Deadline-aware Fair Scheduling for Offloaded Tasks in Fog Computing with Inter-fog Dependency

Mithun Mukherjee, *Member, IEEE*, Mian Guo, *Member, IEEE*, Jaime Lloret, *Senior Member, IEEE*, Razi Iqbal, *Senior Member, IEEE*, and Qi Zhang, *Member, IEEE*

***Abstract*—A fundamental problem in fog computing networks is how to schedule the deadline-aware offloaded tasks that directly arrive from the end-users and via other fog nodes. The computational resource allocation becomes more challenging when the tasks demand different delay-deadlines. In this letter, we aim to propose a scheduling strategy to maximize the number of the completed tasks within their respective deadlines while making the network strongly stable. We exploit Lyapunov drift-plus-penalty function on the queue length to schedule the tasks in the queues. Subsequently, the scheduling policy decides the amount of task to be offloaded to the underloaded fog nodes to fully utilize the computational resources offered by all fog nodes in the network. Our simulation results reveal that the proposed strategy outperforms the baseline schemes, especially when those tasks have distinct delay-deadlines.**

I. Introduction

Due to the proliferation of latency-sensitive services, the resource (e.g., computation, storage, and energy) constrained end-users seek additional resources from the fog/edge computing layers that reside near the network edge. It is expected that the offloaded tasks can be executed within their respective delay-deadlines to obtain the results [1]–[3]. However, at the same time, considering the offloaded tasks with different delay-deadlines from the multiple end-users, it becomes a challenging issue: *how to allocate resources to the high-priority task*. In several cases, offloading to either other fog nodes or the remote cloud can be another way to serve the resource-hungry tasks, however, several downsides cannot be ignored for these delay-bounded task processing. For example, on the one hand, the transmission delay dominates in the total delay when the tasks are offloaded to the cloud. On the other hand, the delay of computation offloading horizontally among fog nodes[1] has a significant impact on the task completion when the fog nodes collaborate for task processing [2], [4]. In addition, when the tasks are offloaded from one fog node to the other fog nodes, it creates a burden on these fog nodes [5]. The main reason is that the fog node comprises of the limited computational resources to process the latency-sensitive tasks, ignoring the scenario when the tasks compete with each other for the transmission resources [6], [7].

In this letter, our main objective is to minimize the failure probability to meet the different delay deadlines for the tasks that are arrived at the fog node. *How to find the optimum scheduling policy that gives higher number of tasks to be processed while making both queues stable* is studied in this letter. We mainly consider two queues in each of the fog nodes, namely, high- and low-priority queues. The tasks directly arrived from the end-users and offloaded from the fog nodes enter these queues based on their respective delay deadlines. In addition, we apply the Lyapunov drift for queue scheduling when the tasks in these two queues have demanding latency requirements.

II. Network Model

A. System Model

Consider a fog computing network with a set of fog nodes $\mathcal{N} = \{1, 2, \ldots, N\}$ and a set of end-users $\mathcal{K} = \{1, 2, \ldots, K\}$. The fog nodes and end-users are uniformly and randomly distributed over the entire network. The end-user generates tasks that are independent and identically distributed (i.i.d.) over time. The CPU cycle required to process one bit of the task is denoted as L, which is assumed to be same for all tasks generated by the end-users. We denote τ_n as the delay-deadline for the task type-n.

Assumption: Assuming a typical binary offloading scenario[2], the end-users offload their entire tasks, i.e., all the generated data packets to a nearby fog node, hereinafter referred to as *primary* fog node. We assume that an end-user can select only one primary fog node. Thus, denoting $\mathcal{K}_i = \{1, 2, \ldots, K_i\}$, $\sum_{i=1}^{N} K_i = K$ as the set of end-users that select the ith fog node as their primary fog node, we write $\mathcal{K}_i \cap \mathcal{K}_{i'} \equiv \emptyset$ for $i \neq i'$. When the primary fog node estimates that the available computing resource is not sufficient to process the tasks arrived at the time t within their delay-deadlines, the primary fog node would offload these tasks to its neighboring fog nodes. Therefore, a fog node receives the tasks from the direct end-users under its primary coverage and the tasks from the other end-users via its neighboring fog

This work was supported in part by the National Natural Science Foundation of China under Grants 61901128 and 61672174.

