An Online Optimization Framework for Distributed Fog Network Formation with Minimal Latency

Gilsoo Lee*, Walid Saad*, and Mehdi Bennis†‡

* Wireless@VT, Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA,
Emails: {gilsoolee,walids}@vt.edu.

† Centre for Wireless Communications, University of Oulu, Finland,

‡Department of Computer Engineering, Kyung Hee University, Yongin 446-701, Korea, Email: bennis@ee.oulu.fi.

Abstract

Fog computing is emerging as a promising paradigm to perform distributed, low-latency computation by jointly exploiting the radio and computing resources of end-user devices and cloud servers. However, the dynamic and distributed formation of local fog networks is highly challenging due to the unpredictable arrival and departure of neighboring fog nodes. Therefore, a given fog node must properly select a set of neighboring nodes and intelligently offload its computational tasks to this set of neighboring fog nodes and the cloud in order to achieve low-latency transmission and computation. In this paper, the problem of fog network formation and task distribution is jointly investigated while considering a hybrid fog-cloud architecture. The goal is to minimize the maximum computational latency by enabling a given fog node to form a suitable fog network and optimize the task distribution, under uncertainty on the arrival process of neighboring fog nodes. To solve this problem, a novel online optimization framework is proposed in which the neighboring nodes are selected by using a threshold-based online algorithm that uses a target competitive ratio, defined as the ratio between the latency of the online algorithm and the offline optimal latency. The proposed framework repeatedly updates its target competitive ratio and optimizes the distribution of the fog node's computational tasks in order to minimize latency. Simulation results show that the proposed framework can successfully select a set of neighboring nodes while reducing latency by up to 19.25% compared to a baseline approach based on the well-known online secretary framework. The results also show how, using the proposed framework, the computational tasks can be properly offloaded between the fog network and a remote cloud server in different network settings.

Index Terms

Fog Computing, Online Optimization, Online Resource Scheduling, Network Formation.

A preliminary conference version [1] of this work was presented at IEEE ICC 2017.

I. Introduction

The Internet of Things (IoT) is expected to connect over 50 billion things worldwide, by 2020 [2]–[4]. To meet the low-latency requirement of task computation for the IoT devices, relying on conventional, remote cloud solutions may not be suitable due to the high end-to-end transmission latency of the cloud [5]. Therefore, to reduce the transmission latency, the local proximity of IoT devices can be exploited for offloading computational tasks, in a distributed manner. Such local computational offload gives rise to the emerging paradigm of *fog computing* [6]. Fog computing allows overcoming the limitations of centralized cloud computation by enabling distributed, low-latency computation at the network edge, for supporting various wireless and IoT applications [7]. The advantages of the fog architecture comes from the transfer of some of the network functions to the network edge. Indeed, significant amounts of data can be stored, controlled, and computed over fog networks that can be configured and managed by end-user nodes [5]. Within the fog paradigm, computational tasks can be intelligently allocated between the fog nodes and the cloud to meet computational and latency requirements [8]. To implement the fog paradigm, a three-layer network architecture is typically needed to manage sensor, fog, and cloud layers [7]. When the computing tasks are offloaded from the sensor layer to the fog and cloud layers, fog computing faces a number of challenges such as fog network formation and radio/computing resource allocation [9]. In particular, it is challenging for fog nodes to dynamically form and maintain a fog network that they can use for offloading their task. This challenge is exacerbated by the fact that fog computing devices are inherently mobile and will join/leave a network sporadically [10]. Moreover, to efficiently use the computing resource pool of the fog network, novel resource management schemes for the hybrid fog-cloud network architecture are needed [11].

To reap the benefits of fog networks, many architectural and operational challenges must be addressed [12]–[25]. A number of approaches for fog network formation are investigated in [12]–[16]. To configure a fog network, the authors in [12] propose the use of a device-to-device (D2D)-based network that can efficiently support networking between a fog node and a group of sensors. Also, to enable connectivity for fog computing, the work in [13] reviews D2D techniques that can be used for reliable wireless communications among highly mobile nodes. The work in [14] proposes a framework for vehicular fog computing in which fog servers can form a distributed vehicular network for content distribution. In [15], the authors study a message exchange procedure to form a local network for resource sharing between the neighboring fog

nodes. The work in [16] introduces a method to form a hybrid fog architecture in the context of transportation and drone-based networks.

Once a fog network is formed, the next step is to share resources and tasks among fog nodes as studied in [17]–[25]. For instance, the work in [17] investigates the problem of scheduling tasks over heterogeneous cloud servers in different scenarios in which multiple users can offload their tasks to the cloud and fog layers. The work in [18] studies the joint optimization of radio and computing resources using a game-theoretic approach in which mobile cloud service providers can decide to cooperate in resource pooling. Meanwhile, in [19], the authors propose a task allocation approach that minimizes the overall task completion time by using a multidimensional auction and finding the best time interval between multiple auctions to reduce unnecessary time overheads. The authors in [20] study a latency minimization problem to allocate the computational resources of the mobile-edge servers. Moreover, the authors in [21] study the delay minimization problem in fog and cloud-assisted networks under heterogeneous delay considerations. Moreover, the work in [22] investigates the problem of minimizing the aggregate cloud fronthaul and wireless transmission latency. In [23], a task scheduling algorithm is proposed to jointly optimize the radio and computing resources to reduce the users' energy consumption while satisfying delay constraints. The problem of optimizing power consumption is also considered in [24] subject to delay constraint using a queueing-theoretic delay model at the cloud. Moreover, the work in [25] studies the power consumption minimization problem in an online scenario subject to uncertain task arrivals. Last, but not least, the work in [26], studies how tasks can be predicted and proactively scheduled.

In all of these existing fog network formation and task scheduling works in fog networks [14]–[24], it is generally assumed that information on the formation of the fog network is completely known to all nodes. However, in practice, the fog network can be spontaneously initiated by a fog node when other neighboring fog nodes start to dynamically join or leave the network. Hence, the presence of a neighboring fog node to which one can offload tasks is unpredictable. Indeed, it is challenging for a fog node to know when and where another fog node will arrive. Thus, there exists an inherent *uncertainty* stemming from the unknown locations and availability of fog nodes. Further, most of the existing works [14], [15], [19]–[23] typically assume a simple transmission or computational latency model for a fog node. In contrast, the use of a queueing-theoretic model for both transmission and computational latency is necessary to capture realistic latency metrics. Consequently, unlike the existing literature [15], [19]–[23] which assumes full

information knowledge for fog network formation and relies on simple delay models, our goal is to design an *online approach* to enable an on-the-fly formation of the fog network, under uncertainty, while minimizing the computational latency given an end-to-end latency model.

The main contribution of this paper is a novel framework for online fog network formation and task distribution in a hybrid fog-cloud network. This framework allows any given fog node to dynamically construct a fog network by selecting the most suitable set of neighboring fog nodes in presence of uncertainty on the arrival order of neighboring fog nodes. The fog node can jointly use its fog network as well as a distant cloud server to compute given tasks. We formulate an online optimization problem whose objective is to minimize the maximum computational latency of all fog nodes by properly selecting the set of fog nodes to which computations will be offloaded while also properly distributing the tasks among those fog nodes and the cloud. To solve this problem without any prior information on the order of future arrivals of fog nodes, we propose an online optimization framework that achieves a target competitive ratio – defined as the ratio between the latency achieved by the proposed algorithm and the optimal latency that can be achieved by an offline algorithm. In the proposed framework, an online algorithm is used to form a fog network when the neighboring nodes arrives in an online manner, the task distribution is optimized among the nodes on the formed network, and the target competitive ratio is repeatedly updated. We show the target competitive ratio can be achieved by iteratively running the proposed algorithm. Simulation results show that the proposed framework can achieve a target competitive ratio of 1.21 in a given simulation scenario. Also, the proposed algorithm can reduce the latency by up to 19.25% compared to the baseline approach that is a modified version of the popular online secretary algorithm [1]. Therefore, the proposed framework is shown to be able to find a suitable competitive ratio that can reduce the latency of fog computing while properly selecting the neighboring fog nodes that have high performance and suitably distributing tasks across fog nodes and a cloud server.

The rest of this paper is organized as follows. In Section II, the system model is presented. We formulate the online problem in Section III. In Section IV, we propose our online optimization framework to solve the problem. In Section V, simulation results are carried out to evaluate the performance of our proposed framework. Conclusions are drawn in Section VI.

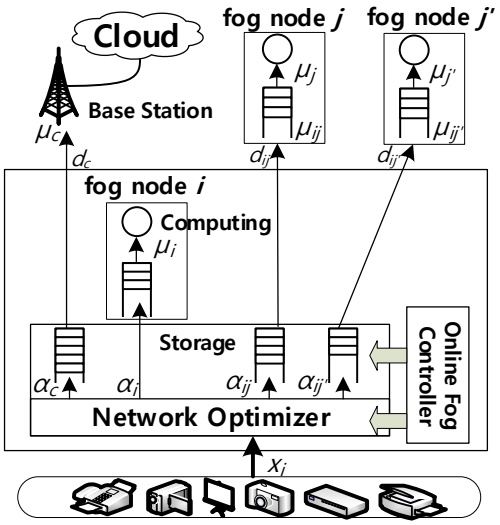

Fig. 1: System model of the fog networking architecture and the cloud.

II. System Model

Consider a fog network consisting of a sensor layer, a fog layer, and a cloud layer as shown in Fig. 1. In this system, the sensor layer includes smart and small-sized IoT sensors with limited computational capability. Here, the sensors offload their task data to the fog and cloud layers for remote distributed computing purposes. The cloud layer can be seen as the conventional cloud computing center. The fog layer refers to the set of IoT devices (also called fog nodes) that can perform fog computing jobs such as storing data and computing tasks. We assume that various kinds of sensors send their task data to a certain fog node i, and the data arrival rate to this node is x_i packets per second where a task packet has a size of K bits. Fog node i assumes the roles of collecting, storing, controlling, and processing the task data from the sensor layer, as is typical in practical fog networking scenarios [5].

In our architecture, for efficient computing, fog node i must cooperate with other neighboring fog nodes and the cloud data center. We consider a network having a set $\mathcal{N}$ of N fog nodes other than fog node i. For a given fog node i, we focus on the fog computing case in which fog node i builds a network with a subset $\mathcal{J} \subset \mathcal{N}$ of J neighboring fog nodes. Also, since the cloud is typically located at a remote location, fog node i must access the cloud via wireless communication links using a cellular base station c.

