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 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 \(\tau_n\) as the delay-deadline for the task type-\(n\).

Assumption: Assuming a typical binary offloading scenario, 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 \(K_i = \{1, 2, \ldots, K_i\}\), \(\sum_{i=1}^{N} K_i = K\) as the set of end-users that select the \(i\)th fog node as their primary fog node, we write \(K_i \cap K_{i'} = \emptyset\) 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 nodes.

1In 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 \( \Delta t \). Initially, the system is idle and all queues are empty when \( t < 0 \). We denote \( A_{ki}^{\text{user}, n}(t) \) as the number of offloaded type-\( n \) tasks arrived from the \( k \)th end-user to the \( i \)th primary fog node in the time interval \([t, t+1)\) with task arrival rate \( A_{ki}^{\text{user}, n}(t) = E[A_{ki}^{\text{user}, n}(t)] \). We further denote \( A_{j,i}^{\text{fog}, n}(t) \) as the number of offloaded type-\( n \) tasks from the \( j \)th neighboring fog node to the \( i \)th 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 \( i \)th fog node. Denote \( Q_{ki}^X(t) \) as the number of tasks queuing in the \( i \)th fog node’s \( X \)-queue (here, \( X = H \) 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:

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

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

\[
Q_{ki}^X(t+1) = \max\{Q_{ki}^X(t) + A_{ki}^X(t) - B_{ki}^X(t), 0\},
\]

where \( A_{ki}^X(t) \) and \( B_{ki}^X(t) \) are the number of tasks arrived at the queue-\( X \) in the \( i \)th fog node and left (i.e., locally processed) from the queue-\( X \) in the \( i \)th 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 \( i \)th fog node is simply expressed as \( \tau_i = L/d_j \), where \( L \) is the task data size and \( d_j \) denotes the CPU clock speed (in cycles/s) of the \( i \)th fog node. We denote \( \mu_i = 1/\tau_i \) as the service rate of the \( i \)th 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 \( i \)th primary fog node and the \( j \)th 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 \( i \)th fog node, \( p_i \) denotes the transmit power for user for node \( i \), \( g_{i,j}(t) \) is the channel gain between the \( i \)th primary fog node and the \( j \)th 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.

III. PROPOSED ALGORITHM

A. Problem Formulation

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

\[
A_{ki}^{\text{user}, n}(t) = \sum_{X \in \{H, L\}} A_{ki}^{\text{user}, n, X}(t) + A_{ki}^{\text{user}, n, \text{offload}}(t)
\]

and

\[
A_{ki}^{\text{fog}, n}(t) = \sum_{k=1}^{K_i} A_{ki}^{\text{user}, n, X}(t) + \sum_{j=1}^{M_i} A_{ki}^{\text{user}, n, \text{offload}}(t), \forall k' \in K \setminus K_i.
\]

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 \( i \)th fog node at the beginning of the timeslot \( t \):

- **When a type-1 (or type-2) task is offloaded from the \( j \)th neighboring fog node to the \( i \)th fog node:** The task is sent to the queue-\( H \) (or queue-\( L \)) in the \( i \)th fog node.
- **When a type-1 task is offloaded directly from the end-user:** If \( Q_{ki}^H(t)/\mu_i \) is lower than \( \tau_1 \), then type-1 task is sent to the queue-\( H \). Otherwise, the \( i \)th fog node tries to offload the task to the other fog node. Now, if the combined transmission time from the \( i \)th fog node to the other fog node...
Algorithm 1: Buffering and scheduling algorithm

1. begin
2. for \( u = 1 \) to \( \lambda^X_i(t) \) do
3. Find a position \( v \) in the \( i \)th fog node’s queue-\( X \) that satisfies:
4. \( Q^X_{i,v}(t) \leq \sum_{\tau_i} Q^X_{i}((\tau_i+1);t) \).
5. Insert the \( \tau_i \) task after the \( v \)th position in the queue
6. \( Q^X_{i,v}(t) = Q^X_{i,v}(t) + 1 \); /* buffering algorithm */
7. end
8. for \( u = 1 \) to \( \lambda^X_i(t) \) do
9. Choose \( B^*_i(a_t, X) \) to maximize
10. \( \mathbb{E}[\sum_{X} Q^X_{i}(t) B_i(a_t, X)|Q_i(t)] - \mathbb{E}[P(a)|Q_i(t)] ; /* scheduling algorithm */
11. end