M. Mukherjee is with the Guangdong Provincial Key Laboratory of Petrochemical Equipment Fault Diagnosis, Guangdong University of Petrochemical Technology, Maoming 525000, China, e-mail: m.mukherjee@ieee.org

M. Guo is with the School of Electronic and Information Engineering, Guangdong University of Petrochemical Technology, Maoming 525000, China, e-mail: mian.guo123@gmail.com

J. Lloret is with the Universitat Politecnica de Valencia, Spain, e-mail: jlloret@dcom.upv.es

R. Iqbal is with the Al-Khawarizmi Institute of Computer Science, University of Engineering and Technology, Pakistan, e-mail: razi.iqbal@ieee.org

Q. Zhang is with the DIGIT, Department of Engineering, Aarhus University, 8000 Aarhus, Denmark, e-mail: qz@eng.au.dk

[1]In this article, we consider *horizontal* collaboration of fog nodes, and a *vertical* collaboration with the cloud subject to transmission capacity is a part of future work.

[2] In a *partial offloading*, a part of the task is offloaded to the fog node, and the rest of the task is locally processed at the end-user side.

nodes. We term the latter as offloaded tasks from fog nodes to explicitly differentiate the tasks arrived at the primary fog node. Moreover, the downloading of the task output is ignored due to the small data size of the results compared to the uploaded task data size. However, we can easily extend our system model by considering the downloading time from the fog nodes to the end-user by calculating the end-to-end delay.

B. Task Arrival at the Fog node and Queue Model

We take a time-slotted system indexed by $t = \{0, 1, , \ldots, t\}$, where the length of each timeslot is Δt. Initially, the system is idle and all queues are empty when $t < 0$. We denote $A_{k,i}^{\text{user},n}(t)$ as the number of offloaded type-n tasks arrived from the kth end-user to the ith primary fog node in the time interval $[t, t+1)$ with task arrival rate $\lambda_{k,i}^{\text{user},n} = \mathbb{E}[A_{k,i}^{\text{user},n}(t)]$. We further denote $A_{j,i}^{\text{fog},n}(t)$ as the number of offloaded type-n tasks from the jth neighboring fog node to the ith fog node in the time interval $[t, t+1)$. We consider that each fog node maintains two virtual queues, namely *high-priority* queue and *low-priority* queue, as depicted in Fig. 1. The priorities of the tasks are determined based on their respective delay-deadlines. Namely, the low-priority tasks can tolerate longer delay compared to the high-priority tasks.

Let M_i be the number of neighboring fog nodes that may offload their tasks to the ith fog node. Denote $Q_i^{\mathsf{X}}(t)$ as the number of tasks queueing in the ith fog node's X-queue (here, $\mathsf{X} = \mathsf{H}$ and L refer to the *high-priority* queue and *low-priority* queue, respectively) at the beginning of the timeslot t and can be expressed as follows:

$$Q_i^{\mathsf{X}}(t) = \sum_{k=1}^{K_i} Q_{k,i}^{\text{user},n,\mathsf{X}}(t) + \sum_{j=1}^{M_i} Q_{j,i}^{\text{fog},n,\mathsf{X}}(t),\ \mathsf{X} \in \{\mathsf{H}, \mathsf{L}\}\ , \quad (1)$$

where $Q_{k,i}^{\text{user},n,\mathsf{X}}(t)$ and $Q_{j,i}^{\text{fog},n,\mathsf{X}}(t)$ are the number of type-n tasks queueing in the ith fog node's queue-X from the kth end-user and the number of type-n tasks queueing in the ith fog node's queue-X from the jth fog node, respectively, at the beginning of time t. As a result, the queue length in the queue-X of the ith fog node will evolve as

$$Q_i^{\mathsf{X}}(t+1) = \max[Q_i^{\mathsf{X}}(t) + A_i^{\mathsf{X}}(t) - B_i^{\mathsf{X}}(t), 0]\ , \quad (2)$$

where $A_i^{\mathsf{X}}(t)$ and $B_i^{\mathsf{X}}(t)$ are the number of tasks arrived at the queue-X in the ith fog node and left (i.e., locally processed) from the queue-X in the ith fog node during the time interval $[t, t+1)$, respectively.