Once the initial fog node i receives x_i tasks, it assigns a fraction of x_i to other nodes. Then, each node within the considered fog-cloud network will locally compute the assigned fraction of x_i. The fraction of tasks locally computed by fog node i is $\lambda_i(\alpha_i) = \alpha_i x_i$. Then, the task arrival rate offloaded from fog node i to fog node $j \in \mathcal{J}$ is $\lambda_{ij}(\alpha_{ij}) = \alpha_{ij} x_i$. Therefore, the

TABLE I: Summary of notations

i	Index of initial fog node	j	Index of neighboring fog nodes in $\mathcal{J}$
c	Index of cloud	$J = \lvert\mathcal{J}\rvert$	Number of neighboring fog nodes
x_i	Total number of packets from sensors	$\alpha_{k\in\{i,ij,c\}}$	Tasks offloaded toward k
μ_{ij}	Fog transmission service rate from i to j	μ_c	Cloud transmission service rate
μ_i	Computing service rate of fog node i	μ_j	Computing service rate of fog node j
$1/\omega_{k\in\{i,j,c\}}$	Processing speed of node k	n	Arrival order
K	Size of a task packet	γ	Target competitive ratio

task arrival rate processed at the fog layer is $(\alpha_i + \sum_{j\in\mathcal{J}} \alpha_{ij})x_i$. The number of remaining tasks $\lambda_c(\alpha_c) = \alpha_c x_i$ will then be offloaded to the cloud. When fog node i makes a decision on the distribution of all input tasks x_i, the task distribution variables are represented as vector $\boldsymbol{\alpha} = [\alpha_i, \alpha_c, \alpha_{i1}, \ldots, \alpha_{ij}, \ldots, \alpha_{iJ}]$ with $\sum_{j\in\mathcal{J}} \alpha_{ij} + \alpha_i = 1$. Naturally, the total task arrival rate that arrives at fog node i will be equal to the sum of the task arrival rates assigned to all computation nodes in the fog and cloud layers. Also, to model the random arrival of tasks from the sensors to fog node i, the total task arrival rate arriving at fog node i can be modeled by a Poisson process [24]. Since the tasks are offloaded according to α_i, α_c, and α_{ij}, $j \in \mathcal{J}$, the tasks offloaded to each node $\lambda_i(\alpha_i)$, $\lambda_c(\alpha_c)$, and $\lambda_{ij}(\alpha_{ij}), j \in \mathcal{J}$, will also follow a Poisson process if the tasks are scheduled in a round robin fashion [27].

When the tasks arrive from the sensors to fog node i, they are first saved in fog node i's storage, incurring a waiting delay before they are transmitted and distributed to other nodes (fog or cloud). This additional delay pertains to the transmission from fog node i to c or j and can be modeled using a *transmission queue*. Moreover, when the tasks arrive at the destination, the latency required to perform the actual computations will be captured by a *computation queue*. In Fig. 1, we show examples of both queue types. For instance, for transmission queues, fog node i must maintain transmission queues for each fog node j and the cloud c. For computation, each fog node has a computation queue. To model the transmission queue, the tasks are transmitted to fog node j over a wireless channel. Then, the service rate (in packets per second) can be defined by

$$\mu_{ij} = \frac{W_l}{K} \log_2 \left(1 + \frac{g_{ij} P_{\text{tx},i}}{W_l N_0}\right), \tag{1}$$

where g_{ij} is the channel gain between fog nodes i and j with d_{ij} being the distance between them. If $d_{ij} \leq 1$ m, $g_{ij} \triangleq \beta_1$, and, if $d_{ij} > 1$ m, $g_{ij} \triangleq \beta_1 d_{ij}^{-\beta_2}$ where β_1 and β_2 are, respectively, the path loss exponent and path loss constant. Also, $P_{\text{tx},i}$ is the transmission power of fog node i and N_0 is the noise power spectral density. The bandwidth per node is given by W_l where

$l = 1$ and 2 indicate, respectively, two types of bandwidth allocation schemes: equal allocation and cloud-centric allocation. For *equal bandwidth allocation*, all nodes in the network will be assigned equal bandwidth, i.e.,$W_1 = \frac{B}{J+1}$ where the total bandwidth B is equally shared by $J + 1$ nodes that include J neighboring fog nodes and the connection to the cloud via the base station. For the *cloud-centric bandwidth allocation*, the bandwidth allocated to the cloud is twice that of the bandwidth used by a fog node, i.e., the cloud and the fog node will be assigned the bandwidth $\frac{2B}{J+2}$ and $\frac{B}{J+2}$, respectively.

Since the tasks arrive according to a Poisson process, and the transmission time in (1) is deterministic, the latency of the transmission queue can be modeled as an M/D/1 system [27]:

$$T_j(\lambda_{ij}(\alpha_{ij}), \mu_{ij}) = \frac{\lambda_{ij}(\alpha_{ij})}{2\mu_{ij}(\mu_{ij} - \lambda_{ij}(\alpha_{ij}))} + \frac{1}{\mu_{ij}}, \tag{2}$$

where the first term is the waiting time in the queue at fog node i, and the second term is the transmission delay between fog nodes i and j. Similarly, when the tasks are offloaded to the cloud, the transmission queue delay will be:

$$T_c(\lambda_c(\alpha_c), \mu_c) = \frac{\lambda_c(\alpha_c)}{2\mu_c(\mu_c - \lambda_c(\alpha_c))} + \frac{1}{\mu_c}, \tag{3}$$

where the service rate μ_c between fog node i and cloud c is given by (1) where fog node j is replaced with cloud c.

Next, we define the computation queue. When a fog node needs to compute a task, this task will experience a waiting time in the computation queue of this fog node due to a previous task that is currently being processed. Since a fog node j receives tasks from not only fog node i but also other fog nodes and sensors, the task arrival process can be approximated by a Poisson process by applying the Kleinrock approximation [27]. Therefore, the computation queue can be modeled as an M/D/1 queue and the latency of fog node j's computation will be:

$$S_j(\lambda_{ij}(\alpha_{ij})) = \frac{\lambda_{ij}}{2\mu_j(\mu_j - \lambda_{ij})} + \frac{1}{\mu_j} + \omega_j\lambda_{ij}(\alpha_{ij}), \tag{4}$$

where the first term is the waiting delay in the computation queue, the second term is the delay for fetching the proper application needed to compute the task, and the third term is the processing delay for the task. The delay of this fetching procedure depends on the performance of the node's hardware which is a deterministic constant that determines the service time of the computation queue. In the first and second terms of (4), μ_j is a parameter related to the overall hardware performance of fog node j. $\omega_j\lambda_{ij}(\alpha_{ij})$ is the actual computation time of the task where ω_j is a constant time needed to compute a task. For example, $1/\omega_j$ can be proportional to the CPU clock frequency

of fog node j. For fog node $j \in \mathcal{J}$, it is assumed that the maximum of computing service rate and processing speed are given by $\bar{\mu}_j$ and $1/\underline{\omega}_j$, respectively. This information can be known in advance if the manufacturers of fog devices can provide the hardware performance in the database. Then, when fog node i locally computes its assigned tasks $\lambda_i(\alpha_i)$, the latency will be:

$$S_i(\lambda_i(\alpha_i)) = \frac{\lambda_i(\alpha_i)}{2\mu_i(\mu_i - \lambda_i(\alpha_i))} + \frac{1}{\mu_i} + \omega_i \lambda_i(\alpha_i), \tag{5}$$

where μ_i is the computing service rate of fog node i (dependent on hardware performance) and $\omega_i \lambda_i(\alpha_i)$ is the fog node i's computing time. Since the cloud is equipped with more powerful and faster hardware than the fog node, the waiting time at the computation queue of the cloud can be ignored. This implies that the cloud initiates the computation for the received tasks without queueing delay; thus, we only account for the actual computing delay. As a result, when tasks are computed at the cloud, the computing delay at the cloud will be:

$$S_c(\lambda_c(\alpha_c)) = \omega_c \lambda_c(\alpha_c). \tag{6}$$

In essence, if a task is routed to the cloud c, the latency will be

$$D_c(\lambda_c(\alpha_c), \mu_c) = T_c(\lambda_c(\alpha_c), \mu_c) + S_c(\lambda_c(\alpha_c)). \tag{7}$$

Also, if a task is offloaded to fog node j, then the latency can be defined as the sum of the transmission and computation queueing delays:

$$D_j(\lambda_{ij}(\alpha_{ij}), \mu_{ij}) = T_j(\lambda_{ij}(\alpha_{ij}), \mu_{ij}) + S_j(\lambda_{ij}(\alpha_{ij})). \tag{8}$$

Furthermore, when fog node i computes tasks locally, the latency will be:

$$D_i(\lambda_i(\alpha_i)) = S_i(\lambda_i(\alpha_i)), \tag{9}$$

since no transmission queue is necessary for local computing. Since x_i is constant, $\lambda_{k \in \{i,ij,c\}}$ is only dependent to α_k. From now on, for notational simplicity, $\lambda_k(\alpha_k)$ is presented by λ_k.

For the defined latency functions $D_c(\lambda_c, \mu_c)$, $D_j(\lambda_{ij}, \mu_{ij})$, and $D_i(\lambda_i)$, we observe the following properties. First, the latency of a fog node decreases as the amount of incoming tasks decreases due to the fact that $D_c(\lambda_c, \mu_c)$, $D_j(\lambda_{ij}, \mu_{ij})$, and $D_i(\lambda_i)$ are increasing functions with respect to λ_c, λ_{ij}, and λ_i, respectively. For instance, for fog node j, it can be observed that $\frac{\partial}{\partial \lambda_{ij}} D_j(\lambda_{ij}, \mu_{ij}) = \frac{\partial}{\partial \lambda_{ij}} T_j(\lambda_{ij}, \mu_{ij}) + \frac{\partial}{\partial \lambda_{ij}} S_j(\lambda_{ij}) > 0$. Second, the latency will decrease when the fog node's service rate increases. Due to the computational latency in (2) and (3), $D_c(\lambda_c, \mu_c)$ and $D_j(\lambda_{ij}, \mu_{ij})$ are decreasing functions with respect to μ_c and μ_{ij}, respectively. Third, the latency of a fog node decreases when its hardware is more capable. Due to the computational delays in (4) and (5), $D_j(\lambda_{ij}, \mu_{ij})$ and $D_i(\lambda_i)$ are decreasing functions with respect to μ_{ij} and μ_i, respectively. For

instance, fog node j can have a lower latency when it has a high service rate and computational speed since $\frac{\partial}{\partial \mu_{ij}} D_j(\lambda_{ij}, \mu_{ij}) = \frac{\partial}{\partial \mu_{ij}} T_j(\lambda_{ij}, \mu_{ij}) + \frac{\partial}{\partial \mu_{ij}} S_j(\lambda_{ij}) < 0$. Given this model, in the next section, we formulate an online latency minimization problem to study how a fog network can be formed and how tasks are effectively distributed in the fog network.

III. Problem Formulation

In distributed fog computing, the maximum latency of computing nodes must be minimized for effective distributed computing. To minimize the maximum latency, fog node i must opportunistically find neighboring nodes to form a fog network and carry out the process of task offload. In practice, such neighbors will dynamically join and leave the system. As a result, the initial fog node i will be unable to know a priori whether an adjacent fog node will be available to assist with its computation. Moreover, since the total number of neighboring fog nodes as well as their locations and their available computing resources are unknown and highly unpredictable, optimizing the fog network formation and task distribution processes is challenging. Under such uncertainty, selecting neighboring fog nodes must also account for potential arrival of new fog nodes that can potentially provide a higher data rate and stronger computational capabilities. To cope with the uncertainty of the neighboring fog node arrivals while considering the data rate and computing capability of current and future fog nodes, we introduce an *online optimization scheme* that can handle the problem of fog network formation and task distribution under uncertainty.

