogeneous due to their different power gains on this server’s associated channel. Thus, the transmission rate of offloading tasks to any server can be modeled as a random variable following the convention in [25], [26].
computing speed of the ES; $\xi_E > 0$ and $\alpha_E \geq 2$ are constant scaling parameters of the power model [28], [29].

Besides, in order to successfully delegate part of the computation workload, i.e., $\lambda_F = \sum_{i \in \mathcal{N}} \lambda_i$, to third-party FNs, the ES has to pay a unit delegation price/reward $\pi_i$ to every FN $i \in \mathcal{N}$ for compensating its cost in processing each task. Thus, the total delegation cost $C_{\text{dele}}$ suffered by the ES is

$$C_{\text{dele}}^{\text{ES}} = \sum_{i \in \mathcal{N}} \lambda_i \pi_i. \tag{4}$$

Naturally, $\pi_i$ should be determined within the range of $[0, V]$ (meaning that the delegation price should be lower than or equal to the original service reward), so as to guarantee a non-negative benefit obtained by the ES from delegating workload $\lambda_i$ to each FN $i \in \mathcal{N}$ (i.e., $\lambda_i(V - \pi_i) \geq 0$). The reason of having this constraint is that all performance analyses are based on long-term measurements, and both $\pi_i$ and $\lambda_i$ are average values over a certain period. Because of this, if $\pi_i > V$ for any FN $i \in \mathcal{N}$, the ES will certainly suffer a utility loss from the workload delegation to this FN $i$ in a long run. Therefore, FNs, which cannot bring non-negative benefit to the ES, will not be enabled in the workload distribution scheduling due to the potential risk. Following the conventions in the literature [17], [30], this trivial case can be ignored without loss of generality.

Substituting (2)–(4) into (1), the utility of the ES can now be rewritten as

$$U_{\text{ES}} = V\lambda_{\text{total}} - \lambda_{\text{ES}} \beta \left( \mathbb{E}[T_{\text{total}}^{\text{trans}}(\lambda_{\text{ES}})] + T_{\text{total}}^{\text{proc}}(\lambda_{\text{ES}}) \right) - \frac{\lambda_{\text{ES}}}{\mu_{\text{ES}}} \xi_{\text{ES}} \cdot (\mu_{\text{ES}}^{c})^{\alpha_{i}} - \sum_{i \in \mathcal{N}} \lambda_i \pi_i. \tag{5}$$

For each third-party FN $i \in \mathcal{N}$, its utility consists of a delegation price/reward gained from the ES, i.e., $\lambda_i \pi_i$, and a cost $C_{\text{total}}^{\text{i}}(\mu_{\text{i}}^{c}, \lambda_i)$ depending on its computing speed $\mu_{\text{i}}^{c}$ and assigned workload $\lambda_i$:

$$U_i = \lambda_i \pi_i - C_{\text{total}}^{\text{i}}(\mu_{\text{i}}^{c}, \lambda_i), \quad \forall i \in \mathcal{N}. \tag{6}$$

Here, $C_{\text{total}}^{\text{i}}(\mu_{\text{i}}^{c}, \lambda_i)$ includes delay and processing costs for both offloaded computation tasks and its own subscribed ones, i.e.,

$$C_{\text{total}}^{\text{i}}(\mu_{\text{i}}^{c}, \lambda_i) = \lambda_i \beta \left( \mathbb{E}[T_{\text{i}}^{\text{trans}}(\lambda_i)] + T_{\text{i}}^{\text{proc}}(\mu_{\text{i}}^{c}, \lambda_i) \right) + \frac{\lambda_i}{\mu_{\text{i}}^{c}} \xi_{\text{i}}(\mu_{\text{i}}^{c})^{\alpha_{i}} + \lambda_i \theta_i \mathbb{E}[T_{\text{i}}^{\text{proc}}(\mu_{\text{i}}^{e} - \mu_{\text{i}}^{c}, \lambda_i)], \tag{7}$$

where $\lambda_i \beta \left( \mathbb{E}[T_{\text{i}}^{\text{trans}}(\lambda_i)] + T_{\text{i}}^{\text{proc}}(\mu_{\text{i}}^{c}, \lambda_i) \right)$ is the cost induced by both transmission and processing delays to the assigned offloaded tasks; $\lambda_i / \mu_{\text{i}}^{c} \xi_{\text{i}}(\mu_{\text{i}}^{c})^{\alpha_{i}}$ is the energy cost introduced by processing offloaded tasks; $\theta_i$ denotes the cost per unit of delay for each FN $i$’s own task, so that $\lambda_i \theta_i \mathbb{E}[T_{\text{i}}^{\text{proc}}(\mu_{\text{i}}^{e} - \mu_{\text{i}}^{c}, \lambda_i)]$ calculates the total cost due to delaying its own subscribed tasks$^5$. Note that other reward and cost terms related to each FN’s own subscribed computation tasks, such as the reward gained from subscribers, transmission delay cost and energy cost, are omitted in our formulation, because they can be considered as constants and have no effects on management decisions.

From the system management perspective, the ES will aim to maximize its utility $U_{\text{ES}}$ by carefully determining i) the admission rate of computation workload for delegation to all FNs, i.e., $\lambda_F$ (or the admission rate of computation workload on the ES, i.e., $\lambda_E$); ii) the distribution of $\lambda_F$ to each FN in set $\mathcal{N}$, i.e., $\lambda_i, \forall i \in \mathcal{N}$; and iii) the unit delegation price $\pi_i$ for each FN $i \in \mathcal{N}$. Given that the computing speed provided by FN $i \in \mathcal{N}$ for offloaded computation tasks is $\mu_{\text{i}}^{c}$, the utility maximization problem at the ES can be formulated as

$$[\mathcal{MP}]: \arg\max_{\{\lambda_F, (\lambda_i, \pi_i)\}_{i \in \mathcal{N}}} U_{\text{ES}} \tag{8}$$

$$s.t., \quad 0 \leq \lambda_F \leq \lambda_{\text{total}}, \quad \lambda_i \geq 0, \quad \forall i \in \mathcal{N}, \tag{9}$$

$$0 \leq \pi_i \leq V, \quad \forall i \in \mathcal{N}, \tag{10}$$

$$\lambda_F + \lambda_E = \lambda_{\text{total}}, \tag{11}$$

$$\sum_{i \in \mathcal{N}} \lambda_i = \lambda_F, \tag{12}$$

$$\sum_{i \in \mathcal{N}} \lambda_i \pi_i - C_{\text{total}}^{\text{i}}(\mu_{\text{i}}^{c}, \lambda_i) \geq 0, \quad \forall i \in \mathcal{N}, \tag{13}$$

where constraints (9) and (10) define ranges of all decision variables; constraints (11) and (12) illustrate limitations on the workload distribution; and constraint (13) states that the resulting utility of each third-party FN should be non-negative, so as to encourage them to participate in the fog computing. It is worth noting that since the decision of delegation price $\{\pi_i\}_{i \in \mathcal{N}}$ is included in utility functions of both the ES and FNs, to solve problem $[\mathcal{MP}]$, we are required to further investigate how $\{\pi_i\}_{i \in \mathcal{N}}$ would affect the inherent strategic behaviors of all FNs.