C. Local Task Processing and Offloading to Other Fog Nodes

1) Local task processing at the fog node: The local execution delay for a single task in the ith fog node is simply expressed as $\tau_i = L\, d/f_i$, where d is the task data size and f_i denotes the CPU clock speed (in cycles/s) of the ith fog node. We denote $\mu_i = 1/\tau_i$ as the service rate of the ith fog node.

2) Offloading: We consider that the fog node uses *orthogonal* bands to offload the task data to the neighboring fog nodes as in 4G cellular networks [8]. Denote $r_{i,j}(t)$ as the offloading rate between the ith primary fog node and the jth neighboring fog node during the time interval $[t, t+1)$ and is expressed as $r_{i,j}(t) = x_i \log_2(1 + p_i\, g_{i,j}(t)/x_i n_0)$, where x_i is the allocated bandwidth for the ith fog node, p_i denotes the transmit power for user for node i, $g_{i,j}(t)$ is the channel gain between the ith primary fog node and the jth neighboring fog node during the time interval $[t, t+1)$, and n_0 is the power spectral density of the additive white Gaussian noise. Here, we consider equal x_i allocation for simplicity, i.e., $x_i = W/N$, where W is the total channel bandwidth.

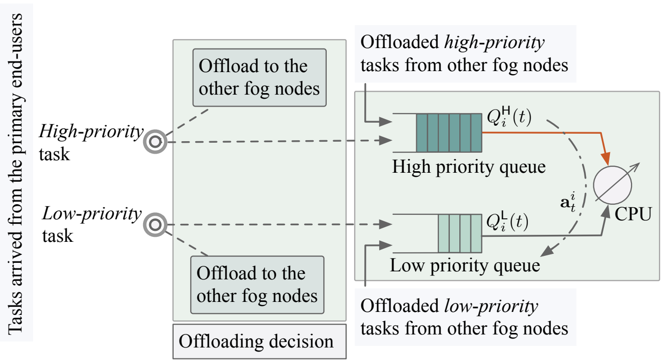
Tasks arrived from the primary end-users
High-priority task
Low-priority task
Offload to the other fog nodes
Offload to the other fog nodes
Offloading decision
Offloaded high-priority tasks from other fog nodes
Offloaded low-priority tasks from other fog nodes
$Q_i^{\mathsf{H}}(t)$
High priority queue
$Q_i^{\mathsf{L}}(t)$
Low priority queue
$\mathbf{a}_t^i$
CPU

Fig. 1. Illustration of queueing model in the ith primary fog node. Assume that the offloaded task from the end-users can be offloaded from a fog node *just once*, hence, a fog node cannot further offload the tasks that arrived from the neighboring fog nodes.

III. Proposed Algorithm

A. Problem Formulation

Denote $A_{k,i}^{\text{user},n,\mathsf{X}}(t)$ and $A_{k,i,j}^{\text{user},n,\text{offload}}(t)$ as the number of type-n tasks offloaded from the kth end-users entering the ith fog node's local queue-X and number of kth end-user's type-n tasks to be offloaded from the ith primary fog node to the jth neighboring fog node. Therefore, we write

$$A_{k,i}^{\text{user},n}(t) = \sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}} A_{k,i}^{\text{user},n,\mathsf{X}}(t) + A_{k,i,j}^{\text{user},n,\text{offload}}(t) \quad (3)$$

and

$$A_i^{\mathsf{X}}(t) = \sum_{k=1}^{K_i} A_{k,i}^{\text{user},n,\mathsf{X}}(t) + \sum_{j=1}^{M_i} \underbrace{A_{k',j,i}^{\text{user},n,\text{offload}}(t)}_{\text{via other fog nodes}}, \forall k' \in \mathcal{K} \setminus \mathcal{K}_i. \quad (4)$$