We formulate the following online fog network formation and task distribution problem whose goal is to minimize the maximum latency when computing a new task that arrives at fog node i:

$$\min_{\mathcal{J}_{\boldsymbol{\sigma}}, \boldsymbol{\alpha}} \quad \max\left(D_i(\lambda_i),\ D_c(\lambda_c, \mu_c),\ D_{j \in \mathcal{J}_{\boldsymbol{\sigma}}}(\lambda_{ij}, \mu_{ij})\right), \tag{10}$$

$$\text{s.t.} \quad \alpha_i + \alpha_c + \textstyle\sum_{j \in \mathcal{J}} \alpha_{ij} = 1, \tag{11}$$

$$\alpha_i \in [0,1], \alpha_c \in [0,1], \alpha_{ij} \in [0,1], \forall j \in \mathcal{J}_{\boldsymbol{\sigma}} \subset \mathcal{N}_{\boldsymbol{\sigma}}, \tag{12}$$

$$\alpha_i x_i \leq \mu_i,\ \alpha_c x_i \leq \mu_c,\ \alpha_{ij} x_i \leq \mu_j,\ \alpha_{ij} x_i \leq \mu_{ij}, \forall j \in \mathcal{J}_{\boldsymbol{\sigma}}, \tag{13}$$

$$\delta N \leq \bar{\delta}. \tag{14}$$

By using an auxiliary variable u, problem (10) can be transformed into the following:

$$\min_{u} \quad u, \tag{15}$$

$$\text{s.t.} \quad u \geq \max\left(D_i(\lambda_i),\ D_c(\lambda_c, \mu_c),\ D_{j \in \mathcal{J}_{\boldsymbol{\sigma}}}(\lambda_{ij}, \mu_{ij})\right), \tag{16}$$

$$(11), (12), (13), (14),$$

where u is the maximum latency of the fog network. In (15), u represents the largest value among $D_i(\lambda_i), D_c(\lambda_c, \mu_c)$, and $D_j(\lambda_{ij}, \mu_{ij})$. Then, minimizing u is equivalent to minimizing the $\max$ function in (10). Hence, problems (10) and (15) are equivalent.

In constraints (11) and (12), all tasks arriving at fog node i are offloaded among the computing nodes in the fog network. Due to constraint (13), the tasks offloaded to a node cannot exceed the service rate of the computing node. In this problem, the initial fog node i determines the set of neighboring fog nodes $\mathcal{J}_{\boldsymbol{\sigma}}$ when they arrive online and the task distribution vector $\boldsymbol{\alpha}$ so as to minimize the computing latency. Also, the time interval between two consecutive arrivals is assumed to be of duration δ during which fog node i observes each arrival of a neighboring fog node. Therefore, δN is the total time period that fog node i needs to observe N neighboring fog nodes. During the maximum observation period $\bar{\delta}$, fog node i can observe a total number of N arriving fog nodes as captured by constraint (14). As the number of observations increases, fog node i may be able to discover neighboring fog nodes that have higher performance. However, due to constraint (14), fog node i cannot wait to observe an infinite number of neighboring fog nodes. Thus, while observing up to N arriving fog nodes, fog node i should select $J \leq N$ neighboring fog nodes to minimize (10).

In our model, we assume that fog node i does not have any prior information on the neighboring fog nodes given by set $\mathcal{N}_{\boldsymbol{\sigma}}$, and the information about each neighboring node is collected sequentially. Such random arrival sequence is denoted by $\boldsymbol{\sigma} = \sigma_1, \ldots, \sigma_n, \ldots, \sigma_N$ where the arrival of n-th neighboring node is shown as σ_n. For example, a smartphone can choose to become a fog node spontaneously if it decides to share its resources. When fog node i does not have complete information on other fog nodes, the nodes in $\mathcal{N}_{\boldsymbol{\sigma}}$ arrive at fog node i in a random order, and index n can be the arriving order of the neighboring fog nodes. At the arrival of a neighboring node, the arrival order n increases by one; thus, n captures the time order of arrival. At time n, node n can transmit a beacon signal to fog node i to indicate its willingness to join the network of fog node i. The beacon signal can include an information tuple on node n that includes the distance d_{in}, computing service rate μ_n, and the processing speed ω_n. At each time that σ_n is known, e.g., by receiving the beacon signal, fog node i will now have information on these parameters that pertain to node n [28]. Therefore, fog node i only knows the information on the nodes that have previously arrived (as well as the current node).

When fog node i observes σ_n and has knowledge of the n-th neighboring node, it has to make an online decision whether to select node n. If fog node n is chosen by the initial fog node i,

it is indexed by j and included in a set $\mathcal{J}_{\boldsymbol{\sigma}}$ which is a subset of $\mathcal{N}_{\boldsymbol{\sigma}}$. Otherwise, fog node i will no longer be able to select fog node n at a later time period since the latter can join another fog network or terminate its resource sharing offer to fog node i. For notational simplicity, $\mathcal{J}_{\boldsymbol{\sigma}}$ and $\mathcal{N}_{\boldsymbol{\sigma}}$ are hereafter denoted as $\mathcal{J}$ and $\mathcal{N}$, respectively. Fog node i will not be able to have complete information about all N neighboring nodes before all neighboring nodes are selected by fog node i. Therefore, since fog node i cannot know any information on future fog nodes, it is challenging for the initial fog node i to form the fog network by determining $\mathcal{J}$.

Even when the information on each node is known to fog node i, it is difficult to calculate the exact service rates of the fog node in the formulated problem. This is due to the fact that the service rate in (1), that includes the wireless data rate, is a function of the network size J. As the number of nodes sharing their wireless bandwidth increases, the available channel bandwidth per node decreases, thus reducing the data rate. Therefore, unlike the constant parameters μ_i and μ_j, the transmission service rates μ_{ij} and μ_c will vary with the network size. As a consequence, in order to calculate the service rates of neighboring nodes, fog node i has to determine the network size. However, the optimal network size can change by the selection of neighboring nodes. Since network size and node selection are related, it is challenging for fog node i to optimize both network size and the set of neighboring nodes that minimize (15). To solve the online problem, we need to find the set of neighboring fog nodes $\mathcal{J}$ and the task distribution vector $\boldsymbol{\alpha}$ that minimize the maximum latency. Moreover, since there is uncertainty about the future arrival of neighboring nodes as well as their service rates, one has to seek an online, sub-optimal solution that is also robust to uncertainty. In the next section, we propose an online optimization framework that minimizes the value of u in (15).

IV. Task Distribution and Network Formation Algorithms

In our problem, fog node i has to decide whether to admit each neighboring node as the different neighboring nodes arrive in a random order. This problem can be formulated as an online stopping problem. In such problems, such as online secretary problem [29], the goal is to develop online algorithms that enable a company to hire a number of employees, without knowing in which order they will arrive to the interview. To apply such known solutions from the stopping problems, the following assumptions are commonly needed. For instance, the number of hiring positions should be deterministic and given in the problem. Also, the decision maker should be able to decide the preference order among the candidates by comparing the values that

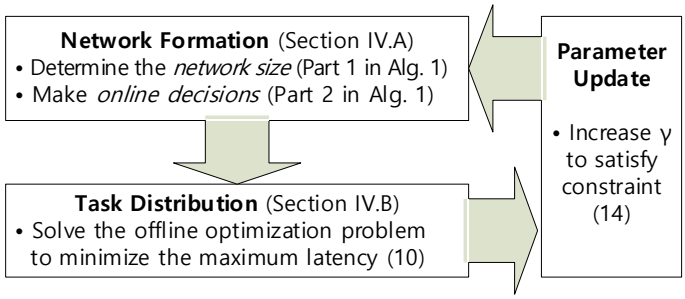

Fig. 2: Online optimization framework for Fog network formation and task distribution.

can be earned by hiring candidates. Under these assumptions, online stopping algorithms can be used to select the best set of candidates in an online manner. In this regard, even though the structures of our fog network formation problem and the secretary problem are similar, the fog network formation case has different assumptions. First, the number of neighboring fog nodes is an optimization variable in our problem. Second, the latency of computing nodes that somewhat maps to the valuation of hiring candidates in the secretary problem is not constant. Moreover, in our problem, each neighboring fog node exhibits two types of latency: transmission latency and computing latency. As a result, it is challenging to define the preference order of the neighboring nodes as done in conventional online stopping problems. To address those challenges, we propose a new online optimization framework that extends existing results from online stopping theory to accommodate the specific challenges of the fog network formation problem.

A. Overview of the Proposed Optimization Framework

Problem (15) has two optimization variables $\mathcal{J}$ and $\boldsymbol{\alpha}$ that constitute the solutions of the network formation and task distribution problems, respectively. To solve (15), fog node i must first optimize the network formation by selecting the neighboring fog nodes, and then decide on its task distribution. This two-step process is required due to the fact that the computing resources of the fog nodes are unknown before the network is formed. The online optimization framework consists of three highly inter-related components as shown in Fig. 2. In the *network formation stage*, an online algorithm is used to find $\mathcal{J}$ by determining the minimal network size and then select the neighboring fog nodes. After $\mathcal{J}$ is determined, the task distribution among the selected nodes is optimized by using an offline optimization method during the *task distribution stage*. Finally, we use a *parameter update stage*, during which the target performance parameter γ that will be used in the next iteration is updated in order to satisfy constraint (14). After repeatedly running three components of our framework, fog node i is able to form a network without any

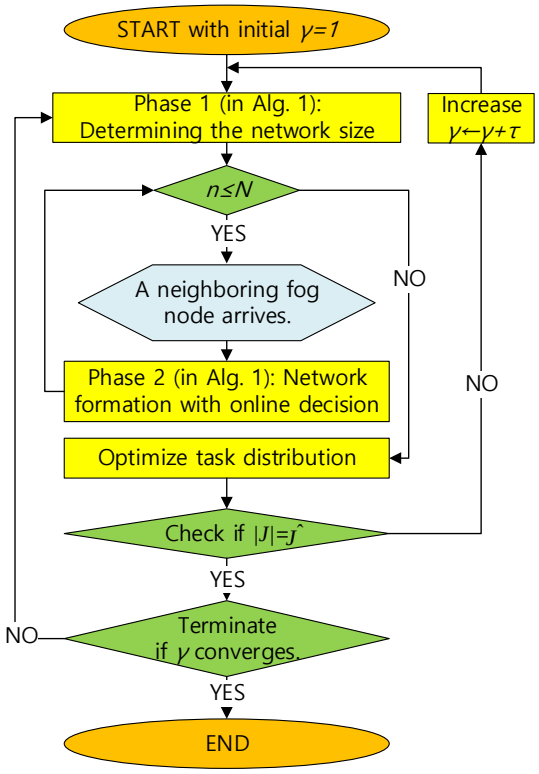

Fig. 3: Flow chart of the proposed framework for fog network formation and task distribution.

prior information on the neighboring nodes and also offload the tasks to the nodes on the fog network. This algorithm is shown to converge in Theorem 3.