Given \( j \)th fog node and queue-\( H \) length of the \( j \)th fog node is lower than the queue-\( H \) length of the \( i \)th fog node, then the type-1 task is offloaded to the \( j \)th fog node, otherwise the task enters to the \( i \)th fog node’s queue-\( H \).

- When a type-2 task is offloaded from the end-user: If \( (Q^H_i(t) + Q^L_i(t))/\mu_i < \tau_2 \), 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 \( i \)th 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., \( \sum_{i=1}^{N} \sum_{k=1}^{K_i} (\lambda_{k,i}^{\text{user},1} + \lambda_{k,i}^{\text{user,2}}) \leq \sum_{i=1}^{N} \mu_i \).

B. Insertion of tasks in a queue

When a new task \( u \) of type-\( n \) arrives at the beginning of time \( t \) in the \( i \)th fog node’s queue-\( X \), then Algorithm 2 is initiated to buffer the new task based on the remaining lifetime [9] that is expressed as \( \sum_{\tau_i} (\tau_n - \tau_n(t))/\tau_n \), where \( \tau_n(t) \) denotes the amount of time spent by the task \( u \) over the network up to the beginning of timeslot \( t \). We further denote \( Q^X_{i,v}(t) \) as the weighted remaining lifetime at the timeslot \( t \) for the task at the \( v \)th position of the \( i \)th fog node’s queue-\( X \).

C. Scheduling of the queues

We denote \( B_i(a_t, X) \) as the number of tasks processed at the \( i \)th fog node’s queue-\( X \) under the scheduling policy \( a_t \) 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 \( i \)th fog node is strongly stable, if

\[
\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau = 0}^{t-1} \mathbb{E}[Q^X_i(\tau)] < \infty, \quad X \in \{H, L\}.
\]

Moreover, the network is strongly stable if all individual \( i \)th fog node’s queues of the network are strongly stable.

**Lyapunov drift-plus-penalty**: Let \( Q_i(t) = (Q^H_i(t), Q^L_i(t)) \) be a stochastic queue-length vector with real-valued components. We define the following quadratic Lyapunov function [9], [10] on \( Q_i(t) \) as a scalar measure of the aggregate congestion of two queues in the \( i \)th fog node

\[
L(Q_i(t)) \triangleq \frac{1}{2} \left[ Q^H_i(t)^2 + Q^L_i(t)^2 \right]
\]

Define \( \Delta L(Q_i(t)) \) as the one-step conditional Lyapunov drift and is expressed as

\[
\Delta L(Q_i(t)) \triangleq \mathbb{E}[L(Q_i(t+1)) - L(Q_i(t)) | Q_i(t)]
\]

**Lemma 1.** Under any scheduling policy, the Lyapunov drift must satisfy the following for every timeslot \( t \) as

\[
\Delta L(Q_i(t)) \leq B + \sum_{X \in \{H,L\}} Q^X_i(t) \lambda^X_i
\]

\[
- \mathbb{E} \left[ \sum_{X \in \{H,L\}} Q^X_i(t) B_i(a_t, X) | Q_i(t) \right],
\]

where \( B \) is a positive constant.