To reduce the system complexity, it is further assumed that an end-user uniformly and randomly generates only one task at a time from a set of two tasks with different delay-deadlines. We assume that the type-1 task is more stringent than the type-2 task, i.e., $\tau_1 < \tau_2$. We have the following cases when the task is offloaded to the ith fog node at the beginning of the timeslot t:

- *When a type-1 (or type-2) task is offloaded from the jth neighboring fog node to the ith fog node:* The task is sent to the queue-H (or queue-L) in the ith fog node.
- *When a type-1 task is offloaded directly from the end-user:* If $Q_i^{\mathsf{H}}(t)/\mu_i$ is lower than τ_1, then type-1 task is sent to the queue-H. Otherwise, the ith fog node tries to offload the task to the other fog node. Now, if the combined transmission time from the ith fog node to the

Algorithm 1: Buffering and scheduling algorithm

```
1 begin
2     for $u = 1$ to $A_i^{\mathsf{X}}(t)$ do
3         Find a position $v$ in the $i$th fog node's queue-X that satisfies: $\mathfrak{T}_{i,v}^{\mathsf{X}}(t) \le \mathfrak{T}_u(t) \le \mathfrak{T}_{i,(v+1)}^{\mathsf{X}}(t)$;
4         Insert the $u$th task after the $v$th position in the queue $Q_i^{\mathsf{X}}(t) = Q_i^{\mathsf{X}}(t) + 1$ ;   /* buffering algorithm */
      end
5     Based on the Theorem 1, we observe the present $\mathbf{Q}_i(t)$ and choose $B_i^*(a_t, \mathsf{X})$ to maximize
      $\mathbb{E}\Big[\sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}} Q_i^{\mathsf{X}}(t) B_i(a_t,\mathsf{X}) \Big| \mathbf{Q}_i(t)\Big)\Big] - V\,\mathbb{E}\big[P(\mathbf{a}_t^i)\big|\mathbf{Q}_i(t)\big]$ ;
      /* scheduling algorithm */
  end
```

jth fog node and queue-H length of the jth fog node is lower than the queue-H length of the ith fog node, then the type-1 task is offloaded to the jth fog node, otherwise the task enters to the ith fog node's queue-H.

- *When a type-2 task is offloaded from the end-user:* If $\big(Q_i^{\mathsf{H}}(t) + Q_i^{\mathsf{L}}(t)\big)/\mu_i < \tau_2$, then the type-2 task enters the queue-L, otherwise, it is offloaded to the other fog node. The task will select the fog node that provides the highest transmission rate among the possible set of fog nodes with lower queue-L length compared to the ith fog node's queue-L length.

We assume same arrival rate for the tasks offloaded from the end-users to the fog node. We further consider that the service rate of all fog nodes is higher or equal to the arrival rate of the end-user tasks, i.e., $\sum_{i=1}^{N}\sum_{k=1}^{K_i}\big(\lambda_{k,i}^{\text{user},1} + \lambda_{k,i}^{\text{user},2}\big) \le \sum_{i=1}^{N}\mu_i$.

B. Insertion of tasks in a queue

When a new task u of type-n arrives at the beginning of time t in the ith fog node's queue-X, then Algorithm 1 is initiated to buffer the new task based on the *weighted remaining lifetime* [9] that is expressed as $\mathfrak{T}_u(t) = \big(\tau_n - \tau'_u(t)\big)/\tau_n$, where $\tau'_u(t)$ denotes the amount of time spent by the task u over the network up to the beginning of timeslot t. We further denote $\mathfrak{T}_{i,v}^{\mathsf{X}}(t)$ as the weighted remaining lifetime at the timeslot t for the task at the vth position of the ith fog node's queue-X.

C. Scheduling of the queues

We denote $B_i(\mathbf{a}_t^i, \mathsf{X})$ as the number of tasks processed at the ith fog node's queue-X under the scheduling policy $\mathbf{a}_t^i$ at timeslot t. We aim to design the scheduling policy that is ergodic with well defined steady state averages.