The performance of our online optimization framework will be evaluated by using *competitive analysis* [30]. In this analysis, the performance is measured by the *competitive ratio* γ that is defined by

$$1 \leq \frac{\text{ALG}(\boldsymbol{\sigma})}{\text{OPT}(\boldsymbol{\sigma})} \leq \gamma, \tag{17}$$

where $\text{ALG}(\boldsymbol{\sigma})$ denotes the latency achieved by the online algorithm and $\text{OPT}(\boldsymbol{\sigma})$ is the optimal latency achieved by an offline algorithm. If the online algorithm finds the optimal solution, the online algorithm achieves $\gamma = 1$. However, since the online algorithm cannot have complete information, it is challenging to find the optimal solution in an offline setting. Therefore, in an online minimization problem, the online algorithm should be able to achieve γ that is close to one. We use this notion of competitive ratio to design our online optimization framework.

The online optimization framework is summarized in Algorithm 1 and in the flow chart shown in Fig. 3. In the network formation stage, fog node i needs to select the set of neighboring fog nodes with high service rates and processing speeds to achieve a given value of γ. At each iteration, to achieve a target competitive ratio γ, fog node i determines the network size $\hat{J}$ and sequentially observes the arrivals of a total of N neighboring fog nodes while making an

Algorithm 1 Online Fog Network Formation Algorithm

inputs: γ, μ_i, ω_i, ω_c, d_c, $\bar{\mu}_j$, $\bar{\omega}_j$.

Part 1: Calculate $\hat{\lambda}_{ij}$, $\hat{J}$, and $\hat{u}$.

initialize: $J = 0$.

while $\Delta \geq 0$

$J \leftarrow J + 1$.

$\Delta \leftarrow [D_j(\lambda_{ij}, \mu_{ij})]_{|\mathcal{J}|=J} - [D_j(\lambda_{ij}, \mu_{ij})]_{|\mathcal{J}|=J-1}$.

end while

Find $\hat{\lambda}_{ij}$ by optimizing task distribution when $|\mathcal{J}| = J - 1$.

Set $\hat{J} = J - 1$ and $\hat{u} = \left[D_j(\hat{\lambda}_{ij}, \mu_{ij})\right]_{|\mathcal{J}|=J-1}$.

Part 2: Decide $\mathcal{J}$.

while $|\mathcal{J}| < \hat{J}$

if $D_n(\hat{\lambda}_{ij}, \mu_{in}) < \gamma\ \hat{u}$,

$\mathcal{J} \leftarrow \mathcal{J} \cup \{n\}$.

end if

end while

online decision. Then, the online framework checks whether the number of selected neighboring nodes is $\hat{J}$. For a small value of γ, fog node i must find the neighboring nodes having a high computing service rate and processing speed so as to achieve low latency. Therefore, in this case, fog node i must observe a large number of neighboring nodes until $\hat{J}$ neighboring nodes are selected. Hence, N observations may not be sufficient to find $\hat{J}$ neighboring nodes. On the other hand, a large γ can allow the target latency to be less stringent, thus allowing the fog node i to select the neighboring nodes with fewer observations. To find the proper value of γ, the proposed framework iteratively updates γ. For instance, the value of γ can be set to one initially. Then, if a smaller γ cannot be achieved in the network formation stage at that iteration, the value of γ increases by a small constant τ. By repeatedly increasing γ, the proposed framework can find the achievable value of γ. In the next section, we present the details of the proposed online algorithm that exploits the updated value of γ for the network formation stage.

B. Fog Network Formation: Online Approach

In problem (15), the decision on $\mathcal{J}$ faces two primary challenges: how many fog nodes are needed in the network and which fog nodes join the network (at which time). Since the transmission service rates are functions of the wireless bandwidth that can vary with the network

size, the service rates of neighboring fog nodes cannot be calculated without having a fixed network size. Therefore, the proposed algorithm includes two phases as shown in Algorithm 1. The goal of the first phase is to determine the parameters including the network size and the temporal task distribution so that the parameters can be used in the second phase of Algorithm 1. Then, the second phase of Algorithm 1 allows fog node i to make an online decision regarding the selection of an arriving node.

In the first phase of Algorithm 1, the goal is to determine the parameters that will be used in the second phase of Algorithm 1. In the given system model, a neighboring node will be referred to as *ideal* in terms of minimizing the latency in (15) if it has the highest computing service rate $\bar{\mu}_n$, processing speed $1/\bar{\omega}_n$, and channel gain $g_{in} = \beta_1$. Such an *ideal node* is denoted by $\bar{j}$. If a network is formed with nodes having high computing resources, a smaller network size can effectively minimize the latency. When the service rates of the nodes are divided by the smallest network size, the transmission service rates of the nodes also can be maximized, and, hence, the latency can be minimized. In the case in which the ideal nodes construct a network, the minimized latency of (15) is denoted by $\hat{u}$. Also, when the latency is $\hat{u}$, the corresponding network size and task distribution are denoted by $\hat{J}$ and $\{\hat{\lambda}_i, \hat{\lambda}_c, \hat{\lambda}_{ij}\}$, respectively.

First phase: The first phase of Algorithm 1 is used to calculate $\hat{J}$ and $\hat{\lambda}_{ij}$. The latency in (15) decreases as the number of neighboring nodes increases since the computational load per node can be reduced. However, if the number of neighboring nodes becomes too large, the bandwidth per fog node will be smaller yielding lower transmission service rates for the nodes. Consequently, the latency can increase with the number of neighboring nodes, due to these bandwidth limitations. By using the relationship between network size and latency, the first phase of Algorithm 1 searches for $\hat{J}$ while increasing the network size incrementally, one by one. Once the network size $\hat{J}$ that minimizes $\hat{u}$ is found, the tasks offloaded to each ideal node is denoted by $\hat{\lambda}_{ij}$. Therefore, we will have $\hat{J}$, $\hat{u}$, and $\hat{\lambda}_{ij}$ as the outputs from the first phase of Algorithm 1 that will be used in the second phase of Algorithm 1.

Second phase: In the second phase of Algorithm 1, fog node i decides on whether to select each neighboring node or not, by using a threshold-based algorithm. Since comparing two values is a simple operation, a threshold-based algorithm can be executed with low latency (below δ). Our algorithm uses a single threshold so that the latency of each arriving node can be compared with the threshold value. However, before the network formation process is completed, fog node i is not able to know the optimal latency of each node, and, therefore, finding the distribution

of tasks that must be offloaded to each node is not possible. Nonetheless, fog node i must set a threshold before the first neighbor arrives. To this end, fog node i sets this initial threshold by assuming that an equal amount of tasks, $\hat{\lambda}_{ij}$, is offloaded to each one of the $\hat{J}$ neighboring nodes. Thus, in our threshold-based algorithm, the threshold value is compared with the latency that results from offloading $\hat{\lambda}_{ij}$ tasks. For example, when a neighboring node n arrives, the algorithm compares the latency of node n, $D_n(\hat{\lambda}_{ij}, \mu_{in})$, to the threshold $\gamma\hat{u}$. If the latency of node n is smaller than the threshold, fog node i will immediately select node n. This procedure is repeated until fog node i observes N arrivals and selects $\hat{J}$ neighboring nodes. After the fog network is formed, the task distribution is done to minimize latency. In the next section, we investigate the property of the optimal task distribution, and show that the threshold can satisfy (17).

C. Task Distribution: Offline Optimization

Once the nodes are selected to form a network, the task distribution can be performed using an offline optimization problem which can be solved using known algorithms such as the interior-point algorithm [31]. From problem (15), the following properties can be derived, for a given $\mathcal{J}$.

Theorem 1. *If there exists a task distribution $\boldsymbol{\alpha}^*$ satisfying $u^* = D_i(\lambda_i) = D_c(\lambda_c, \mu_c) = D_j(\lambda_{ij}, \mu_{ij})$, $\forall j \in \mathcal{J}$, then $\boldsymbol{\alpha}^*$ is the unique and optimal solution of problem* (10).

Proof. Let $\boldsymbol{\alpha}$ be the initial task distribution, and assume that any other task distribution $\boldsymbol{\alpha'}$ different from $\boldsymbol{\alpha}$ is the optimal distribution. When $\boldsymbol{\alpha'}$ is considered, we can find a certain node A satisfying $\alpha'_A < \alpha_A$ where $\alpha'_A \in \boldsymbol{\alpha'}$ and $\alpha_A \in \boldsymbol{\alpha}$. This, in turn, yields $D_A(\alpha'_A) < D_A(\alpha_A)$. Due to the constraint (11), there exists another node B such that $B \neq A$, $\alpha'_B > \alpha_B$, and $D_B(\alpha'_B) > D_B(\alpha_B)$ where $\alpha'_B \in \boldsymbol{\alpha'}$ and $\alpha_B \in \boldsymbol{\alpha}$. Since $D_B(\alpha'_B) > D_B(\alpha_B) = D_A(\alpha_A) > D_A(\alpha'_A)$, we must decrease α'_B to minimize the maximum, i.e., $D_B(\alpha'_B)$. Thus, we can clearly see that $\boldsymbol{\alpha'}$ is not optimal, and, thus, the initial distribution $\boldsymbol{\alpha}$ is optimal.

Furthermore, $D_j(\lambda_{ij}, \mu_{ij})$ is a monotonically increasing function with respect to $\lambda_{ij} = x_i\alpha_{ij}$ since $\frac{\partial}{\partial\lambda_{ij}} D_j(\lambda_{ij}, \mu_{ij}) > 0$. Therefore, there are no more than two points of $\boldsymbol{\alpha}^*$ that have the same u^*. Hence, the distribution $\boldsymbol{\alpha}$ is unique and optimal. □

Theorem 1 shows that the optimal solution of the offline latency minimization problem results in an equal latency for all fog nodes and the cloud on the network (whenever such a solution is feasible). According to Theorem 1, selecting the node that has high computing resources is beneficial to minimize latency. Once fog node i determines the task distribution, the efficiency of the task distribution can be derived by applying the definition of task scheduling efficiency in [32]:

Definition 1. *For a task distribution $\boldsymbol{\alpha}$, the* efficiency *is given by*

$$\Gamma = 1 + \frac{\sum_{k\in\{i,c,\{ij|j\in\mathcal{J}\}\}} \max\{D_i(\alpha_i), D_c(\alpha_c,\mu_c), D_{j\in\mathcal{J}}(\alpha_{ij},\mu_{ij})\} - D_k}{D_i(\alpha_i) + D_c(\alpha_c) + \sum_{j\in\mathcal{J}} D_j(\alpha_{ij})} \geq 1. \tag{18}$$

In other words, Γ is defined as one plus the ratio between the total idle time of the fog computing nodes and the total transmission and computing time. Therefore, $\Gamma = 1$ means that all nodes in the fog network can complete their assigned tasks with the same latency. Theorem 1 shows that the optimal latency is $u^* = D_i(\lambda_i) = D_c(\lambda_c,\mu_c) = D_j(\lambda_{ij},\mu_{ij})$. Since u^* is the maximum value among $D_i(\lambda_i)$, $D_c(\lambda_c,\mu_c)$, and $D_j(\lambda_{ij},\mu_{ij})$, from (10), the efficiency of the optimal task distribution will be equal to one. Thus, if the efficiency of the task distribution becomes one, the latency of the task distribution is the optimal latency u^* according to Theorem 1.