As a smart and rational individual, each FN $i \in \mathcal{N}$ can strategically allocate its computing resource or control the computing speeds for serving its subscribed computation tasks and the offloaded ones (denoted by $\mu_{\text{i}}^{e} - \mu_{\text{i}}^{c}$ and $\mu_{\text{i}}^{c}$, respectively) with the objective of maximizing its own utility $U_i$, as shown in (6). For achieving this goal, each FN $i$ will compete with other FNs in set $\mathcal{N} \setminus \{i\}$ by adjusting $\mu_{\text{i}}^{c}$ to potentially increase the delegation reward $\lambda_i \pi_i$ gained from serving the offloaded computation tasks (e.g., a larger amount of workload $\lambda_i$ may be granted to FN $i$ if it provides a better computing service with shorter delays than the others). However, a larger $\lambda_i$ in turn increases the delay cost of offloaded tasks. In addition, increasing $\mu_{\text{i}}^{c}$ is equivalent to decreasing $\mu_{\text{i}}^{e} - \mu_{\text{i}}^{c}$ which will certainly increase the delay cost of its own subscribed tasks. Therefore, all FNs have to strategically determine their computing speeds by considering the impacts of their competitions/interactions in the workload distribution and the complicated tradeoffs in the queuing performance of fog computing. Obviously, this leads to a noncooperative queueing game, which can be formally defined as

$$\mathcal{G} = \{\mathcal{N}, \mathcal{B}, \{U_i(\mu_{\text{i}}^{c}|\mu_{\text{-i}}^{c}, \lambda_i, \pi_i)\}_{i \in \mathcal{N}}\}, \tag{14}$$

where FNs in set $\mathcal{N}$ act as players in the game; $\mathcal{B}$ signifies the strategy set of FNs’ computing speeds $\mu_{\text{i}}^{c}, \forall i \in \mathcal{N}$; and $U_i(\mu_{\text{i}}^{c}|\mu_{\text{-i}}^{c}, \lambda_i, \pi_i)$ is the utility of FN $i \in \mathcal{N}$ in terms of i) its own strategy $\mu_{\text{i}}^{c}$ given that the strategy of all other FNs is $\mu_{\text{-i}}^{c} = \{\mu_{\text{j}}^{c}\}_{j \neq i}$, ii) the assigned workload $\lambda_i$, and iii) the unit delegation price $\pi_i$. 

---

$^5$ One can also impose a stringent constraint such that $\mu_{\text{i}}^{e} - \mu_{\text{i}}^{c} \geq \mu_{\text{i}}^{c}$, where $\mu_{\text{i}}^{c}$ defines the amount of computing resource that has to be reserved for subscribers with certain delay requirements. Obviously, under this setting, the proposed analytical framework still holds, and the only modification that we need to make is to re-define the available computing capacity of each FN $i \in \mathcal{N}$ as $\mu_{\text{i}}^{c} - \mu_{\text{i}}^{e}$ instead of $\mu_{\text{i}}^{c}$. 

Solving the formulated system management problem $[MP]$ together with the underlying queueing game $G$ is very challenging because i) $[MP]$ is an optimization problem upon the queueing game $G$; ii) the outcome of game $G$ highly depends on the endogenous delay performance of each FN $i \in N$, and the delay is a function of the FN’s strategy $\mu_i$ and the ES’s workload distribution decision $\lambda$, with implicit relationships; and iii) $\{\pi_i\}_{i \in N}$ is not a simple decision vector and may need to be in a specific function form to guarantee the satisfaction of constraint (13). In the following section, by jointly analyzing the queueing and game performances, we will propose an efficient algorithm to derive the corresponding solution (i.e., the system management decisions of the ES and the strategic computing speed of each FN).

4 Queueing Game Approach and Analyses with Different Delegation Price Settings

In this section, we first consider a special case of the problem in which the unit delegation price determined by the ES for all FNs is a constant, and then extend it to a more general case with the utility-dependent delegation price setting.

4.1 Workload Distribution and Computing Speed Control Based on a Constant Delegation Price

Let the unit delegation price $\pi_i$ for any third-party FN $i \in N$ be a constant satisfying constraint (10). For example, we can set $\pi_i, \forall i \in N$, as its maximum possible value $\bar{V}$, i.e., the average service reward gained by the ES for each computation offloading request. This means that the ES directly transfers all associated service rewards collected from mobile users to FNs who help to undertake offloaded computation tasks, and thus the total delegation cost $C_{ES}^{dele}$ suffered by the ES is

$$C_{ES}^{dele} = \sum_{i \in N} \lambda_i \pi_i = \sum_{i \in N} \lambda_i V = \lambda F V. \quad (15)$$

Substituting (15) into (5) and recalling that $\lambda_{total} - \lambda_F = \lambda_E$, the utility of the ES becomes

$$U_{ES} = V\lambda_E - \lambda_E \beta \left[ E[T_{E}^{\text{tran}}(\lambda_E) + T_{E}^{\text{proc}}(\lambda_E)] \right] - \frac{\lambda_E \xi \lambda_E}{\mu_E} (\mu_E)^{\alpha - 1}. \quad (16)$$

Since $U_{ES}$ shown in (16) is now a function of $\lambda_E$ only, the optimal $\lambda_E$ can be determined by

$$[CP1]: \lambda_E^* = \arg\max_{0 \leq \lambda_E \leq \lambda_{total}} U_{ES}. \quad (17)$$

Note that $[CP1]$ is a subproblem of $[MP]$ and is independent from the workload distribution decision $\lambda_i, \forall i \in N$. Taking the second-order derivative of $U_{ES}$ with respect to $\lambda_E$, we have

$$\frac{\partial^2 U_{ES}}{\partial \lambda_E^2} = -\beta \left( \frac{\partial E[T_{E}^{\text{tran}}(\lambda_E)]}{\partial \lambda_E} + \frac{\partial^2 E[T_{E}^{\text{proc}}(\lambda_E)]}{\partial \lambda_E^2} \right) < 0, \quad (18)$$

6. Here, we take $\pi_i = V, \forall i \in N$, as an example to show the analysis procedure, while any other constant values of $\pi_i$ that satisfies constraint (10) can also be employed with slight mathematical modifications.

where $E[T_{E}^{\text{tran}}(\lambda_E)] = E[T_{E}^{\text{tran}}(\lambda_E) + T_{E}^{\text{proc}}(\lambda_E)]$ represents the end-to-end system delay experienced by each task assigned to the ES. Inequality (18) holds because both transmission and processing delays are naturally increasing and convex with respect to the arrival rate $\lambda_E$ [31], or in other words, $\frac{\partial E[T_{E}^{\text{tran}}(\lambda_E)]}{\partial \lambda_E} > 0$ and $\frac{\partial^2 E[T_{E}^{\text{proc}}(\lambda_E)]}{\partial \lambda_E^2} > 0$. Thus, $U_{ES}$ is a concave function, and the optimal $\lambda_E$ must exist, which can be calculated according to

$$\lambda_E^* = \min\{\lambda_E| \frac{\partial E[T_{E}^{\text{tran}}(\lambda_E)]}{\partial \lambda_E} = 0; \lambda_{total}\}. \quad (19)$$

Then, the optimal $\lambda_F$ (denoted by $\lambda_F^*$) is directly obtained as

$$\lambda_F^* = \lambda_{total} - \lambda_E^*.$$

Given $\lambda_F^*$, the ES can further determine the distribution of $\lambda_F$ to all FNs in set $N$, i.e., $\lambda_i, \forall i \in N$, based on

$$[CP2]: \arg\min_{\lambda_i \geq 0} \sum_{i \in N} \left( \beta (E[T_{i}^{\text{tran}}(\lambda_i)] + T_{i}^{\text{proc}}(\mu_i(\lambda_i), \lambda_i)) \right) d\lambda_i$$

s.t., $\sum_{i \in N} \lambda_i = \lambda_F^*$, \quad (20)

where $\mu_i(\lambda_i)$ stands for the computing speed provided by FN $i \in N$, which depends on the workload distribution $\lambda_i$. The objective function of this management problem $[CP2]$ is built for the interests of mobile users, i.e., minimizing the average delay cost of each offloaded computation task delegated to FNs, because $U_{ES}$ has already been maximized (i.e., the objective of the original system management problem $[MP]$ has already been achieved) after solving problem $[CP1]$. Recall that smart and rational FNs may compete to maximize their own utilities by strategically controlling their computing speeds, leading to the queueing game $G$, as defined in (14). Therefore, given the solution to $[CP1]$, i.e., $\lambda_E^*$ (or $\lambda_F^*$), our remaining problem is to solve $[CP2]$ by jointly considering the underlying queueing game $G$ for the optimal workload distribution $\lambda_i$ and the equilibrium computing speed $\mu_i^*$ of each FN $i \in N$. To this end, we introduce a load-balancing rule as follows.