Lyapunov drift for stability: Considering both queue-H and queue-L, the ith fog node is *strongly stable*, if

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big[Q_i^{\mathsf{X}}(\tau)\big] < \infty\,, \quad \mathsf{X}\in\{\mathsf{H},\mathsf{L}\}\,. \tag{5}$$

Moreover, the network is *strongly stable* if all individual ith fog node's queues of the network are strongly stable.

Lyapunov drift-plus-penalty: Let $\mathbf{Q}_i(t) \triangleq \big(Q_i^{\mathsf{H}}(t), Q_i^{\mathsf{L}}(t)\big)$ be a stochastic queue-length vector with real-valued components. We define the following *quadratic Lyapunov function* [9], [10] on $\mathbf{Q}_i(t)$ as a scalar measure of the aggregate congestion of two queues in the ith fog node

$$L(\mathbf{Q}_i(t)) \triangleq \frac{1}{2}\Big[Q_i^{\mathsf{H}}(t)^2 + Q_i^{\mathsf{L}}(t)^2\Big] \tag{6}$$

Define $\varDelta\,L(\mathbf{Q}_i(t))$ as the one-step *conditional Lyapunov drift* and is expressed as

$$\varDelta\,L(\mathbf{Q}_i(t)) \triangleq \mathbb{E}\big[L(\mathbf{Q}_i(t+1)) - L(\mathbf{Q}_i(t))|\mathbf{Q}_i(t)\big] \tag{7}$$

Lemma 1. *Under any scheduling policy, the Lyapunov drift must satisfies the following for every timeslot t as*

$$\varDelta\,L(\mathbf{Q}_i(t)) \le B + \sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}} Q_i^{\mathsf{X}}(t)\,\lambda_i^{\mathsf{X}} - \mathbb{E}\Bigg[\sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}} Q_i^{\mathsf{X}}(t)B_i(a_t,\mathsf{X})\Big|\mathbf{Q}_i(t)\Big)\Bigg], \tag{8}$$

where B is a positive constant.

Proof. From (2), we obtain the following bound for each ith queue in the networks as

$$Q_i^{\mathsf{X}}(t+1)^2 \le \Big(Q_i^{\mathsf{X}}(t) + A_i^{\mathsf{X}}(t) - B_i^{\mathsf{X}}(t)\Big)^2\,. \tag{9}$$

Therefore,

$$\frac{1}{2}\sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}} Q_i^{\mathsf{X}}(t+1)^2 \le \frac{1}{2}\sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}} Q_i^{\mathsf{X}}(t)^2 + \frac{1}{2}\sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}}\Big(A_i^{\mathsf{X}}(t) - B_i^{\mathsf{X}}(t)\Big)^2 + \sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}} Q_i^{\mathsf{X}}(t)\Big(A_i^{\mathsf{X}}(t) - B_i^{\mathsf{X}}(t)\Big)\,.$$

It follows that

$$\varDelta\,L(\mathbf{Q}_i(t)) \le \frac{1}{2}\sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}}\mathbb{E}\Big[\Big(A_i^{\mathsf{X}}(t) - B_i^{\mathsf{X}}(t)\Big)^2\Big|\mathbf{Q}_i(t)\Big] + \sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}} Q_i^{\mathsf{X}}(t)\,\mathbb{E}\Big[A_i^{\mathsf{X}}(t) - B_i^{\mathsf{X}}(t)\Big|\mathbf{Q}_i(t)\Big]\,. \tag{10}$$