Next, we show that the proposed framework can achieve the target competitive ratio γ.

Theorem 2. *Algorithm 1 satisfies ALG($\boldsymbol{\sigma}$)/OPT($\boldsymbol{\sigma}$) $\leq \gamma$ if (i) a given γ enables fog node i to select $\hat{J}$ nodes and (ii) the optimal task distribution can always be found, i.e.,* $\Gamma = 1$.

Proof. The offline optimal latency of the nodes in $\mathcal{J}$ is greater than or equal to $\hat{u}$, i.e., $\hat{u} \leq$ OPT($\boldsymbol{\sigma}$). Also, in Algorithm 1, the selected nodes satisfy $D_j(\hat{\lambda}_{ij},\mu_{ij}) \leq \gamma\ \hat{u}$, $\forall j \in \mathcal{J}$ where $|\mathcal{J}| = \hat{J}$. When the task distribution is not yet optimized with respect to $\mathcal{J}$, the latency that results from using distribution $\{\hat{\lambda}_i, \hat{\lambda}_c, \hat{\lambda}_{ij}\}$ can be shown as

$$\text{ALG}_b(\boldsymbol{\sigma}) = \max\left\{D_i(\hat{\lambda}_i),\ D_c(\hat{\lambda}_c,\mu_c),\ D_{j\in\mathcal{J}}(\hat{\lambda}_{ij},\mu_{ij})\right\}.$$

Recall that $\hat{u} \triangleq \max\left\{D_i(\hat{\lambda}_i),\ D_c(\hat{\lambda}_c,\mu_c),\ D_{\bar{j}}(\hat{\lambda}_{ij},\mu_{i\bar{j}})\right\}$, and, by Theorem 1, $\hat{u} = D_i(\hat{\lambda}_i) = D_c(\hat{\lambda}_c,\mu_c) = D_{\bar{j}}(\hat{\lambda}_{ij},\mu_{i\bar{j}})$. Since the service rates and computing speeds of node $j \in \mathcal{J}$ are less than those of the ideal node $\bar{j}$, we have $\hat{u} \leq D_{j\in\mathcal{J}}(\hat{\lambda}_{ij},\mu_{ij})$. Therefore, we have the following:

$$\text{ALG}_b(\boldsymbol{\sigma}) = \max\left\{\hat{u},\ D_j(\hat{\lambda}_{ij},\mu_{ij})\right\} = \max\left\{D_j(\hat{\lambda}_{ij},\mu_{ij})\right\} \leq \gamma\ \hat{u}, \forall j \in \mathcal{J}.$$

By optimizing the task distribution for the nodes in $\mathcal{J}$, the latency can be further reduced, i.e, ALG($\boldsymbol{\sigma}$) $\leq$ ALG$_b$($\boldsymbol{\sigma}$). Hence, it is possible to conclude that ALG($\boldsymbol{\sigma}$) $\leq$ ALG$_b$($\boldsymbol{\sigma}$) $\leq \gamma\ \hat{u} \leq$ γ OPT($\boldsymbol{\sigma}$) and, therefore, ALG($\boldsymbol{\sigma}$)/OPT($\boldsymbol{\sigma}$) $\leq \gamma$. □

This result shows that the online optimization framework can achieve the target competitive ratio γ by determining a proper network size $\hat{J}$ and optimizing the task distribution. According to Theorem 2, the ratio between the latency achieved by Algorithm 1 and an offline optimal latency can be bounded by the value of γ. To satisfy the first condition of Theorem 2, the proper value of γ needs to be found iteratively as shown in Fig. 3. Then, we prove that γ converges to

an upper bound. For this proof, we define the maximum of d_{ij} as $\bar{d}_{ij}$, and the lowest computing service rate and the processing speed as $\underline{\mu}_j$ and $1/\bar{\omega}_j$, respectively.

Theorem 3. *In Algorithm 1, the target competitive ratio γ converges to $D_j(\hat{\lambda}_{ij}, \underline{\mu}_{ij})/\hat{u}$.*

Proof. We show that there exists an upper bound of γ denoted by $\bar{\gamma}$. Therefore, for a given sequence $\boldsymbol{\sigma}$, we show that $\frac{\text{ALG}(\boldsymbol{\sigma})}{\text{OPT}(\boldsymbol{\sigma})} \leq \frac{\max_{\boldsymbol{\sigma}'} \text{ALG}(\boldsymbol{\sigma}')}{\min_{\boldsymbol{\sigma}'} \text{OPT}(\boldsymbol{\sigma}')} = \bar{\gamma}$, where $\boldsymbol{\sigma}'$ denotes any sequence. In the first phase of Algorithm 1, since $\hat{u}$ is calculated by assuming that all neighboring nodes are ideal nodes, the lower bound of the offline latency for any sequence is given by $\min_{\boldsymbol{\sigma}'} \text{OPT}(\boldsymbol{\sigma}') = \hat{u}$. Also, if $\hat{J}$ neighboring nodes are located at the farthest distance $\bar{d}_{ij}$, the lowest fog transmission service rate denoted as $\underline{\mu}_{ij}$ is derived. Then, the worst case is defined by assuming that the neighboring nodes have the lowest service rates and computing speed, i.e., $\underline{\mu}_{ij}$, $\underline{\mu}_j$, and $1/\bar{\omega}_j$. Therefore, the latency in the worst case can be presented by $\max_{\boldsymbol{\sigma}'} \text{ALG}(\boldsymbol{\sigma}') = D_j(\hat{\lambda}_{ij}, \underline{\mu}_{ij})$. Finally, γ always increases when it is updated, and, hence, γ converges to $\bar{\gamma} = \frac{D_j(\hat{\lambda}_{ij}, \underline{\mu}_{ij})}{\hat{u}}$. □

Therefore, the proposed framework is able to find the target competitive ratio by iteratively updating γ when $\bar{d}_{ij}$, $\underline{\mu}_j$, and $1/\bar{\omega}_j$ are not known to fog node i. Thus, once γ is found through the iterative process, Algorithm 1 is used to select the neighboring nodes, and the tasks are offloaded to the neighboring nodes as stated in Theorem 1. As a result, the proposed framework yields the set of selected neighboring nodes and the corresponding task distribution that can achieve the target competitive ratio as shown in Theorem 2. Next, we provide a theoretical analysis of the task distribution resulting from the proposed framework.

D. Task Distribution Analysis: Case Studies

By using the results of Theorem 1, we analyze how the different parameters in the problem (15) affect the task distribution in two exemplary cases. For each example, the optimal task distribution $\boldsymbol{\alpha}$ is calculated by solving the equations $D_i(\lambda_i) = D_j(\lambda_{ij}, \mu_{ij}) = D_c(\lambda_c, \mu_c)$ with respect to $\boldsymbol{\alpha}$. The optimal task distribution derived from Theorem 1 consists of many variables in problem (10), and, hence, the relationships between the variables cannot be readily observed. Therefore, by considering special network cases, we simplify the solution of the problem and gain insights on how the different variables affect the optimal task distribution. We consider two different network scenarios. In the first scenario, we consider a fog network that includes multiple neighboring nodes ($J \geq 2$) with large service rates and, thus, the queueing delay in T_j and S_j is negligible. This first scenario shows the impact of the processing speed of the fog nodes. In the second scenario, we consider a network having only two fog nodes and a cloud

with the cloud transmission service rate being set to a value that is much larger than the fog transmission service rate. This second scenario illustrates how many tasks are offloaded to the cloud with respect to the cloud transmission service rate.

1) Scenario 1 - Large service rates: We assume that the initial fog node offloads its tasks to J neighboring fog nodes and the cloud when the transmission service rates of all nodes are very large, i.e., $\mu_{ij}, \mu_c \to \infty$. In this scenario, the computing service rates are also taken to be large, i.e., $\mu_i, \mu_j \to \infty$. Under these assumptions, the latency functions of the nodes will be $D_i(\lambda_i) = \omega_i \lambda_i$, $D_j(\lambda_{ij}, \mu_{ij}) = \omega_j \lambda_{ij}$, and $D_c(\lambda_c, \mu_c) = \omega_c \lambda_c$. Moreover, the processing latency becomes the only factor to be considered.

We find λ_i, λ_{ij}, and λ_c by using $D_i(\lambda_i) = D_j(\lambda_{ij}, \mu_{ij}) = D_c(\lambda_c, \mu_{ij})$. From $\omega_i \lambda_i = \omega_j \lambda_{ij}$, the task distribution of node j is given by $\lambda_{ij} = \frac{\omega_i}{\omega_j} \lambda_i$. Also, $\lambda_c = x_i - \left(1 + \sum_j \frac{\omega_i}{\omega_j}\right) \lambda_i$ due to (11). Then, by using $\omega_i \lambda_i = \omega_c \lambda_c$, the task distribution on the network will be

$$\lambda_i = \frac{\omega_c x_i}{\omega_i + \omega_c \left(1 + \sum_{j=1}^{J} \frac{\omega_i}{\omega_j}\right)}, \tag{19}$$

$$\lambda_{ij} = \frac{\omega_i \omega_c x_i}{(\omega_i + \omega_c)\omega_j + \omega_i + \omega_j \sum_{j'=1, j' \neq j}^{J} \frac{\omega_i}{\omega_{j'}}}, \tag{20}$$

$$\lambda_c = x_i - \left(1 + \sum_{j=1}^{J} \frac{\omega_i}{\omega_j}\right) \frac{x_i}{\frac{\omega_i}{\omega_c} + \left(1 + \sum_{j=1}^{J} \frac{\omega_i}{\omega_j}\right)}. \tag{21}$$

These results show the optimal task distribution between the fog nodes and the cloud. It can be seen that the number of offloaded tasks to a node can increase if the node has a higher processing speed, e.g., $1/\omega_i$, $1/\omega_j$, or $1/\omega_c$. Also, if the processing speed of the cloud is much faster than that of the fog nodes, i.e., $1/\omega_c \to \infty$, then the task distribution is given by $\lim_{\omega_c \to 0} \lambda_c = x$ and $\lim_{\omega_c \to 0} \lambda_i = \lim_{\omega_c \to 0} \lambda_{ij} = 0$. In other words, all tasks can be offloaded to the cloud if the service rates are high enough and the processing delay at the cloud is negligibly small.

2) Scenario 2 - Two fog node case: Here, the initial fog node offloads the traffic to only one neighboring fog node and the cloud. It is assumed that the fog transmission service rate is high, i.e., $\mu_{ij} \to \infty$ while the cloud transmission service rate is finite. This represents the case in which the distance to the neighboring fog node is very close whereas the distance to the cloud is large. Here, we also consider highly capable nodes having large computing service rates, i.e., $\mu_i, \mu_j \to \infty$ and equal processing speeds, i.e., $1/\omega_i = 1/\omega_j = 1/\omega$. In this scenario, the latency functions of the nodes will be given by $D_i(\lambda_i) = \omega \lambda_i$, $D_j(\lambda_{ij}, \mu_{ij}) = \omega \lambda_{ij}$, and $D_c(\lambda_c, \mu_c) = \frac{\lambda_c}{2\mu_c(\mu_c - \lambda_c)} + \frac{1}{\mu_c} + \omega_c \lambda_c$, respectively.

