Energy-Efficient Computation Offloading in Collaborative Edge Computing

Lin, Rongping; Xie, Tianze; Luo, Shan; Zhang, Xiaoning; Xiao, Yong; Moran, Bill; Zukerman, Moshe

Published in:
IEEE Internet of Things Journal

Published: 01/11/2022

Document Version:
Post-print, also known as Accepted Author Manuscript, Peer-reviewed or Author Final version

Publication record in CityU Scholars:
Go to record

Published version (DOI):
10.1109/JIOT.2022.3179000

Publication details:

Citing this paper
Please note that where the full-text provided on CityU Scholars is the Post-print version (also known as Accepted Author Manuscript, Peer-reviewed or Author Final version), it may differ from the Final Published version. When citing, ensure that you check and use the publisher's definitive version for pagination and other details.

General rights
Copyright for the publications made accessible via the CityU Scholars portal is retained by the author(s) and/or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights. Users may not further distribute the material or use it for any profit-making activity or commercial gain.

Publisher permission
Permission for previously published items are in accordance with publisher's copyright policies sourced from the SHERPA RoMEO database. Links to full text versions (either Published or Post-print) are only available if corresponding publishers allow open access.

Take down policy
Contact lbscholars@cityu.edu.hk if you believe that this document breaches copyright and provide us with details. We will remove access to the work immediately and investigate your claim.
Abstract—Edge computing is an indispensable technology that overcomes delay limitations of cloud computing. In edge computing, computational resources are deployed at the network edge, and computational tasks and data of end terminals can be efficiently processed by edge nodes. Considering the computational resource limitations of edge nodes, collaborative edge computing integrates computational resources of edge nodes and provides more efficient computing services for end terminals. This paper considers a computation offloading problem in collaborative edge computing networks, where computation offloading and resource allocation are optimized by means of a collaborative load shedding approach: a terminal can offload a computing task to an edge node, which either can process the task with its computing resource or further offload the task to other edge nodes. Long-term objectives and long-term constraints are considered, and Lyapunov optimization is applied to convert the original non-convex computation offloading problem into a second problem that approximate the original problem and it is still non-convex but has a special structure, which gives rise to a new distributed algorithm that optimally solves the second problem. Finally, the performance and provable bound of the distributed algorithm is theoretically analyzed. Numerical results demonstrate that the distributed algorithm can achieve a guaranteed long-term performance, and also demonstrate the improvement in performance achieved over the case of computation offloading without collaborating edge nodes.

Index Terms—Computation offloading, collaborative edge computing, Lyapunov optimization

I. INTRODUCTION

Edge computing is a promising complement to cloud computing [1], [2], which provides computational resources at the edge of the network, e.g., at base stations [3] and access points [4]. With this new computing paradigm, end users can offload computational tasks to edge nodes for processing instead of sending them to the cloud. This reduces the need to transmit data to the cloud, and this in turn reduces service response times and mitigates both network congestion and security problems because less data is sent through the network [5]–[7]. A challenge of computation offloading and the related resource allocation problem is that computational resources at the network edge are limited and distributed. This highlights the need for efficient computation offloading and resource allocation. A significant research effort has focused on computation offloading and resource allocation (the term computation offloading usually includes resource allocation). Existing literature can be categorized according to the optimization objectives. Examples include: minimizing response delay [8]–[12], minimizing energy consumption [13]–[15], maximizing network profit or social welfare [3], [16]–[18]. Binary and partial offloading of computational tasks have been also discussed in [19], [20], where the term binary offloading is used when the entire task is offloaded to one edge node or remains in the original terminal [21]–[23], and partial offloading implies that a task is offloaded to multiple locations simultaneously [24] which is the approach adopted by this paper.

The computational resources at the edge nodes are limited as compared to the cloud and, at the same time, more and more computational resources are required by terminals because of the emergence of computation demanding applications such as virtual/augmented reality (VR/AR) [25], machine learning [26], 3D games, and editing video. To satisfy the low computational delay requirements of many applications and increase resource utilizations, task offloading to only one edge node may be insufficient. Instead, computing tasks from a terminal that need processing by more than one single edge node may be offloaded to other computing platforms to provide better overall computational performances. Several existing research publications consider computing task offloading to edge nodes and the cloud [27]–[30]. However, most applications at terminals are time critical and require very low response delays, whereas processing at a geographically distant or hop-distant cloud far from the terminals is suitable only for tasks with very high delay tolerances. For the sake of simplicity, computation offloading to the cloud is not considered in this paper, but the work in this paper can be easily extended if the option of computation offloading to the cloud is appropriate.

Collaborative edge computing that enables cooperative computing among edge nodes is promising in exploiting edge computing, because it can achieve improved system efficiency by further offloading workload from heavily loaded edge nodes to lightly loaded ones. With this computing paradigm, multiple
edge nodes jointly provide satisfactory computing services to terminals, and the resources utilization can be optimized. Computation intensive applications can especially benefit from collaborative edge computing. For example, an intelligent navigation system relies on input data of real-time road conditions [31] and performs a wide range of computational tasks. These applications can offload computation intensive tasks to multiple edge nodes along the route with cooperative computing in collaborative edge computing networks. To implement the collaborative edge computing system, workload offloaded from terminals can be further offloaded to other edge nodes in multiple hops to provide required quality of service (QoS) for terminals and achieve the workload balance among edge nodes. This requires consideration of the additional transmission delay between edge nodes, and efficient resource management and task scheduling, that can be addressed by jointly optimizing resource allocation and offloading decisions of terminals and edge nodes in a distributed way. There is no existing work on distributed algorithm designs of computation offloading for collaborative edge computing, taking account of both edge nodes and terminals in general edge computing networks. In addition, the cooperative computing among edge nodes and terminals is highly affected by the distribution of the network workload which usually varies dynamically, giving rise to the need for adaptive computation offloading and resource allocation. On the other hand, there is a need to conserve long-run energy consumption of terminals. In this paper, motivated by these challenges, we consider a dynamic collaborative edge computing system in a general edge computing network, controlled in a distributed way, and optimize jointly computation offloading decisions for cooperating edge nodes and terminals to minimize the long-term energy consumption.