Since $\lambda_i^{\mathsf{X}} = \mathbb{E}\big[A_i^{\mathsf{X}}(t)\big]$ and $B_i^{\mathsf{X}}(t) \le \mu_i$, we have $(\lambda_i^{\mathsf{X}} - \mu_i)^2 \le \max\Big[\big(\lambda_i^{\mathsf{X}}\big)^2, \big(\mu_i - \lambda_i^{\mathsf{X}}\big)^2\Big]$. Let $B = \max\Big[\frac{1}{2}\big(\lambda_i^{\mathsf{X}}\big)^2, \frac{1}{2}\big(\mu_i - \lambda_i^{\mathsf{X}}\big)^2\Big]$ and substituting in (11), we obtain the following

$$\varDelta\,L(\mathbf{Q}_i(t)) \le B + \sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}} Q_i^{\mathsf{X}}(t)\,\mathbb{E}\Big[A_i^{\mathsf{X}}(t) - B_i^{\mathsf{X}}(t)\Big|\mathbf{Q}_i(t)\Big]\,.$$

Finally, as λ_i^{X} is independent of $\mathbf{Q}_i(t)$, we write

$$\sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}} Q_i^{\mathsf{X}}(t)\,\mathbb{E}\Big[A_i^{\mathsf{X}}(t) - B_i^{\mathsf{X}}(t)\Big|\mathbf{Q}_i(t)\Big] = \sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}} Q_i^{\mathsf{X}}(t)\,\lambda_i^{\mathsf{X}} - \mathbb{E}\Big[\sum_{\mathsf{X}\in\{\mathsf{H},\mathsf{L}\}} Q_i^{\mathsf{X}}(t)B_i(a_t,\mathsf{X})\Big|\mathbf{Q}_i(t)\Big)\Big].$$

Then, the statement follows. ■

Particularly, the inequality in (8) satisfies the stability of the both queues in the ith fog node, however, it ignores the priority

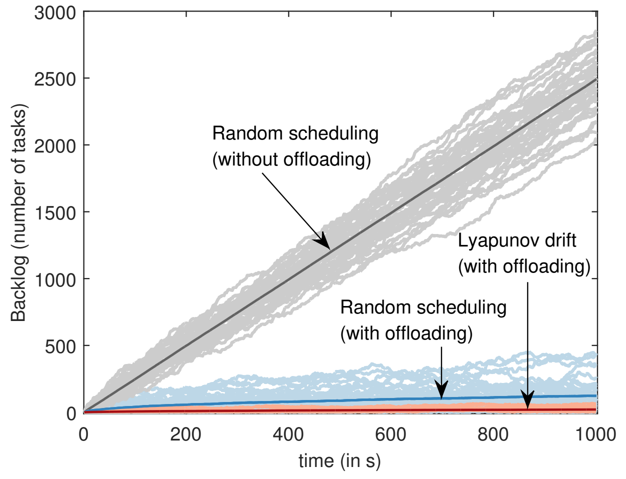

Fig. 2. *Random scheduling without offload:* when the tasks from end users are computed in a standalone fog node (no offloading to other fog nodes) with random scheduling of the queues in the fog node, *random scheduling with offload:* when the fog collaboration with ($M_i = 4$) is assumed with random scheduling of the queues.

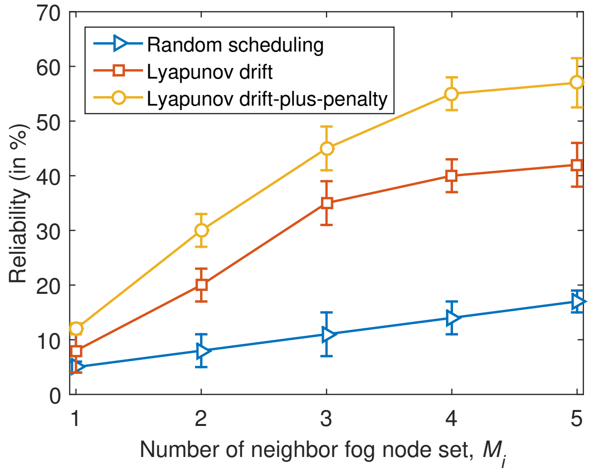

Fig. 3. Performance of reliability, $V = 50, \tau_1 = 1$ s, and, $\tau_2 = 2$ s.