Load-balancing rule: Denote the set of FNs selected for undertaking offloaded computation tasks by $N_a \subseteq N$, i.e., FN $i \in N_a$ if and only if $\lambda_i > 0$. It is required that the expected end-to-end system delay of each offloaded task on any FN $i \in N_a$ is the same, i.e., $E[T_{i}^{\text{tran}}(\lambda_i)] + T_{i}^{\text{proc}}(\mu_i(\lambda_i), \lambda_i)) = \phi, \forall i \in N_a$, and the experienced delay must be larger than or equal to $\phi$ if an offloaded task is dispatched to FN $j \in N \setminus N_a$. Mathematically, this rule can be written as

$$\phi = E[T_{i}^{\text{tran}}(\lambda_i)] + T_{i}^{\text{proc}}(\mu_i(\lambda_i), \lambda_i)] \leq E[T_{j}^{\text{tran}}(\lambda_j)] + T_{j}^{\text{proc}}(\mu_j(\lambda_j), \lambda_j), \forall i \in N_a, \forall j \in N, \quad (21)$$

$$\sum_{i \in N_a} \lambda_i = \lambda_F^*. \quad (22)$$

Later, we will prove that applying such rule in queueing game $G$ can result in $\{\lambda_i\}_{i \in N}$ and $\{\mu_i\}_{i \in N}$ converging to an equilibrium solution which meets the objective of $[CP2]$.

Definition 1 (The NE of queueing game $G$ with the load-balancing rule). Under the load-balancing rule, the Nash equilibrium (NE) of queueing game $G$ is defined as a triplet $\{\lambda_i\}_{i \in N}, \{\mu_i\}_{i \in N}, \phi$ satisfying constraints (21), (22) and

$$U_i(\mu_i^*, \phi, \lambda_i) \geq U_i(\mu_i, \phi, \lambda_i), \forall i \in [0, \bar{\mu}], \forall i \in N, \quad (23)$$
which means that the utility of each FN $i \in \mathcal{N}$ is maximized when all FNs determine their strategic computing speeds as $\{\hat{\mu}_i^c\}_{i \in \mathcal{N}}$.

Substituting this NE $\{\hat{\lambda}_i\}_{i \in \mathcal{N}}, \{\hat{\mu}_i^c = \mu_i^c(\hat{\lambda}_i)\}_{i \in \mathcal{N}}, \hat{\phi}$ into (21) and (22), and after some manipulations, we have

$$(\mathbb{E}[T_{i}^{\text{tran}}(\hat{\lambda}_i) + T_{i}^{\text{proc}}(\mu_i^c(\hat{\lambda}_i), \hat{\lambda}_i)] - \hat{\phi})(\lambda_i - \hat{\lambda}_i) \geq 0, \quad \sum_{i \in \mathcal{N}} \hat{\lambda}_i = \lambda^*_F,$$  

(24)

where $\lambda_i^* \geq 0$ represents an arbitrary non-negative value.

**Theorem 1.** Following the load-balancing rule, the NE of queueing game $\mathcal{G}$ satisfies the necessary conditions for the optimality of problem $[CP2]$.

**Proof:** Let $\{\lambda_i^*\}_{i \in \mathcal{N}}$ be the optimal workload distribution decision for $[CP2]$ and $\nu$ be the Lagrange multiplier with respect to its constraint (20). Then, by applying Karush-Kuhn-Tucker (KKT) optimality conditions [32], we have

$$\beta(\mathbb{E}[T_{i}^{\text{tran}}(\lambda_i^*) + T_{i}^{\text{proc}}(\mu_i^c(\lambda_i^*), \lambda_i^*)] + \nu) = 0, \quad \forall i \in \mathcal{N}, \quad \sum_{i \in \mathcal{N}} \lambda_i^* = \lambda^*_F.$$  

(26)

If we further define $\lambda_i^* = \hat{\lambda}_i$ and $-\nu/\beta = \hat{\phi}$, one can easily derive that (26) and (27) are actually equivalent to (24) and (25) with an inequality transformation. This completes the proof.

Next, we show that, under the load-balancing rule, there exists a unique NE in queueing game $\mathcal{G}$. For explicit expressions, hereafter, we illustrate all analyses by assuming that both the task transmission and execution processes are Markovian. Then, the utility function of each FN $i \in \mathcal{N}$, i.e., (6), can be expanded as

$$U_i = \lambda_i \pi_i - \frac{\lambda_i \beta}{\mu_i^c - \lambda_i} - \frac{\lambda_i \beta}{\mu_i^c - \lambda_i} - \frac{\lambda_i \beta}{\mu_i^c - \lambda_i} - \frac{\lambda_i \beta}{\mu_i^c - \lambda_i} - \frac{\lambda_i \beta}{\mu_i^c - \lambda_i} - \frac{\lambda_i \beta}{\mu_i^c - \lambda_i} - \frac{\lambda_i \beta}{\mu_i^c - \lambda_i} - \frac{\lambda_i \beta}{\mu_i^c - \lambda_i}.$$  

(28)

**Theorem 2.** The game $\mathcal{G} = \{\mathcal{N}, \mathcal{B}, \{U_i(\mu_i^c | \mu_i^c, \lambda_i, \pi_i)\}_{i \in \mathcal{N}}\}$ has at least one NE.

**Proof:** Since the strategy of each FN $i \in \mathcal{N}$ is its self-controlled computing speed $\mu_i^c$ for serving the offloaded computation tasks which must be within the range from 0 to its maximum capacity $\bar{\mu}_i$, the strategy set $\mathcal{B}$ of game $\mathcal{G}$ can be expressed in the form of Cartesian product as

$$\mathcal{B} = \prod_{i=1}^{\mathcal{N}} [0, \bar{\mu}_i] \subset \mathbb{R}^\mathcal{N},$$  

(29)

which is obviously nonempty, compact and convex.

Taking the first-order derivative of $U_i$, $\forall i \in \mathcal{N}$, in (28) with respect to $\mu_i^c$ yields

$$\frac{\partial U_i}{\partial \mu_i^c} = \frac{\lambda_i \beta}{(\mu_i^c - \lambda_i)^2} - \frac{\lambda_i \beta}{(\mu_i^c - \lambda_i)^2} - \frac{\lambda_i \beta}{(\mu_i^c - \lambda_i)^2} - \frac{\lambda_i \beta}{(\mu_i^c - \lambda_i)^2} - \frac{\lambda_i \beta}{(\mu_i^c - \lambda_i)^2} - \frac{\lambda_i \beta}{(\mu_i^c - \lambda_i)^2}.$$  

(30)

Then, the second-order derivative of $U_i$ can be derived as

$$\frac{\partial^2 U_i}{\partial (\mu_i^c)^2} = -\frac{2\lambda_i \beta}{(\mu_i^c - \lambda_i)^3} - \frac{2\lambda_i \beta}{(\mu_i^c - \lambda_i)^3} - \frac{2\lambda_i \beta}{(\mu_i^c - \lambda_i)^3} - \frac{2\lambda_i \beta}{(\mu_i^c - \lambda_i)^3}.$$  

(31)

As the queueing stability imposes a strict requirement that the service rate must be larger than the arrival rate, we have

$$\mu_i^c > \lambda_i \text{ and } \mu_i^c - \lambda_i > \Lambda_i.$$  

(32)

Substituting (32) into (31) gives $\frac{\partial^2 U_i}{\partial (\mu_i^c)^2} < 0$, which indicates that $U_i$ is a concave function of $\mu_i^c$.

To sum up, since the strategy set is nonempty, compact and convex, and the utility functions are continuous and concave, there is at least one NE in game $\mathcal{G}$ [33].

**Theorem 3.** Under the load-balancing rule, the NE of queueing game $\mathcal{G}$ is unique.

**Proof:** Let us first explore the relationship between the workload distribution decision $\lambda_i$ and the speed control strategy $\mu_i^c$ of each FN $i \in \mathcal{N}$. From Theorem 2, we can see that, to maximize utility $U_i$, $\mu_i^c(\lambda_i)$ should satisfy $\frac{\partial^2 U_i}{\partial \mu_i^c} = 0$. Based on this, we are able to express $\mu_i^c(\lambda_i)$ in terms of $\lambda_i$. Although such expression may be too long and tedious, taking derivatives of $\mu_i^c(\lambda_i)$ with respect to $\lambda_i$, we must have

$$0 < \frac{\partial \mu_i^c(\lambda_i)}{\partial \lambda_i} \leq 1, \quad \text{and} \quad \frac{\partial^2 \mu_i^c(\lambda_i)}{\partial \lambda_i^2} \geq 0, \quad \forall i \in \mathcal{N},$$  

(33)

and thus $\mu_i^c(\lambda_i)$ is strictly increasing with $\lambda_i$.