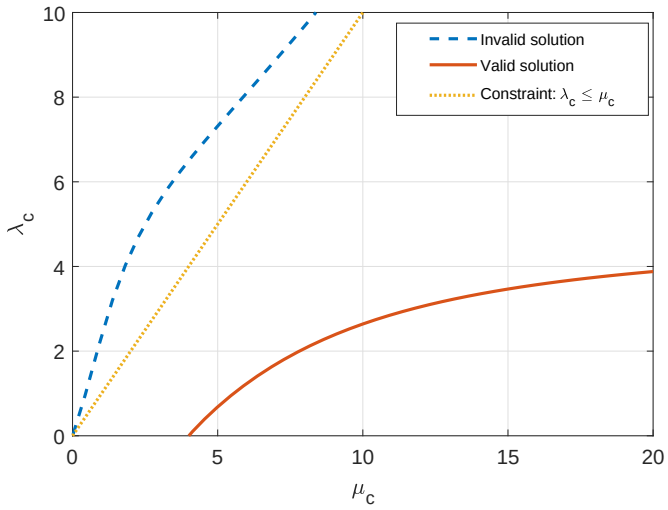

Fig. 4: Solutions of (22) when $\omega = 0.05, \omega_c = 0.025$, and $x_i = 10$.

From $D_i(\lambda_i) = D_j(\lambda_{ij}, \mu_{ij})$, fog nodes i and j process the same amount of tasks, i.e., $\lambda_i = \lambda_{ij}$, and, hence, the tasks sent to the cloud will be $\lambda_i = \frac{x-\lambda_c}{2}$ since $\lambda_i + \lambda_{ij} + \lambda_c = x_i$. Then, the relationship $D_i(\lambda_i) = D_c(\lambda_c, \mu_c)$ will yield:

$$(\omega + 2\omega_c)\lambda_c^2 - \left((\omega + 2\omega_c)\,\mu_c + \omega x_i - \frac{1}{\mu_c}\right)\lambda_c + \omega x_i \mu_c - 2 = 0. \tag{22}$$

The solution of (22), which represents the tasks that are offloaded to the cloud, is shown in Fig. 4. The dashed curve is an infeasible solution since it does not satisfy constraint (13), and the solid line represents the only valid solution. As the cloud transmission service rate increases, the latency of the cloud can be reduced. Therefore, a higher service rate μ_c at the cloud leads to more tasks being offloaded to the cloud.

In contrast, it is possible to consider that the cloud transmission serive rate decreases. Then, from (22), we can study a special case in which no task is offloaded to the cloud. Let us consider the case $\omega x_i = \frac{2}{\mu_c}$ where ωx_i represents the latency when all tasks are computed by the initial fog node. If $\omega x_i = \frac{2}{\mu_c}$, (22) becomes:

$$(\omega + 2\omega_c)\lambda_c \left(\lambda_c - \left(\mu_c + \frac{cx_i - 1/\mu_c}{\omega + 2\omega_c}\right)\right) = 0. \tag{23}$$

Therefore, the optimal task arrival rate offloaded to the cloud can be given by either 0 or $\mu_c + \frac{\omega x_i - 1/u_c}{\omega + 2\omega_c}$. However, $\lambda_c = \mu_c + \frac{\omega x_i - 1/u_c}{\omega + 2\omega_c}$ is an invalid task distribution since $\lambda_c = \mu_c + \frac{\omega x_i - 1/u_c}{\omega + 2\omega_c} = \mu_c + \frac{1/\mu_c}{\omega + 2\omega_c}$ is greater than μ_c, which violates constraint (13). Therefore, the only valid solution is $\lambda_c = 0$, which proves that fog node i does not offload any task to the cloud. Here, all tasks x will be equally distributed between the initial fog node i and the neighboring fog node j, thus resulting in a latency of $D_i(\lambda_i) = D_j(\lambda_{ij}, \mu_{ij}) = \frac{\omega x_i}{2}$ as shown in Fig. 4 with

TABLE II: Simulation parameters

Notation	Value
$\omega_i = \omega_j$, ω_c	50, 25 msec/packet
$\underline{\mu}_i = \underline{\mu}_j$, $\bar{\mu}_i = \bar{\mu}_j$	15, 40 packet/sec
N, τ	300, 0.002 (0.005 in Fig. 7)
$P_{\text{tx},i}$, β_1, β_2	20 dBm, 10^{-3}, 4
K	64 kilobytes
B, N_0	3 MHz, -174 dBm/Hz

$\mu_c = 4$ and $\omega x = \frac{2}{\mu_c} = 0.5$. Moreover, from Fig. 4, we can see that no task is assigned to the cloud if the service rate of the cloud is less than 4. A low cloud transmission service rate leads to a substantially higher latency for cloud communication, i.e., the maximum among all latencies in the network.

When no task is offloaded to the cloud, the cloud has a minimum latency of $\frac{1}{\mu_c}$ since $T_c(\lambda_c = 0, \mu_c) = \frac{1}{\mu_c}$ in (7). This minimum latency can be seen as the time delay for a small packet exchange to maintain the link between fog node i and the cloud in an active status even when there is no actual task assignment to the cloud [33]. If $\frac{1}{\mu_c}$ is greater than the latency of any one of the fog nodes, then the maximum latency of the fog computing layer will be determined by the cloud's latency. In that case, the scheduling efficiency is greater than one since the numerator in (18) is positive, i.e., $\max\{D_i(\lambda_i), D_c(\lambda_c, \mu_c), D_j(\lambda_{ij}, \mu_{ij})\} - D_k = \frac{1}{\mu_c} - D_k > 0$. Hence, for the case with large $\frac{1}{\mu_c}$, no task distribution can be optimal since there is a non-zero idle time. This implies that we cannot fully take advantage of fast fog transmission service rates for distributed computing since the the large cloud transmission latency dominantly determines the maximum latency. Thus, to enhance the scheduling efficiency, the transmission latency of the cloud must be reduced by increasing μ_c. To this end, the system needs to allocate more radio resources for the transmission to the cloud, and, hence, the cloud-centric bandwidth allocation scheme can be useful to achieve the optimal scheduling efficiency.

V. Simulation Results and Analysis

For our simulations, we consider an initial fog node that can connect to neighboring fog nodes uniformly distributed within a circular area of radius 50 m. The arrival sequence of the fog nodes follows a uniform distribution. The task arrival rate at fog node i is $x_i = 10$ packets per second. The computing service rate of the fog nodes is randomly drawn from a uniform distribution over a range of 15 to 40 packets per second. All statistical results are averaged over a large number of simulation runs with the parameters listed in Table II.

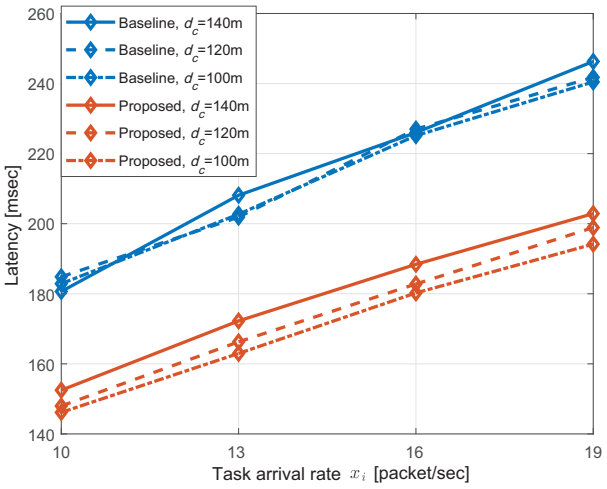

Fig. 5: Latency for different task arrival rates at the initial fog node i.

A. *Performance Evaluation of the Online Optimization Framework*

Fig. 5 shows the latency when the total task arrival rate increases from 10 to 19 packets per second with $d_c = 100$, 120, and 140 m, respectively. For comparison purposes, we use a baseline algorithm in which the algorithm observes the first 110 over 300 observations nodes and then selects the neighboring nodes from the rest of the arrivals by using the *secretary algorithm* in [1]. In Fig. 5, we show that the proposed framework can reduce the latency compared to the baseline, for all task arrival rates. For instance, the latency can be reduced by up to 19.25% compared to the baseline when $x_i = 19$ and $d_c = 140$ m. Also, from Fig. 5, we can see that the latency decreases as the distance to the cloud is reduced. With a shorter distance to the cloud, the cloud transmission service rate becomes higher. Therefore, the cloud is able to process more tasks with a low latency, and the overall latency of the fog network is improved. For example, at $x_i = 19$, if d_c decreases from 140 m to 100 m, the latency is reduced by 4.29%. Moreover, we show that the latency decreases as less tasks arrive at the initial fog node i. For instance, when x_i decreases from 19 to 10, the latency is reduced by about 25% with $d_c = 100$ m.

Fig. 6 shows the relationship between the latency and the number of neighboring nodes when the total task arrival rate is given by $x_i = 10$ and 13 packets per second, respectively, and the processing delays of the fog nodes are given by $\omega_i = \omega_j = 50$ and 30 milliseconds, respectively. In Fig. 6, a smaller processing delay indicates that the fog nodes have a higher processing speed. From Fig. 6, we can observe the tradeoff between scenarios having a large number of fog nodes with low processing power and scenarios having a small number of fog nodes with high processing power. If fog nodes with higher processing speed are deployed, latency is reduced,

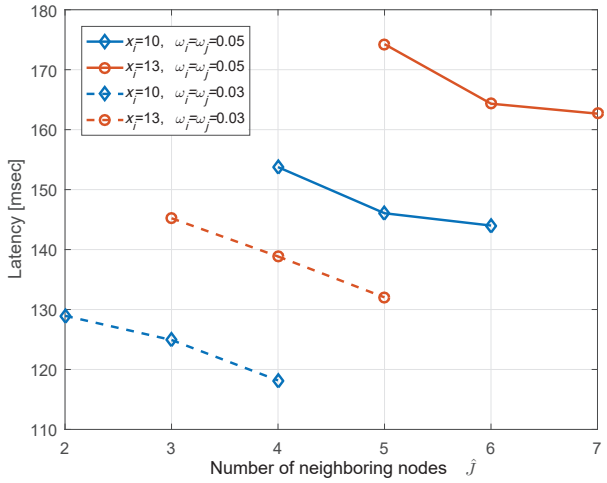

Fig. 6: Latency for different number of neighboring nodes.

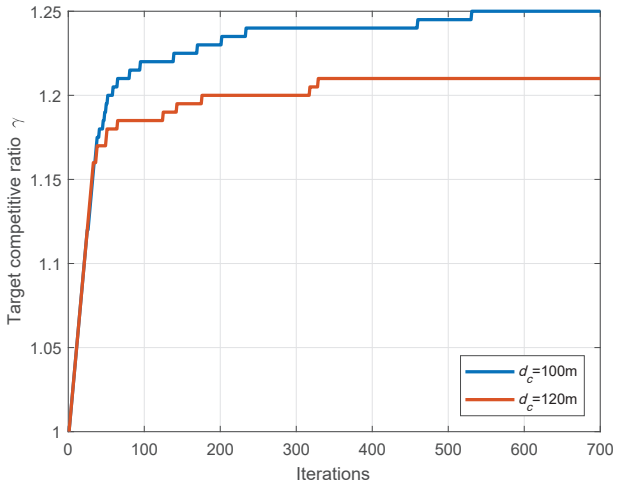

Fig. 7: Changes in the target competitive ratio γ over 700 updates.

and the formed network size decreases. This is due to the fact that the fog nodes having a faster processing speed do not need to form a large network. In fact, a larger network size can lead to lower transmission service rates. For instance, if the processing delay of fog nodes decreases from 50 to 30 milliseconds, the latency is reduced by up to 18.8% while the number of neighboring nodes decreases from 7 to 5.