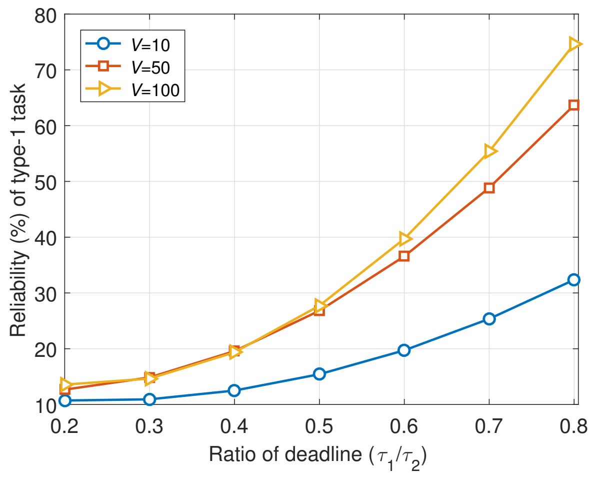

Fig. 4. Performance of reliability, we fix $\tau_2 = 2$ s, $M_1 = 2$.

based on the deadline. Thus, to introduce the task's delay-deadline, we define the Lyapunov *drift-plus-penalty* function on $\mathbf{Q}_i(t)$ as $\Delta L(\mathbf{Q}_i(t)) + V\,\mathbb{E}\big[P(\mathbf{a}_t^i)\big|\mathbf{Q}_i(t)\big]$, where $P(\mathbf{a}_t^i)$ is the penalty function and V is a non-negative control parameter that will affect the delay-deadline priority of the tasks and steady state of the queues tradeoff. We express the penalty function under a scheduling policy $\mathbf{a}_t^i$ in the ith fog node at timeslot t to consider the delay-deadline priority as

$$P(\mathbf{a}_t^i) = \sum_{X\in\{H,L\}} \sum_{u}^{Q_i^X(t)} \frac{\tau_u'(t) + \tau_u^{\text{wait}}(\mathbf{a}_t^i)}{\tau_n}, \tag{11}$$

where $\tau_u^{\text{wait}}(\mathbf{a}_t^i)$ is the expected waiting time for the uth task in the ith fog node's queue under the current scheduling policy $\mathbf{a}_t^i$ at timeslot t. Particularly, we aim to minimize the penalty function that is the *weighted total waiting time* in the ith fog node's queue under scheduling policy $\mathbf{a}_t^i$. Moreover, it is important to note that the above penalty function also considers the respective deadline of the type-n task (i.e., τ_n) to calculate the *weighted total waiting time*.

Theorem 1. *Under any scheduling policy, the Lyapunov drift-plus-penalty must satisfies the following for every $\mathbf{Q}_i(t)$ as*

$$\Delta L(\mathbf{Q}_i(t)) + V\,\mathbb{E}\big[P(\mathbf{a}_t^i)\big|\mathbf{Q}_i(t)\big] \le\ B + \sum_{X\in\{H,L\}} Q_i^X(t)\,\lambda_i^X$$
$$+ V\,\mathbb{E}\big[P(\mathbf{a}_t^i)\big|\mathbf{Q}_i(t)\big] - \mathbb{E}\Bigg[\sum_{X\in\{H,L\}} Q_i^X(t) B_i(a_t, X)\Big|\mathbf{Q}_i(t)\Bigg)\Bigg].$$

Proof. Adding $V\,\mathbb{E}\big[P(\mathbf{a}_t^i)\big|\mathbf{Q}_i(t)\big]$ in the both side of the Lemma 1 concludes the proof. ■

IV. Simulation Results

In the simulation setup, we take $N = 10$, $K = 20$, $W = 1$ MHz, noise power $n_0 = -174$ dBm/Hz, $p_i = 0.1$ Watt. Besides, $g_{i,j}(t)$ is uniformly distributed over $[-50, -30]$ dBm, $L = 297.62$ [cycles/bit], each $d = 4096$ bit, $f_i = 4 \times 10^9$ [cycles/s] [2], $\lambda_{k,i}^{\text{user},n} = 3$ packet/s. The results are averaged over $10,000$ different runs with Monte Carlo simulations.