**Proof.** From [2], we obtain the following bound for each \( i \)th queue in the networks as

\[
Q^X_i(t+1)^2 \leq \left( Q^X_i(t) + A^X_i(t) - B^X_i(t) \right)^2.
\]

Therefore,

\[
\frac{1}{2} \sum_{X \in \{H,L\}} Q^X_i(t+1)^2 \leq \frac{1}{2} \sum_{X \in \{H,L\}} Q^X_i(t)^2 + \frac{1}{2} \sum_{X \in \{H,L\}} \left( A^X_i(t) - B^X_i(t) \right)^2 + \sum_{X \in \{H,L\}} Q^X_i(t) (A^X_i(t) - B^X_i(t)).
\]

It follows that

\[
\Delta L(Q_i(t)) \leq \frac{1}{2} \sum_{X \in \{H,L\}} \mathbb{E} \left[ \left( A^X_i(t) - B^X_i(t) \right)^2 | Q_i(t) \right] + \sum_{X \in \{H,L\}} Q^X_i(t) \mathbb{E} \left[ A^X_i(t) - B^X_i(t) | Q_i(t) \right].
\]

Since \( \lambda^X_i = \mathbb{E}[A^X_i(t)] \) and \( B^X_i(t) \leq \mu_i \), we have

\[
(\lambda^X_i - \mu_i)^2 \leq \max \left( (\lambda^X_i)^2, (\mu_i - \lambda^X_i)^2 \right).
\]

Let \( B = \max \left( \frac{1}{2} (\lambda^X_i)^2, \frac{1}{2} (\mu_i - \lambda^X_i)^2 \right) \) and substituting in (11), we obtain the following

\[
\Delta L(Q_i(t)) \leq B + \sum_{X \in \{H,L\}} Q^X_i(t) \mathbb{E} \left[ A^X_i(t) - B^X_i(t) | Q_i(t) \right].
\]

Finally, as \( \lambda^X_i \) is independent of \( Q_i(t) \), we write

\[
\sum_{X \in \{H,L\}} Q^X_i(t) \mathbb{E} \left[ A^X_i(t) - B^X_i(t) | Q_i(t) \right] = \sum_{X \in \{H,L\}} Q^X_i(t) \lambda^X_i
\]

\[
- \mathbb{E} \left[ \sum_{X \in \{H,L\}} Q^X_i(t) B_i(a_t, X) | Q_i(t) \right].
\]

Then, the statement follows.
based on the deadline. Thus, to introduce the task’s delay-deadline, we define the Lyapunov drift-plus-penalty function on \( Q_i(t) \) as \( \Delta L(Q_i(t)) + V \mathbb{E}[P(a_i^i)]Q_i(t) \), where \( P(a_i^i) \) 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 \( a_i^i \) in the \( i \)th fog node at timeslot \( t \) to consider the delay-deadline priority as

\[
P(a_i^i) = \sum_{X \in \{H,L\}} Q_i^X(t) \tau_u'(t) + \tau_u^{\text{wait}}(a_i^i), \tag{11}
\]

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

**Theorem 1.** Under any scheduling policy, the Lyapunov drift-plus-penalty must satisfies the following for every \( Q_i(t) \) as

\[
\Delta L(Q_i(t)) + V \mathbb{E}[P(a_i^i)]Q_i(t) \leq B + \sum_{X \in \{H,L\}} Q_i^X(t) \lambda_i^X + V \mathbb{E}[P(a_i^i)]Q_i(t) - \mathbb{E} \left[ \sum_{X \in \{H,L\}} Q_i^X(t) B_i(a_i^i, X | Q_i(t)) \right].
\]

**Proof.** Adding \( V \mathbb{E}[P(a_i^i)]Q_i(t) \) in the both side of the Lemma 1 concludes the proof. \( \blacksquare \)

### 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,

\[
L = 297.62 \text{ [cycles/bit]}, \quad d = 4096 \text{ bit}, \quad f_i = 4 \times 10^9 \text{ [cycles/s]}. \quad \lambda_{k,i}^{\text{tar},n} = 3 \text{ packet/s}.
\]

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 \( (\tau_1/\tau_2) \) between the tasks affects the reliability performance of the high-priority tasks. The deadline ratio indicates the difference between the

---

**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, \( M_1 = 2 \).
deadlines of high- and low-priority tasks. When $\tau_1/\tau_2$ 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 $\tau_1/\tau_2$ 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