Fig. 7 plots the value of γ during 700 updates for different distances to the cloud, $d_c = 100$ m and 120 m, respectively. Fig. 7 shows that the value of γ approaches a constant value. For instance, γ first reaches 1.17 at 38 iterations with $d_c = 120$ m. Then, γ becomes 1.21 at 329 iterations, and this value is maintained thereafter. From Fig. 7, we can see that fog node i can find a proper γ after a finite number of trials and updates. Also, the results of Fig. 7 show that γ becomes larger as the distance to the cloud is closer. This is because $\hat{u}$ and the threshold value

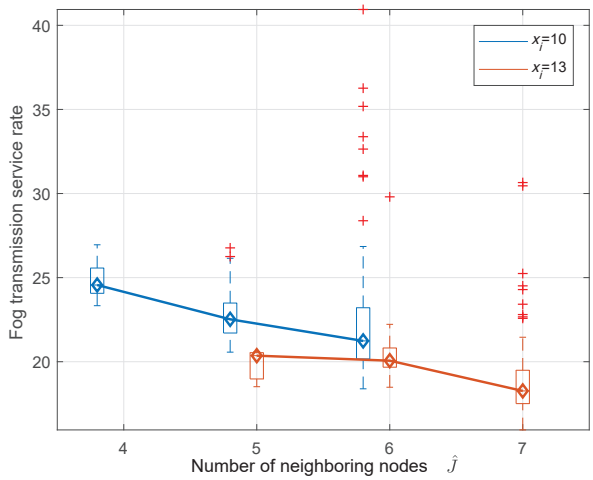

Fig. 8: Fog transmission service rate with respect to the number of neighboring nodes.

decrease when d_c is reduced. If the threshold value decreases, it becomes more challenging to select the $\hat{J}$ neighboring nodes within the limited number of observations since the selected neighboring nodes must have a lower latency than the threshold. Therefore, in order to maintain a proper threshold value, γ will be larger when d_c decreases.

Fig. 8 shows the relationship between the fog transmission service rate and the number of neighboring nodes when $x_i = 10$ and 13, respectively. Here, we can see that the fog transmission service rate increases as the number of neighboring nodes decreases. This stems from the fact that the bandwidth per node increases as less fog nodes share the total bandwidth. For instance, the fog transmission service rate can increase by 15.6% if $\hat{J}$ goes from 6 to 4 with $x_i = 10$. Fig. 8 also shows that the formed network size becomes larger if x_i increases. This is due to the fact that offloading tasks to a larger size of the network can reduce the tasks per node, and, hence, the maximum latency of the network will decrease. For instance, when $x_i = 10$, the range of $\hat{J}$ is between 4 and 6. However, if $x_i = 13$, $\hat{J}$ falls in the range between 5 and 7.

In Fig. 9, we show the task distribution among neighboring nodes, the cloud, and fog node i for different numbers of neighboring nodes when two bandwidth allocation approaches are used, respectively. It can be seen that the cloud-centric bandwidth allocation increases the tasks offloaded to the cloud when compared to the equal-bandwidth allocation. This is because the cloud transmission service rate increases, so offloading more tasks to the cloud can lower the latency. For instance, if the cloud-centric bandwidth allocation is used and $\hat{J} = 4$, the cloud is allocated 22.86% more tasks than in the case of equal bandwidth allocation. Also, in Fig. 9, we show that the optimal network size is different, depending on the bandwidth allocation

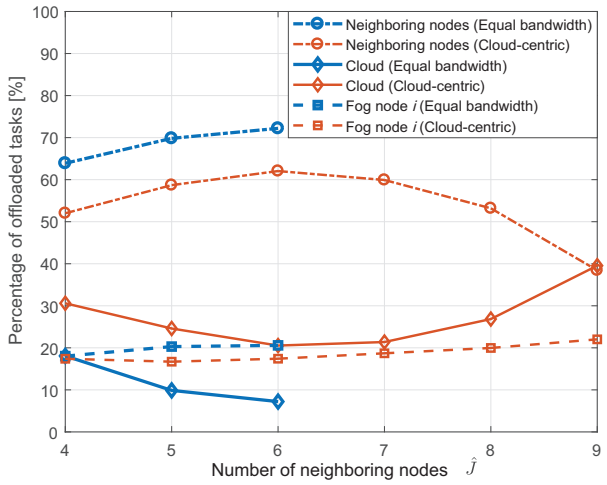

Fig. 9: Task distribution with respect to the number of neighboring nodes.

scheme. For instance, the cloud-centric bandwidth allocation yields a larger network size than the equal bandwidth allocation. When the network size is large, the cloud can maintain a high transmission service rate by using the cloud-centric bandwidth allocation. Therefore, the high cloud transmission service rate enables to offload most tasks to the cloud with a low transmission latency. For example, Fig. 9 shows that the number of neighboring nodes is between 4 and 6 if equal bandwidth allocation is used. However, if the cloud-centric bandwidth allocation is used, the number of neighboring nodes varies from 4 to 9. Moreover, Fig. 9 shows that the number of tasks offloaded to the cloud decreases when $\hat{J}$ increases from 4 to 6 for both bandwidth allocation schemes. In this phase, the number of tasks offloaded to neighboring nodes will increase because offloading more tasks at the fog layer can reduce the latency at the cloud. However, if the number of neighboring nodes increases when using the cloud-centric bandwidth allocation, e.g., there are 7 or more neighboring nodes, the number of tasks offloaded to the neighboring nodes will decrease with the network size. This is due to the fact that the fog transmission service rates are smaller for larger networks which yields higher fog transmission latency. As a result, more tasks will be allocated to the cloud so as to utilize its fast computing resources.

B. Performance Evaluation of Algorithm 1 for a fixed γ

In Figs. 10 and 11, we evaluate the performance of Algorithm 1 when the proposed framework uses a fixed value of γ without constraint (14). By using a predefined γ, the update step of γ is not needed, which can be useful for scenarios in which the delay of this update can hinder the network latency. Fig. 10 shows the latency for the different preset values of γ ranging from 1.2 to 1.5 with $d_c = 100$ m and 120 m, respectively. From Fig. 10, we can see that

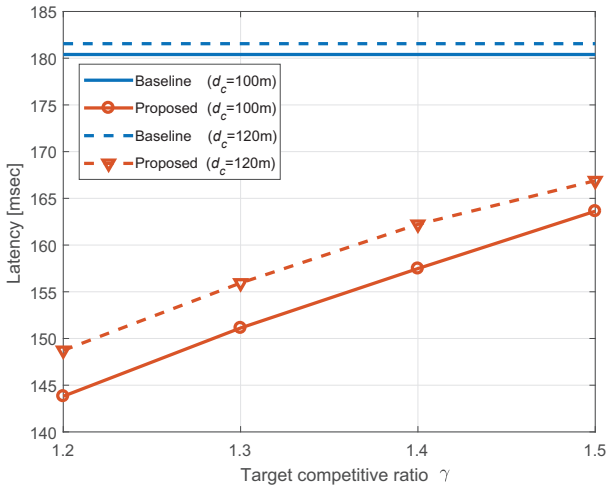

Fig. 10: Latency comparison versus the target competitive ratio.

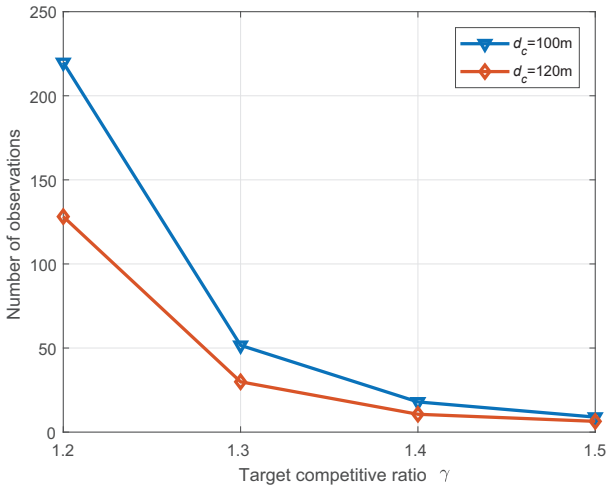

Fig. 11: The required number of observations for different values of γ.

the proposed algorithm achieves lower latencies than the baseline, for all γ. For instance, the proposed algorithm can reduce the latency by 20.3% compared to the baseline if $\gamma = 1.2$ and $d_c = 100$ m. Also, Fig. 10 shows that the latency achieved by the proposed algorithm becomes smaller when γ decreases. This stems from the fact that a low threshold value with small γ allows the algorithm to only select fog nodes having a high performance. For example, the latency can be reduced by up to 12.1% if γ decreases from 1.5 to 1.2 with $d_c = 100$ m.

In Fig. 11, we show the number of observations of the neighboring node arrivals until $\hat{J}$ neighboring nodes are selected for different γ with $d_c = 100$ m and 120 m, respectively. In this figure, we can see that a large value of γ results in a small number of observations due to the associated increase in the threshold value. For instance, as γ increases from 1.2 to 1.5, the number of observations can be reduced by about 96% with $d_c = 100$ m. Fig. 11 shows that

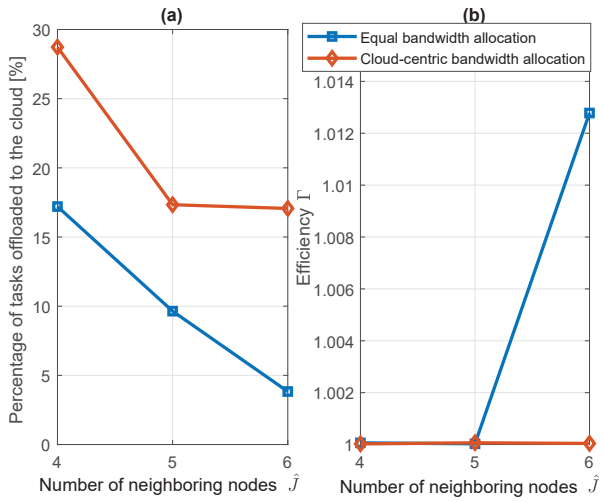

Fig. 12: Performance comparison of two bandwidth allocation schemes with respect to the number of neighboring nodes.

a large value of d_c lowers the number of observations since increasing d_c results in a large $\hat{u}$ and threshold value. For example, the number of observations can be reduced by about 42% if d_c increases from 100 m to 120 m with $\gamma = 1.2$. Moreover, from Figs. 10 and 11, we can characterize the tradeoff between the latency and the number of observations. In particular, a small γ results in a lower latency, but requires a large number of observation.