Fig. 2 illustrates that applying Lyapunov drift at the scheduling maintains the stability of the queues, whereas the backlog size increases with random scheduling. Moreover, the *reliability* is expressed as to interpret *how many tasks meet their deadlines* in the current scheduling process. From Fig. 3, we see that the scheduling with only with Lyapunov drift outperforms the random scheduling in term of reliability. It is further observed that the use of penalty in Lyapunov drift-plus-penalty function (see, (10)), has a positive impact on reliability maximization and significantly improves the reliability performance. However, as the resource utilization in term of service rate to support the task arrival rate reaches to its maximum point, the further increasing of neighboring fog node does not further improve the reliability.

Finally, Fig. 4 shows how the deadline ratio (τ_1/τ_2) between the tasks affects the reliability performance of the high-priority tasks. The deadline ratio indicates the difference between the deadlines of high- and low-priority tasks. When τ_1/τ_2 is low, i.e., the high-priority task has much stringent deadline, the number of high-priority tasks completed within the deadlines is small, i.e., low reliability. When τ_1/τ_2 is increased, i.e., the deadline of the high-priority task gets relaxed, it becomes easier for the system to satisfy the high-priority tasks deadline requirements, i.e., high reliability. This result is expected. What is interesting to see in the figure is that controlling parameter V can affect the reliability, which means that we can adjust V to improve the reliability of high-priority tasks.

V. Conclusion

In this letter, we studied the optimum scheduling policy of the two queues system in a fog node that allows a higher number of deadline-aware offloaded tasks to be processed while making both queue stable. The proposed approach is decoupled into two strategies: the priority-aware scheduling policy by applying Lyapunov drift-plus-penalty function and the fog nodes collaboration based on each fog node's queueing status under the scheduling policy. Furthermore, the simulation results suggest that under the same resource configuration, our scheme can guarantee as more as tasks completion within the deadlines. To find the optimum scheduling policies under different resource configurations with a combination of reward-penalty is a part of future work.

References

[1] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132–4150, June 2019.

[2] X. Gao, X. Huang, S. Bian, Z. Shao, and Y. Yang, "PORA: Predictive offloading and resource allocation in dynamic fog computing systems," in *Proc. IEEE ICC*, May 2019, pp. 1–6.

[3] H. Cheng, Z. Su, J. Lloret, and G. Chen, "Service-oriented node scheduling scheme for wireless sensor networks using markov random field model," *Sensors*, vol. 14, no. 11, pp. 20 940–20 962, Nov. 2014.

[4] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative task offloading in three-tier mobile computing networks: An ADMM framework," *IEEE Trans. on Vehi. Technol.*, vol. 68, no. 3, pp. 2763–2776, Mar 2019.

[5] M. Mukherjee, S. Kumar, M. Shojafar, Q. Zhang, and C. X. Mavromoustakis, "Joint task offloading and resource allocation for delay-sensitive fog networks," in *Proc. IEEE ICC*, May 2019, pp. 1–7.

[6] J. J. Rodrigues, L. Zhou, L. D. Mendes, K. Lin, and J. Lloret, "Distributed media-aware flow scheduling in cloud computing environment," *Computer Commun.*, vol. 35, no. 15, pp. 1819–1827, Sept. 2012.

[7] C.-F. Liu, M. Bennis, and H. V. Poor, "Latency and reliability-aware task offloading and resource allocation for mobile edge computing," in *Proc. IEEE GLOBECOM Workshops*, Dec. 2017.

[8] J. Huang, V. Subramanian, R. Agrawal, and R. Berry, "Joint scheduling and resource allocation in uplink OFDM systems for broadband wireless access networks," *IEEE J. Select. Areas in Commun.*, vol. 27, no. 2, pp. 226–234, Feb. 2009.

[9] M. Guo, Q. Guan, W. Chen, F. Ji, and Z. Peng, "Delay-optimal scheduling of VMs in a queueing cloud computing system with heterogeneous workloads," *IEEE Trans. Services Comput.*, pp. 1–14, 2019.

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