Furthermore, from (32) and (33), we can easily prove that

$$\frac{\partial}{\partial \lambda_i} \left[ \frac{1}{\mu_i^c - \lambda_i} \right] = \frac{1}{(\mu_i^c - \lambda_i)^2} + \frac{1}{(\lambda_i - \lambda_i)^2} > 0,$$

This result implies that the optimization problem $[CP2]$ is strictly convex, and the optimal workload distribution decision $\{\lambda_i^*\}_{i \in \mathcal{N}}$ must be unique. Moreover, since $\mu_i^c(\lambda_i)$ is a strictly increasing (monotone) function of $\lambda_i$, the optimal strategy $\mu_i^c$ of each FN $i \in \mathcal{N}$ must also be unique.

In addition, the impact of NE on the queueing model of the fog computing framework can be investigated. For each FN $i \in \mathcal{N}$, define $\mu_i^c(0) = \lim_{\lambda_i \to -\infty} \mu_i^c(\lambda_i)$, so that $f_i(0) = 1/\mu_i^c + 1/\mu_i^c(0)$ represents the “base” end-to-end system delay that any offloaded computation task would experience if it is assigned to this FN. Without loss of generality, assume that FNs in set $\mathcal{F} = \{1, 2, \ldots, N\}$ are ordered in the sequence of $1/\mu_i^c + 1/\mu_i^c(0) \leq 1/\mu_i^c + 1/\mu_i^c(0) \leq \ldots \leq 1/\mu_N^c + 1/\mu_N^c(0)$.

**Theorem 4.** When the NE of queueing game $\mathcal{G}$ is reached, the set of FNs selected for undertaking offloaded computation tasks can be described as $\mathcal{N}_\alpha = \{1, 2, \ldots, \alpha\}$ with a threshold value $n \leq N$ that meets the requirement:

$$\sum_{i=1}^{n} f_i^{-1}(\frac{1}{\mu_i^c} + \frac{1}{\mu_i^c}(0)) < \lambda_N^F \leq \sum_{i=1}^{n} f_i^{-1}(\frac{1}{\mu_i^c} + \frac{1}{\mu_i^c(0)}),$$  

(34)

where $f_i^{-1}(\cdot)$ is the inverse of $f_i(\lambda_i) = \frac{1}{\mu_i^c} + \frac{1}{\mu_i^c(\lambda_i)}$.

**Proof:** From the load-balancing condition of the NE in (24), we can see that

$$\frac{1}{\mu_i^c} + \frac{1}{\mu_i^c(0)} \begin{cases} < \hat{\phi}, & \text{if } \text{FN } i \in \mathcal{N}_\alpha; \\ \geq \hat{\phi}, & \text{otherwise.} \end{cases}$$  

(35)
Algorithm 1: Adaptive Algorithm for Queueing Game \( G \) with a Constant Delegation Price (AACP)

**Output:** The NE of game \( G \), i.e., \( \{\lambda_i\}_{i \in \mathcal{N}} \) and \( \{\mu_i^t\}_{i \in \mathcal{N}} \).

1. Initialize: \( \mathcal{N}_0 = \mathcal{N} \), \( \lambda_i = \frac{\lambda_i^0}{N} \); Define \( \epsilon \) and \( K \) as the adaption rate and the maximum number of iterations; 
2. **for** iteration index \( k = 1 \) to \( K \) **do** 
   3. **for** FN \( i \in \mathcal{N}_k \) **do** 
      4. Adjust its strategic computing speed as \( \mu_i^t = \mu_i^t(\lambda_i) \), where \( \mu_i^t(\lambda_i) \) is the solution to \( \frac{\partial U_i}{\partial \mu_i^t} = 0 \) given \( \lambda_i \); 
      5. Measure the expected end-to-end delay performance as \( f_i(\lambda_i) = \mathbb{E}[T_i^{\text{trans}}(\lambda_i) + T_i^{\text{prop}}(\mu_i^t(\lambda_i), \lambda_i)] = \frac{1}{\mu_i} + \frac{1}{\mu_i^t(\lambda_i) - \lambda_i} \); 
   6. if \( f_i(\lambda_i) \) is not identical for any FN \( i \in \mathcal{N}_k \) then 
      7. Calculate a specific delay threshold as \( \phi = \frac{1}{N} \sum_{i \in \mathcal{N}_k} f_i(\lambda_i) = \frac{1}{N} \sum_{i \in \mathcal{N}_k} \frac{1}{\mu_i^t(\lambda_i) - \lambda_i} \); 
   8. **for** FN \( i \in \mathcal{N}_k \) **do** 
      9. if \( f_i(\lambda_i) > \phi \) then 
         10. Update \( \lambda_i = \lambda_i - \epsilon (f_i(\lambda_i) - \phi) \); 
      11. else 
         12. Update \( \lambda_i = \lambda_i + \epsilon (\phi - f_i(\lambda_i)) \); 
   13. Update \( \mathcal{N}_k = \{i \in \mathcal{N} | \lambda_i > 0\} \); 
   14. **else** 
      15. Determine \( \tilde{\lambda}_i = \lambda_i, \tilde{\mu}_i = \mu_i \) and \( \tilde{\pi}_i = V_i \); 

This further indicates that

\[
\tilde{\lambda}_i = \begin{cases} 
\tilde{\mu}_i - \frac{\mu_i^t}{\phi - 1}, & \text{if } \frac{1}{\mu_i^t} + \frac{1}{\mu_i^t(0)} < \phi, \\
0, & \text{if } \frac{1}{\mu_i^t} + \frac{1}{\mu_i^t(0)} \geq \phi.
\end{cases}
\]

(36)

As \( 1/\mu_i^t + 1/\mu_i^t(0) \) has been defined to be decreasing with the increase of \( i \in \mathcal{N} \), it can be concluded that \( \mathcal{N}_k \) must take the form of \( \mathcal{N}_k = \{1, 2, \ldots, n\} \) with a threshold value \( n \leq N \).

Besides, if \( n \) is the threshold value in \( \mathcal{N}_0 = \{1, 2, \ldots, n\} \), we have

\[
\frac{1}{\mu_n^t} + \frac{1}{\mu_n^t(0)} < \phi \leq \frac{1}{\mu_{n+1}^t} + \frac{1}{\mu_{n+1}^t(0)}.
\]

(37)

Since \( f_i(\lambda_i) \) is obviously an increasing function of \( \lambda_i \), inequality (37) can be equivalently transformed as

\[
\sum_{i=1}^{n} f_i^{-1}(\phi) \leq \sum_{i=1}^{n} f_i^{-1}(\phi) \leq \sum_{i=1}^{N} f_i^{-1}(\phi) \leq \sum_{i=1}^{N} f_i^{-1}(\phi) = \sum_{i=1}^{N} f_i^{-1}(\phi) = \sum_{i=1}^{N} \tilde{\lambda}_i = \lambda_F^t \.
\]

(38)

By substituting \( f_i^{-1}(\phi) = \tilde{\lambda}_i \) and \( \sum_{i=1}^{N} \tilde{\lambda}_i = \lambda_F^t \) (stated by the load-balancing conditions (24) and (25), respectively) into (38), inequality (34) can be derived.

\[\square\]

4.2 Workload Distribution and Computing Speed Control Based on Utility-Dependent Delegation Prices

In the previous subsection, we solved the problem with a constant unit delegation price for all FNs. Now, we extend this result to a more general case, where the delegation prices for FNs depend on their utility functions. It is worth noting that, the consideration of utility-dependent delegation prices raises additional challenges: i) the admission rate of total computation workload for delegation, i.e., \( \lambda_F \), can no longer be obtained independently as that in [CP1] because \( \pi_i, \forall i \in \mathcal{N} \), cannot be cancelled out and will be a decision variable along with \( \lambda_F \) in the utility of the BS; and ii) the workload distribution among FNs, i.e., \( \lambda_i, \forall i \in \mathcal{N} \), has to be determined by taking into account its impact to
the ES rather than for the interests of FNs only. To overcome these difficulties, we propose a new solution algorithm, called adaptive algorithm for utility-dependent delegation prices (namely AASP), based on some preliminary results obtained from Section 4.1 and analyze its performance accordingly.