Fig. 12 shows the percentage of tasks offloaded to the cloud and the scheduling efficiency of the task distribution when two bandwidth allocation schemes are used, respectively, with $\gamma = 1.2$ and $d_c = 100$ m. In Fig. 12 (a), the tasks offloaded to the the cloud decreases as the number of fog nodes increases since the cloud transmission service rate decreases. Also, Fig. 12 (b) shows that, when equal bandwidth allocation is used for a large network size, the scheduling efficiency may not be optimal, i.e., $\Gamma > 1$ due to a large latency for the transmissions to the cloud. In this case, though the equal-bandwidth allocation still achieves Γ that is close to 1, the cloud-centric bandwidth allocation can be used to enhance efficiency. This is because the cloud-centric bandwidth allocation increases the cloud transmission service rate by allocating more bandwidth. It can be seen for instance that the equal bandwidth allocation yields $\Gamma = 1.013$ in the case of 6 neighboring nodes, but the efficiency of the cloud-centric bandwidth allocation becomes $\Gamma = 1$.

C. Optimal Network Size in an Offline Setting

Fig. 13 shows the optimal latency for different network sizes when all neighboring nodes are located at d_{ij} varying from 10 m to 40 m. In Fig. 13, it is assumed that complete information on the network is known and that the fog nodes have identical parameters, i.e., $\mu_i = \mu_j = 20$

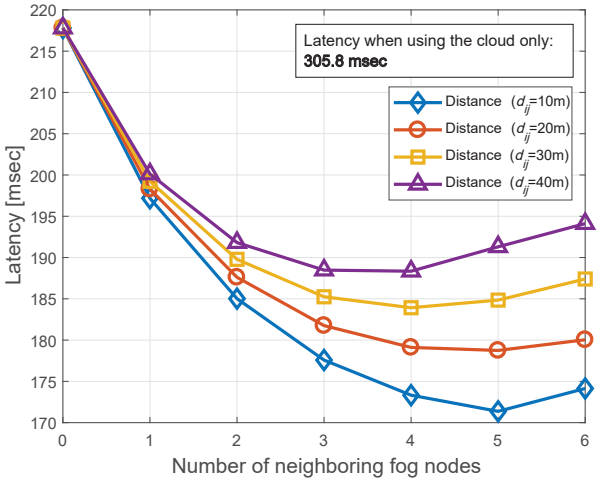

Fig. 13: Latency for different number of neighboring fog nodes in an offline setting.

when $d_c = 150$ m. In this offline setting, we study the impact of the network size on the latency by using an offline optimization solver to find the optimal latency for a given network. Fig. 13 shows that the optimal latency is directly affected by the number of neighboring nodes. When the network size increases, latency starts to decrease since fewer tasks can be offloaded to each neighboring node. However, if the network size increases, the latency will eventually increase since the bandwidth per node decreases is lowered. For example, the optimal latency decreases when the number of neighboring nodes increases from 1 to 3 with $d_{ij} = 40$. However, once the number of neighboring nodes increases beyond 3, the latency starts to increase. Moreover, from Fig. 13, we can see that the optimal network size changes with the distances between fog nodes. For instance, for $d_{ij} = 40$ m, the latency can be minimized when there are 3 neighboring nodes in the fog network. However, if $d_{ij} = 10$ m, the latency is minimized when the number of neighboring nodes is 5. Therefore, if the fog transmission service rate is high (for shorter distances), increasing the number of neighboring nodes to 5 can reduce the latency. On the other hand, if the fog transmission service rate is low (due to poor wireless channel), having a smaller network size with 3 nodes is required to minimize the latency. Finally, Fig. 13 clearly shows that the latency is reduced by offloading the tasks to both the fog layer and the cloud, instead of relying solely on the cloud. For example, if the tasks are offloaded to the cloud, initial fog node, and 5 neighboring nodes located at $d_{ij} = 10$ m, the latency can be reduced by up to 43.9% compared to the case using the cloud only.

VI. Conclusion

In this paper, we have proposed a novel framework to jointly optimize the formation of fog networks and the distribution of computational tasks in a hybrid fog-cloud system. We have addressed the problem using an online optimization formulation whose goal is to minimize

the maximum latency of the nodes in the fog network in presence of uncertainty about fog nodes' arrivals. To solve the problem, we have proposed online optimization algorithms whose target competitive ratio is achieved by suitably selecting the neighboring nodes while effectively offloading the tasks to the neighboring fog nodes and the cloud. The theoretical analysis and simulation results have shown that the proposed framework achieves a low target competitive ratio while successfully minimizing the maximum latency in fog computing. The simulation results have shown that a judicious selection of neighboring nodes can reduce the latency by up to 19.25% compared to the baseline.

REFERENCES

[1] G. Lee, W. Saad, and M. Bennis, "An online secretary framework for fog network formation with minimal latency," in *Proc. IEEE Int. Conf. on Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.

[2] Z. Dawy, W. Saad, A. Ghosh, J. G. Andrews, and E. Yaacoub, "Toward massive machine type cellular communications," *IEEE Wireless Communications*, vol. 24, no. 1, pp. 120–128, Feb. 2017.

[3] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Unmanned aerial vehicle with underlaid device-to-device communications: Performance and tradeoffs," *IEEE Trans. Wireless Commun.*, vol. 15, no. 6, pp. 3949–3963, Jun. 2016.

[4] T. Park, N. Abuzainab, and W. Saad, "Learning how to communicate in the Internet of Things: Finite resources and heterogeneity," *IEEE Access*, vol. 4, pp. 7063–7073, Nov. 2016.

[5] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[6] Cisco, "Fog computing and the Internet of Things: Extend the cloud to where the things are," *Cisco white paper*, 2015.

[7] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: issues and challenges," *IEEE Network*, vol. 30, no. 4, pp. 46–53, Jul. 2016.

[8] M. S. ElBamby, M. Bennis, and W. Saad, "Proactive edge computing in latency-constrained fog networks," in *Proc. European Conference on Networks and Communications (EuCnC)*, Oulu, Finland, May 2017, pp. 1–6.

[9] G. Lee, W. Saad, and M. Bennis, "Online optimization techniques for effective fog computing under uncertainty," *MMTC Communications-Frontiers*, vol. 12, no. 4, pp. 19–23, Jul. 2017.

[10] M. Yannuzzi, R. Milito, R. Serral-Graci, D. Montero, and M. Nemirovsky, "Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing," in *Proc. IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Athens, Greece, Dec 2014, pp. 325–329.

[11] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. 1st MCC workshop on Mobile cloud computing*. Helsinki, Finland: ACM, Aug. 2012, pp. 13–16.

[12] C. Vallati, A. Virdis, E. Mingozzi, and G. Stea, "Exploiting LTE D2D communications in M2M fog platforms: Deployment and practical issues," in *Proc. IEEE 2nd World Forum on IoT*, Milan, Italy, Dec. 2015, pp. 585–590.

[13] A. Khelil and D. Soldani, "On the suitability of device-to-device communications for road traffic safety," in *Proc. IEEE World Forum on Internet of Things (WF-IoT)*, Seoul, Korea, Mar. 2014, pp. 224–229.

[14] T. H. Luan, L. X. Cai, J. Chen, X. Shen, and F. Bai, "Vtube: Towards the media rich city life with autonomous vehicular content distribution," in *Proc. 8th Annual IEEE Commun. Society Conference on Sensor, Mesh and Ad Hoc Commun. and Networks*, Salt Lake City, UT, USA, Jun. 2011, pp. 359–367.

[15] T. Nishio, R. Shinkuma, T. Takahashi, and N. B. Mandayam, "Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud," in *Proc. 1st Int. Wksh. on Mobile Cloud Comput. Netw.*, Bangalore, India, Jul. 2013, pp. 19–26.

[16] V. Sharma, J. D. Lim, J. N. Kim, and I. You, "SACA: Self-aware communication architecture for IoT using mobile fog servers," *Mobile Information Systems*, vol. 2017, pp. 1–17, Apr. 2017.

[17] T. Zhao, S. Zhou, X. Guo, and Z. Niu, "Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing," in *Proc. IEEE Int. Conf. on Commun. (ICC)*, Paris, France, May 2017, pp. 1–7.

[18] R. Kaewpuang, D. Niyato, P. Wang, and E. Hossain, "A framework for cooperative resource management in mobile cloud computing," *IEEE J. Sel. Areas in Commun.*, vol. 31, no. 12, pp. 2685–2700, Dec. 2013.

[19] M. Khaledi, M. Khaledi, and S. K. Kasera, "Profitable task allocation in mobile cloud computing," in *Proc. 12th Int. Symp. on QoS and Security for Wireless and Mobile Networks*, Malta, Nov. 2016.

[20] I. Ketyk, L. Kecsks, C. Nemes, and L. Farkas, "Multi-user computation offloading as multiple knapsack problem for 5G mobile edge computing," in *Proc. European Conference on Networks and Communications (EuCnC)*, Athens, Greece, Jun. 2016, pp. 225–229.

[21] V. B. C. Souza, W. Ramirez, X. Masip-Bruin, E. Marn-Tordera, G. Ren, and G. Tashakor, "Handling service allocation in combined fog-cloud scenarios," in *Proc. IEEE Int. Conf. on Commun. (ICC)*, Kuala Lumpur, Malaysia, May 2016, pp. 1–5.

[22] S.-H. Park, O. Simeone, and S. Shamai, "Joint cloud and edge processing for latency minimization in fog radio access networks," in *Proc. IEEE 17th Int. Workshop on Signal Process. Adv. in Wireless Commun.*, Edinburgh, UK, Jul. 2016, pp. 1–5.

[23] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and CPU time allocation for mobile edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington DC, USA, Dec. 2016.

[24] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *Proc. IEEE Int. Conf. on Commun. (ICC)*, London, UK, Jun. 2015, pp. 3909–3914.

[25] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington DC, USA, Dec. 2016.

[26] G. Lee, W. Saad, and M. Bennis, "Online optimization for low-latency computational caching in fog networks," in *Proc. Fog World Congress 2017*, Santa Clara, CA, USA, Jun. 2017.

[27] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data networks*. Prentice-Hall International New Jersey, 1992, vol. 2.

[28] K. Doppler, C. B. Ribeiro, and J. Kneckt, "Advances in D2D communications: Energy efficient service and device discovery radio," in *Proc. 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE)*, Chennai, India, Feb. 2011, pp. 1–6.

[29] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg, "A knapsack secretary problem with applications," in *Proc. 10th International Workshop on Approximation and 11th International Workshop on Randomization, and Combinatorial Optimization*, Princeton, NJ, USA, Aug. 2007, pp. 16–28.

[30] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. Cambridge University Press, 2005.

[31] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[32] S. Mirshekarian and D. N. Sormaz, "Correlation of job-shop scheduling problem features with scheduling efficiency," *Expert Systems with Applications*, vol. 62, pp. 131–147, 2016.

[33] W. Saad, Z. Han, T. Başar, M. Debbah, and A. Hjørungnes, "Network formation games among relay stations in next generation wireless networks," *IEEE Trans. Wireless Commun.*, vol. 59, no. 9, pp. 2528–2542, Sep. 2011.