Following the formulation of FNs’ utility functions in (6), the utility-dependent delegation price \( \pi_i \) for each FN \( i \in N \) can be assumed in a form as

\[
\pi_i = \frac{\omega_i}{\lambda_i} C_i^{total}(\mu^c_i, \lambda_i), \quad \forall i \in N,
\]

where \( \omega_i \geq 1 \) denotes a pre-assigned weight factor of each FN \( i \in N \). With (39), one can easily verify that constraint (13) in problem [MP] must be satisfied, and constraint (10) can also be met by adjusting the weight factor. Moreover, (39) shows that \( \pi_i \) is a function of both \( \lambda_i \) and \( \mu^c_i \) for each FN \( i \in N \), and it is automatically determined once the optimal workload distribution decision \( \lambda_i \) and strategic computing speed \( \mu^c_i \) have been found.

From (5), we can see that the first, second and third terms of \( U_{ES} \) are related to \( \lambda_F \) (or \( \lambda_E \)) only, regardless of any other decisions (i.e., \( \lambda_i, \forall i \in N \)). This implies that all these three terms become constants if \( \lambda_F \) is given. Thus, the ES can choose to carefully tune \( \lambda_F \) in the range of \( [0, \lambda_{total}] \), and for any value of \( \lambda_F \), determine \( \{\lambda_i\}_{i \in N} \) by addressing a relaxed problem of the original [MP]:

\[
[S^P]: \quad \arg \min_{\{\lambda_i \geq 0\}_{i \in N}} \sum_{i \in N} \lambda_i \pi_i = \sum_{i \in N} \omega_i C_i^{total}(\mu^c_i, \lambda_i)
\]

s.t.,

\[
\sum_{i \in N} \lambda_i = \lambda_F,
\]

which is a workload distribution problem with the objective of minimizing the total delegation price suffered by the ES given \( \lambda_F \). Note that since smart and rational FNs will compete to maximize their own utilities by strategically controlling their computing speeds, the resulted queueing game \( G \), as defined in (14), is also implied in [SP]. Hence, next, we aim to solve problem [SP] with the underlying queueing game \( G \).

**Theorem 6.** There always exists a unique optimal solution of \( \{\lambda_i\}_{i \in N} \) to problem [SP].

**Proof:** Recall that, with the assumption of Markovian task transmission and execution processes, the expression of \( C_i^{total}(\mu^c_i, \lambda_i) \) can be expanded as

\[
C_i^{total} = \frac{\lambda_i \beta}{\mu^c_i - \lambda_i} + \frac{\lambda_i \beta}{\mu^c_i - \lambda_i} + \frac{\lambda_i \beta}{\mu^c_i - \mu^c_i - \lambda_i} + \frac{\lambda_i \beta}{\mu^c_i - \xi_i(\mu^c_i)^{\alpha_i}}.
\]

Taking the first-order derivative of \( C_i^{total} \) with respect to \( \lambda_i \) yields

\[
\frac{\partial C_i^{total}}{\partial \lambda_i} = \frac{\beta}{(\mu^c_i - \lambda_i)^2} + \frac{\beta}{(\mu^c_i - \lambda_i)^2} + \frac{\lambda_i \beta}{(\mu^c_i - \mu^c_i - \lambda_i)^2} + \xi_i(\mu^c_i)^{\alpha_i - 1} + (\alpha_i - 1) \lambda_i \xi_i(\mu^c_i)^{\alpha_i - 2} \frac{\partial \mu^c_i}{\partial \lambda_i},
\]

which must be larger than 0 due to (32) and (33) presented in Section 4.1 (and these two inequalities still hold because the utility functions of FNs are unchanged). With some further manipulations, it can also be shown that the second-order derivative of \( C_i^{total} \) with respect to \( \lambda_i \) is larger than 0. In other words, we have

\[
\frac{\partial^2 C_i^{total}}{\partial \lambda_i^2} > 0 \quad \text{and} \quad \frac{\partial C_i^{total}}{\partial \lambda_i} > 0.
\]

This indicates that \( C_i^{total} \) is a strictly increasing and convex function of \( \lambda_i \). Therefore, the optimization problem \( [SP] \) is a convex one with a nonempty and compact feasible set, so that there always exists a unique optimal solution.

Let the optimal solution of problem \( [SP] \) be \( \{\lambda_i^*\}_{i \in N} \). Then, applying KKT optimality conditions to \( [SP] \) gives

\[
\left( \omega_i \frac{\partial C_i^{total}}{\partial \lambda_i} / \partial \lambda_i - \varphi \right) (\lambda_i^* - \lambda_i^*) \geq 0, \quad \forall i \in N,
\]

\[
\sum_{i \in N} \lambda_i^* = \lambda_F, \quad \forall \lambda_i^* \geq 0, \quad \forall \lambda_i^* \geq 0,
\]

where \( - \varphi \) is the corresponding Lagrange multiplier, and \( \lambda_i^* \geq 0 \) represents an arbitrary non-negative value. Observe that conditions (45) and (46) are analogous to (24) and (25), and \( \partial C_i^{total} / \partial \lambda_i \) is increasing with the workload distribution decision \( \lambda_i \) as proved in (44). These suggest a modified load-balancing rule among FNs as follows.

**Modified load-balancing rule:** Denote the set of FNs for undertaking offloaded computation tasks by \( N_\omega \subset N, \text{i.e., } i \in N_\omega \text{ if and only if } \lambda_i^* > 0 \). It is required that the marginal delegation price paid to any FN \( i \in N_\omega \) in completing each offloaded task is the same, i.e., \( \omega_i \frac{\partial C_i^{total}}{\partial \lambda_i} = \varphi, \forall i \in N_\omega \); and such marginal delegation price must be higher than or equal to \( \varphi \) if an offloaded task is dispatched to FN \( j \in N \backslash N_\omega \). Mathematically, this rule can be written as

\[
\varphi = \omega_i \frac{\partial C_i^{total}}{\partial \lambda_i} / \partial \lambda_i \leq \omega_j \frac{\partial C_j^{total}}{\partial \lambda_j}, \forall i \in N_\omega, \forall j \in N, \quad (47)
\]

\[
\sum_{i \in N_\omega} \lambda_i^* = \lambda_F, \quad (48)
\]

Obviously, since such modified load-balancing rule is specifically designed to meet the optimality conditions (45) and (46), adopting it in queueing game \( G \) will lead to an equilibrium solution which achieves the objective of [SP].

**Definition 2 (The NE of queueing game \( G \) with the modified load-balancing rule):** Under the modified load-balancing rule, the Nash equilibrium (NE) of queueing game \( G \) is defined as a triplet \( \{(\lambda_i^*), \{\mu^c_i\}_{i \in N}, \varphi \} \) satisfying constraints (47), (48) and

\[
U_i(\mu^c_i | \mu^c_{-i}, \lambda_i) \geq U_i(\mu^c_i^* | \mu^c_{-i}, \lambda_i), \quad \forall \mu^c_i \in [0, \mu^c_i^*], \forall i \in N, \quad (49)
\]

which means that the utility of all FNs are maximized when the NE is reached.

Considering that game behaviors/strategic interactions among FNs in the scenario with utility-dependent delegation prices are similar to those in the scenario with a constant unit delegation price, except that the system objectives (i.e., the objectives of [CP2] and [SP]) are different.

7. Although different forms of the delegation price are employed in Sections 4.1 and 4.2, since these payments are made from the ES to FNs after the computation services, FNs are unaware of this information in advance, and hence their ex-ante utility functions remain unchanged.

8. This is because, as clarified previously, the strategy set \( B \) and ex-ante utility functions \( \{U_i\}_{i \in N} \) of FNs in \( N \) are independent from the delegation price settings.
Algorithm 2: Adaptive Algorithm for Queueing Game \( G \) with Utility-Dependent Delegation Prices (AASP)

Output: \( \lambda_F^*, \{ \lambda_i^*, \mu_i^*, \pi_i^* \}_{i \in \mathcal{N}} \).

1. Initialize: Let the tuning index \( \tau = 1 \); Define \( \epsilon \) and \( K \) as the adaption rate and the maximum number of iterations; 
2. for \( \lambda_F = 0 \) to \( \lambda_{total} \) do
   3. Initialize \( \mathcal{N}_0 = \mathcal{N} \) and increase tuning index \( \tau = \tau + 1 \);
   4. Determine the workload distribution as \( \lambda_i = \frac{\lambda}{\mathcal{N}}, \forall i \in \mathcal{N} \);
   5. for iteration index \( k = 1 \) to \( K \) do
      6. for \( FN \in \mathcal{N}_0 \) do
         7. Adjust its computing speed as \( \mu_i^* = \mu_i^*(\lambda_i) \), where \( \mu_i^*(\lambda_i) \) is the solution to \( \frac{\partial F}{\partial \lambda_i} = 0 \) given \( \lambda_i \); 
         8. Estimate the marginal delegation price for each task as \( g_i(\lambda_i) = \omega_i \frac{\partial C_{total}}{\partial \lambda_i} \), where \( \frac{\partial C_{total}}{\partial \lambda_i} \) is obtained according to (43); 
         9. if \( g_i(\lambda_i) \) is not identical for any \( FN \in \mathcal{N}_0 \) then
            10. Calculate a specific threshold value as \( \varphi = \frac{1}{\mathcal{N}} \sum_{i \in \mathcal{N}_0} g_i(\lambda_i) = \frac{1}{\mathcal{N}} \sum_{i \in \mathcal{N}_0} \omega_i \frac{\partial C_{total}}{\partial \lambda_i} \); 
            11. for \( FN \in \mathcal{N}_0 \) do
               12. if \( g_i(\lambda_i) > \varphi \) then
                  13. Update \( \lambda_i = \lambda_i - \epsilon (g_i(\lambda_i) - \varphi) \); 
               else
                  14. Update \( \lambda_i = \lambda_i + \epsilon (\varphi - g_i(\lambda_i)) \); 
               end
            end
            15. Update \( \mathcal{N}_0 = \{ i \in \mathcal{N} | \lambda_i > 0 \} \);
            else
               16. Determine \( \hat{\lambda}_i(\tau) = \lambda_i, \hat{\mu}_i^*(\tau) = \mu_i, \forall i \in \mathcal{N} \);
               17. Substitute \( \hat{\lambda}_i(\tau), \hat{\mu}_i^*(\tau) \) into (39) so as to get \( \pi_i(\tau) \); 
            end
         end
         19. Calculate and compare \( U_{ES}(\tau) \) with \( U_{ES}(\tau - 1) \); 
         20. if \( U_{ES}(\tau) \leq U_{ES}(\tau - 1) \) then
         end
   end
   21. Determine \( \lambda_F^* = \lambda_F^*, \lambda_i^* = \hat{\lambda}_i(\tau), \mu_i^* = \hat{\mu}_i^*(\tau), \pi_i^* = \pi_i(\tau) \).

queueing game \( G \) integrated modified load-balancing rule shares exactly the same structure with the one discussed in Section 4.1, and thus the following theorem is intuitive.

**Theorem 7.** Under the modified load-balancing rule, there exists a unique NE in queueing game \( G \).

*Proof:* This theorem can be easily proved referring to the same proof procedures for Theorems 2 and 3.

Based on the above results and the similar idea of Algorithm 1 (AACP), a new adaptive algorithm for finding the optimally tuned \( \lambda_F^* \) and the NE of game \( G \) with utility-dependent delegation prices given \( \lambda_F^* \), i.e., \( \{ \lambda_i^*, \mu_i^*, \pi_i^* \}_{i \in \mathcal{N}} \), is proposed, which is detailedly summarized as shown in Algorithm 2 (AASP).

The major difference between Algorithms 1 (AACP) and 2 (AASP) is that an iteration of tuning \( \lambda_F \) from 0 to \( \lambda_{total} \) for maximizing \( U_{ES} \) is added besides the queueing game \( G \). Note that \( U_{ES} \) is a concave function of \( \lambda_F \) (which can be easily verified by (18) and (44)), and thus the tuning process can be early terminated as soon as \( U_{ES} \) is no longer increasing with \( \lambda_F \). In addition, since the modified load-balancing rule is applied in Algorithm 2 (AASP), the individual decisions \( \lambda_i \) and \( \mu_i^* \) of each FN \( i \in \mathcal{N} \) are adapted till converging to the NE by repeatedly checking the optimality conditions that have to be satisfied by the marginal delegation price instead of the delay performance as considered in Algorithm 1 (AACP). By running Algorithm 2 (AASP), the optimal solution, denoted by \( (\lambda_F^*, \{ \lambda_i^*, \mu_i^*, \pi_i^* \}_{i \in \mathcal{N}}) \), to the problem of optimal workload distribution and strategic computing speed control based on the utility-dependent delegation price can be produced.

**Theorem 8.** The computational complexity of the proposed Algorithm 2 (AASP) is \( O\left( \frac{\lambda_{total}}{\lambda_F} N \log 1/\delta \right) \), where \( N \) is the total number of FNs, \( \delta \) denotes the precision of the condition in line 9 (i.e., such condition fails if and only if \( \max_{i \in \mathcal{N}_0} g_i(\lambda_i) - \min_{i \in \mathcal{N}_0} g_i(\lambda_i) \leq \delta \), and \( \Delta \lambda_F \) is the step-size in tuning \( \lambda_F \).

*Proof:* Intuitively, the maximum number of steps for tuning \( \lambda_F \) to reach \( \lambda_F^* \) is \( \frac{\log 1/\delta}{\Delta \lambda_F} \). Moreover, given any value of \( \lambda_F \), the total number of iterations needed for queueing game \( G \) with modified load-balancing rule to converge is the same as that of Algorithm 1 (AACP), i.e., \( 2N \log 1/\delta \). Hence, the computational complexity of Algorithm 2 (AASP) is \( O\left( \frac{\lambda_{total}}{\lambda_F} 2N \log 1/\delta \right) = O\left( \frac{\lambda_{total}}{\lambda_F} N \log 1/\delta \right) \). To theoretically evaluate the performance difference of employing the utility-dependent delegation price setting and the constant one in solving the original problem \( [M \mathcal{P}] \), we further compare the outcome of Algorithm 2 (AASP), i.e., \( (\lambda_F^*, \{ \lambda_i^*, \mu_i^*, \pi_i^* \}_{i \in \mathcal{N}}) \), with that of Algorithm 1 (AACP), i.e., \( (\lambda_F^*, \{ \lambda_i^*, \mu_i^*, \pi_i^* \}_{i \in \mathcal{N}}) \), in terms of the utility of the ES \( U_{ES} \). Since \( \lambda_F^* \) is same for both algorithms (as they optimize \( \lambda_F \) with the same objective), after cancelling all terms with respect to \( \lambda_F^* \) only in (5), the comparison in terms of \( U_{ES} \) can be simplified to the comparison in terms of the ES’s delegation cost \( C_{dele}^{ES} \).

**Theorem 9.** Define \( [C_{dele}^{ES}]^{CP} \) and \( [C_{dele}^{ES}]^{SP} \) as the delegation cost suffered by the ES in the system under the constant delegation price and the utility-dependent delegation price, respectively. Then, we have

\[
\eta = \frac{[C_{dele}^{ES}]^{CP}}{[C_{dele}^{ES}]^{SP}} \geq \left( \frac{V}{\lambda_F^*} \right)^{\alpha-1} \max_{i \in \mathcal{N}} \omega_i \xi_i, \quad N, \tag{50}
\]

where \( \alpha = \max_{i \in \mathcal{N}} \alpha_i \) and \( \eta \) is a ratio reflecting the superiority of AASP over AACP in maximizing \( U_{ES} \) or minimizing \( C_{dele}^{ES} \).

*Proof:* For the system with the constant delegation price, i.e., \( \pi_i = V, \forall i \in \mathcal{N} \), the delegation cost of the ES is

\[
[C_{dele}^{ES}]^{CP} = \sum_{i \in \mathcal{N}} \lambda_i \pi_i = V \sum_{i \in \mathcal{N}} \lambda_i = V \lambda_F^*. \tag{51}
\]

For the system with utility-dependent delegation prices, i.e., \( \pi_i^* = \omega_i C_{total}^{\pi*} \), the delegation cost of the ES is

\[
[C_{dele}^{ES}]^{SP} = \sum_{i \in \mathcal{N}} \lambda_i^* \pi_i^* = \sum_{i \in \mathcal{N}} \omega_i C_{total}^{\pi_i^*}. \tag{52}
\]

9. As explained when introducing the modified load-balancing rule, if the algorithm converges to the NE, the KKT conditions (i.e., (45) and (46)) are satisfied automatically, and hence problem \( [SP] \) is solved to the optimum.
By the heavy traffic approximation [35], i.e., $\lambda_i^* / \mu_i^* \to 1$, we can observe that

$$C_i^{total} \approx \xi_i (\lambda_i^*)^{\alpha_i} \text{ and } \lambda_i^* \approx \frac{1}{\sum_{j \in N} \omega_{ij}} \lambda_i^* \beta_i.$$ \hspace{1cm} (53)

Substituting (53) into (52) yields

$$|C_{ES}^{dele}|_{SP} = \sum_{i \in N} \omega_i C_i^{total} \leq \frac{1}{\sum_{i \in N} \omega_{ii}} \lambda_i^*$$

$$\leq \frac{1}{\max_{i \in N} \omega_{ii}} \lambda_i^* = \frac{\max_{i \in N} \omega_{ii}}{N} (\lambda_i^*)^\alpha.$$ \hspace{1cm} (54)

With the help of (51) and (54), it can be shown that

$$\eta \geq (V \lambda_i^*) \left( \frac{\max_{i \in N} \omega_{ii} \xi_i}{N} (\lambda_i^*)^\alpha \right)$$

$$= \frac{V}{(\lambda_i^*)^{\alpha-1} \max_{i \in N} \omega_{ii} \xi_i} N,$$ \hspace{1cm} (55)

which completes the proof.

Theorem 9 indicates that $\eta$ is scaled with the number of FNs in the system, i.e., $N$. Therefore, we can conclude that AASP (with the utility-dependent delegation price) is more suitable to be used for fog computing in large-scale networks (due to its significant performance improvement), while AACP (with the constant delegation price) may be more preferable in small-scale ones (because it is relatively easier to be implemented, while the performance degradation may be marginal when $N$ is small). The reason for this is that, by adopting the utility-dependent delegation price setting rather than the constant one, $\pi_i$ for each FN $i \in N$ does not need to be fixed as its upper bound $V$, and thus AASP can reduce the total delegation cost suffered by the ES compared to AACP.

## 5 Simulation Results

In this section, simulations are conducted to numerically evaluate the performance of the proposed queueing game based management framework and algorithms for workload distribution and strategic computing speed control in the fog computing system. All results are obtained by taking averages over 100 runs with various system parameters.

### 5.1 Simulation Settings

Consider a MATLAB-based simulation environment for a fog computing system with a dedicated ES that has a computing capacity of 15 GHz and $N = 10$ third-party FNs, each of which has a computing capacity randomly chosen from 4 to 6 GHz. Following the configurations of fog computing assisted mobile gaming applications [12], [36], the mean values of the packet size and the required CPU cycles of each computation task are approximated as 500 Kb and 1000 Megacycles, respectively. According to the 4G network characteristics [37], the wireless transmission rate of offloading each computation task to each server in fog computing is determined randomly from 2 to 5 Mbps. In addition, assume that the aggregate arrival rate of computation offloading requests collected at the ES, i.e., $\lambda_{total}$, is 20 per second, and the arrival rate of each FN’s own subscribed computation tasks, i.e., $\Lambda_i, \forall i \in N$, is selected within $[0.5, 1]$ per second. Furthermore, let $V = 1, \beta = 1$, $\theta_i = 1, \omega_i = [1, 2], \forall i \in N$, and define the power scaling parameters as $\alpha_i = 3, \xi_i = [0, 1], \forall i \in N \cup ES$. Similar settings have also been employed in the literature [10], [13], [18], [38], [39]. Note that some parameters may be varied for different evaluation purposes.

### 5.2 Performance Evaluations

Figs. 2 and 3 examine the convergence of the proposed adaptive algorithms, i.e., AACP and AASP, for solving the formulated queueing game $G$ with constant and utility-dependent delegation prices, respectively. From both figures, we can observe that the workload distribution decision on any FN $i \in N$, i.e., $\lambda_i$, converges very quickly, although the convergence speed of the proposed algorithms may be slightly slower than other existing iterative algorithms, such as multi-armed bandit algorithms [40], [41], which ignores the consideration of guaranteeing individual utilities. Besides, due to the heterogeneities among FNs in terms of their computing capacities, power costs and own subscribed...
computation burdens, the optimal workload distribution decisions on different FNs eventually converge to different values, while the sum of them equals the optimal value of $\lambda_F^*$ (which can be calculated based on (19) for AACP or generated by fine tuning for AASP). These results match the theoretical analyses in Section 4, and also demonstrate the efficiency of AACP and AASP. Moreover, it is shown by comparing Figs. 2 and 3 that the outcome of AASP (i.e., workloads assigned on FNs) is more evenly distributed than that of AACP. This is because AASP takes the FNs’ energy consumption costs into account in the objective function of the system management problem via the utility-dependent delegation prices, and uneven workload distributions may result in larger energy costs at FNs with larger computing speeds, which offsets the benefits of workload balancing in delay cost reductions.

In Fig. 4, we investigate the equilibrium performance of queueing game $\mathcal{G}$ (with the constant delegation price setting as an example) by illustrating the utility of a FN $i \in \mathcal{N}$ with different computing speeds for serving offloaded computation tasks (reflected by the ratio of its capacity, i.e., $\mu_i^c/\bar{\mu}_i^c$). In the considered system, each FN can strategically adjust its computing speed to potentially increase its individual utility. From the trend of curves in Fig. 4, we can see that the utility of FN $i$ first increases with its computing speed $\mu_i^c$. This is because a larger $\mu_i^c$ indicates a better computing service for offloaded computation tasks, and $\lambda_i$ will increase accordingly (implied by the objective function of problem [CP2]), so that more service rewards can be obtained. However, after a certain point, i.e., the NE of queueing game $\mathcal{G}$ computed by Algorithm 1, since the delay cost becomes dominant (resulted from the limited computing resource reserved for its own subscribed computation tasks and the increasingly heavier workload of offloaded ones), the utility decreases. Obviously, in order to maximize the individual utility, the optimal strategy of each FN is to set its computing speed following the NE of queueing game $\mathcal{G}$.

Fig. 5 shows the performance of the proposed queueing game based management framework (with the AACP algorithm) in terms of the average delay cost of offloaded computation tasks on FNs, i.e., the objective of [CP2]. Here, a proportional workload distribution scheme based on computing capacities of FNs and a uniform workload distribution among all FNs are simulated as benchmarks. Intuitively, a larger capacity of the ES lead to a smaller $\lambda_F$ (or a larger $\lambda_E$ as more offloaded computation tasks will be processed by the ES), and thus the average delay cost on FNs decreases. Besides, from this figure, we can see that the uniform workload distribution scheme has the worst performance because it completely ignores the heterogeneities among FNs. In contrast, the proportional workload distribution scheme outperforms the uniform one due to the consideration of FNs’ heterogeneous computing capacities. Moreover, the proposed AACP achieves the best performance in Fig. 5 (with the delay cost reduced by 20.97% and 31.45% on average compared to the proportional and uniform workload distribution schemes, respectively). This is because FNs’ heterogeneities in both their computing capacities and their own subscribed computation burdens are considered. In other words, the workload distribution decision derived by the proposed AACP can well balance gains and costs on all FNs, so that their computing capacities are fully exploited and the average delay cost of each computation task on FNs is minimized.

Fig. 6 illustrates the performance of the proposed queueing game based management framework (with both AACP and AASP algorithms) in terms of the total delegation price suffered by the ES, i.e., $C_{\text{ES}}^{\text{dele}}$. It is shown that, with the increase of $N$ (the number of FNs), $C_{\text{ES}}^{\text{dele}}$ maintains as a constant for AACP, while decreasing for AASP. This is because, for AACP, $C_{\text{ES}}^{\text{dele}} = \lambda_F V$ does not vary with $N$ (as $\lambda_F$ is determined by problem [CP1] which is independent from FNs, and $V$ is a pre-determined constant). However, for AASP, $C_{\text{ES}}^{\text{dele}} = \sum_{i \in \mathcal{N}} \lambda_i \pi_i$ and the individual delegation price $\pi_i, \forall i \in \mathcal{N}$, decreases when the competition among FNs becomes more intense (resulted by the increase of $N$). This observation matches the analysis in Theorem 9 that the gap between AASP and AACP in $C_{\text{ES}}^{\text{dele}}$ (or the performance improvement of AASP over AACP) is scaled with $N$. It is worth noting that such performance improvement is not
In Figs. 7 and 8, the superiority of applying the proposed queueing game based adaptive algorithms (i.e., AACP and AASP) in the fog computing system with strategic computing speed control is demonstrated. For comparison purpose, two existing algorithms are also simulated, i.e., the energy-efficient optimal workload distribution scheme (EEWD) [18] and the edge computing scheme without the fog assistance (ECNF) [10]. EEWD manages the workload distributions for fog-cloud computing with the aim of minimizing the overall energy consumption, and ECNF optimizes the joint resource allocations for mobile edge computing without enabling third-party fog nodes.

Fig. 7 first compares the targeted objective of the original system management problem $[MP]$, i.e., $U_{ES}$, achieved by different algorithms (i.e., AACP, AASP, EEWD and ECNF) with respect to the number of FNs $N$. It can be observed that, since a larger $N$ implies more candidate servers with potentially low energy consumptions and/or more vacant resources in providing supplementary computing services, leading to a higher cost efficiency of the whole system, $U_{ES}$ increases with $N$ for all algorithms, except for ECNF which always has the worst performance (as it does not exploit any fog assistance). Besides, we can see that the proposed AACP outperforms EEWD because, unlike EEWD which only minimizes the energy consumption, AACP considers a joint delay and energy cost minimization on the ES. Moreover, it is shown that the proposed AASP can result in an even larger $U_{ES}$ than AACP. This is because AASP adopts the utility-dependent delegation price setting (rather than the constant one) which further takes into account the delay and energy costs on all FNs for reducing the total delegation cost suffered by the ES.

We then compare different algorithms (i.e., AACP, AASP, EEWD and ECNF) with respect to $\beta$ (the cost per unit of delay for offloaded computation tasks) in Fig. 8, and similar observations as in Fig. 7 can be obtained. It is worth noting that, since the delay performances of both EEWD and ECNF are constant regardless of $\beta$, $U_{ES}$ decreases linearly with the increase of $\beta$ for these two existing algorithms (as the slopes of their decreasing trends maintain as constant). In contrast, $U_{ES}$ decreases more gently for the proposed AASP and AACP because, with the increase of $\beta$, the workload distributions will be rebalanced for possibly enhancing the delay performance for offloaded computation tasks at the expenses of more costs in energy consumptions and higher penalties from delaying FNs’ own computation tasks.

6 CONCLUSION

In this paper, the management of the fog computing system with strategic computing speed control has been studied. To characterize i) the dynamic nature of the system (i.e., random task arrival, transmission and processing), ii) the computation workload distribution among different servers (the dedicated edge server and third-party fog nodes), and iii) the objective of minimizing the system management cost on the edge server (including its processing, delay and delegation costs) while at the same time maximizing the individual utility of each strategic fog node, a queueing game model is formulated and analyzed. Adaptive algorithms
(i.e., AACP and AASP) are proposed for the constant and utility-dependent delegation price/reward settings, respectively, to derive the optimal workload distribution decision and equilibrium computing speed. Theoretical analyses and simulation results examine the efficiency of the proposed solution algorithms, and show that the delay performance and the management profit can be significantly enhanced compared to the counterparts.

**Acknowledgments**

This work was supported in part by National Natural Science Foundation of China (NSFC) under Grants 62002164, 61801215, 61701230 and 62071230, and Natural Science Foundation of Jiangsu Province under Grant BK20170805, and in part by Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant, CRD Grant and Concordia University PERFORM Research Chair program. A preliminary version [42] of this paper has been presented at IEEE Globecom 2019.

**References**


Jun Cai (M’04-SM’14) received the Ph.D. degree from the University of Waterloo, ON, Canada, in 2004. From June 2004 to April 2006, he was with McMaster University, Canada, as a Natural Sciences and Engineering Research Council of Canada (NSERC) Postdoctoral Fellow. From July 2006 to December 2018, he has been with the Department of Electrical and Computer Engineering, University of Manitoba, Canada, where he was a full Professor and the NSERC Industrial Research Chair. Since January 2019, he has joined the Department of Electrical and Computer Engineering, Concordia University, Canada, as a full Professor and the PERFORM Centre Research Chair. His current research interests include edge/fog computing, eHealth, radio resource management in wireless communication networks, and performance analysis. Dr. Cai served as the Technical Program Committee (TPC) Co-Chair for IEEE GreenCom 2018; Track/Symposium TPC Co-Chair for the IEEE VTC-Fall 2019, IEEE CCECE 2017, IEEE VTC-Fall 2012, IEEE Globecom 2010, and IWoC 2008; the Publicity Co-Chair for IWoC 2010, 2011, 2013, 2014, 2015, 2017, 2020; and the Registration Chair for QShine 2005. He also served on the editorial board of IEEE Internet of Things Journal, IET Communications, and Wireless Communications and Mobile Computing. He received the Best Paper Award from Chinacom in 2013, the Rh Award for outstanding contributions to research in applied sciences in 2012 from the University of Manitoba, and the Outstanding Service Award from IEEE Globecom 2010.

Kun Zhu (M’16) received the Ph.D. degree from School of Computer Engineering, Nanyang Technological University, Singapore, in 2012. He was a research fellow with the Wireless Communications Networks and Services Research Group in University of Manitoba, Canada, from 2012 to 2015. He is currently a Professor in the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA), and Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China. He is also a Jiangsu specially appointed professor. His research interests include resource allocation in 5G, wireless virtualization, and self-organizing networks. He has published more than fifty technical papers and has served as TPC for several conferences. He won several research awards including IEEE WCNC 2019 Best paper awards, ACM China rising star chapter award.

Ran Wang (M’18) is currently an Associate professor at College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA), and Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, P.R. China. He received his B.E. in Electronic and Information Engineering from Honors School, Harbin Institute of Technology (HIT), P.R. China in July 2011 and Ph.D. in Computer Science and Engineering from Nanyang Technological University (NTU), Singapore in April 2016. He was a research fellow with the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore from October 2015 to August 2016. He has authored or coauthored over 50 papers in top-tier journals and conferences. He received the Nanyang Engineering Doctoral Scholarship (NEDS) Award, Singapore and innovative and entrepreneurial Ph.D. Award of Jiangsu Province, China in 2011 and 2017, respectively. His current research interests include intelligent management and control in smart grid, network performance analysis and internet of electric vehicles, etc.

Changyan Yi (S’16-M’18) is currently a Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA), and is also affiliated with the Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China. He received the M.Sc. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Manitoba, MB, Canada, in 2014 and 2018, respectively. From September 2018 to August 2019, he worked as a research associate in University of Manitoba, MB, Canada. He was awarded Changfeng Scholar of NUAA in 2018, Chinese Government Award for Outstanding Students Abroad in 2017, University of Manitoba Graduate Fellowship (UMGF) for 2015-2018, and IEEE ComSoc Student Travel Grant for IEEE Globecom 2016. His research interests include mechanism design, game theory, queuing theory and their applications in various wireless networks including edge/fog computing, IoT, 5G and beyond.