Optimizing the long-term performance measures of a network system is important because such measures provide an aggregate of short-term (stable or unstable) correlated measures over a long period of time, and such aggregations have important economic and business implications, e.g. long-term cost and long-term user experiences. Such long-term considerations complicate the optimization problem, however, because of the need to take into account fluctuation of resource availability and QoS provisioning, as well as additional long-term constraints. The long-term constraints can be converted to a problem of stabilisation of virtual queues [32], and Lyapunov optimization is a powerful technique that allows the stabilisation of such virtual queues while optimizing a system [33]. In our previous work [34], we applied Lyapunov optimization to develop a distributed algorithm for computation offloading in an edge computing network with one edge node and multiple terminals, to minimize the long-term average response time delay.

To the best of our knowledge, there are still no solutions that minimize the long-term energy consumption of terminals for the computation offloading problem, subject to computational resource and response delay constraints in collaborative edge computing, where a task can be offloaded to multiple edge nodes for simultaneous processing. In this paper, we investigate the computation offloading problem in collaborative edge computing with the objective of minimizing the long-term average energy consumption of terminals, where terminals are subjected to limited battery power and have energy harvesting ability to power the terminals accordingly. We focus on the long-term average energy consumption minimization problem under two types of constraints: the long-term average computational resource and the response time threshold constraints. Such a perspective includes consideration of adaptive resource allocation in a long-term viewpoint to achieve long-term performance, where some performance degradations can be tolerated for short periods. In this paper, we apply Lyapunov optimization to the present problem, and convert the original long-term problem into a deterministic upper bound optimization problem with fluctuations captured by the variation of virtual queues [32]. The main contributions of this paper are as follows.

1) A mathematical model to accommodate the computation offloading problem for multiple terminals and multiple edge nodes is provided for collaborative edge computing networks, where the long-term average energy consumption objective is optimized under long-term computational and response time constraints; the original non-convex problem is converted into a second problem that approximates the original problem and it is still non-convex but has a special structure.

2) By adopting the special structure of the approximation problem, a distributed algorithm is proposed to solve the approximation problem optimally, involving a limited amount of information exchanged between terminals and edge nodes, where each individual terminal determines its own resource allocation and computation offloading.

3) An upper bound for the gap between the proposed distributed algorithm solution and the original optimal solution is derived, and a proof that the distributed algorithm solution satisfies the long-term constraints is provided.

4) Extensive numerical results evaluate the performance of the proposed algorithm and show that it stabilizes the virtual queues and achieves the required performance.

The remainder of this paper is organized as follows. Section II discusses existing publications related to computation offloading in collaborative edge computing. In Section III, we provide system model and problem formulation for the computation offloading problem in a collaborative edge computing network with a long-term average energy consumption minimization objective. In Section IV, we convert the computation offloading problem into an upper bound optimization problem, and the performance gap between the two problems is analyzed. In Section V, the upper bound problem is studied and a distributed algorithm is provided. Section VI numerically evaluates and validates the new algorithm. Section VII concludes this paper.

II. RELATED WORK

In edge computing networks, edge nodes provide distributive and limited computational resources, on the other hand, more computational resources are required by terminals
because of the emergence of applications with extremely high computing demands. To provide an efficient computation service and obtain a high resource utilization, collaborative edge computing attracts a large research effort. In this section, the most relevant research publications that focus on related computation offloading problems for collaborative edge computing networks are surveyed.

Some publications [35]–[39] study offloading to nearby available terminals with acceptable costs or satisfaction of resource constraints. Pu et al. [35] proposed a device-to-device collaboration framework for computational task processing, where devices can dynamically and beneficially share the computation and communication resources with each other. The problem was to minimize time-average energy consumption with long-term incentive constraints. Li et al. [36] investigated an online incentive mechanism for collaborative task offloading in edge computing, where nearby terminals (called collaborators) can help base station processing tasks to maximize the sum of all of the collaborators’ utilities. He et al. [37] proposed an auction-based incentive mechanism for collaborative task offloading in edge computing to maximize long-term utility, defined as the difference between payment received and the cost of service, where an edge node asked its resourceful neighboring terminals to help execute tasks in return for certain rewards. Hu et al. [38] investigated a cooperative computing problem to maximize system computing capacity in device-to-device (D2D) enhanced communications mobile edge networks, where each terminal can split its task into three parts for local computing, D2D offloading to other terminals, and offloading to the edge node. You et al. [39] proposed a cooperative computing method among terminals, referred to as co-computing, where a terminal offloads computational tasks to another terminal. This method increases the computing resource utilization of terminals. However, all of the above methods may cause privacy or security problems because of the processing of data of one user at the terminals of other users.

Several publications consider computation offloading to multiple edge nodes directly from terminals [40]–[43]. Dinh et al. [40] proposed a computation offloading framework where a computational task of a terminal can be offloaded to multiple edge nodes for processing, with an objective to minimize task execution latency and energy consumption of terminals. Tran et al. [41] investigate a computation offloading problem involving multiple users and multiple edge nodes. In their problem, a weighted sum of task completion time and energy consumption was optimized, but with a constraint that a task can be processed at only one selected edge node. Li et al. [42] proposed a computation replication method that allows a task to be directly offloaded to multiple edge nodes for repetitive processing. This can reduce the downlink latency caused by computation results being sent back, where the downlink channel could experience strong interference or deep fading. This method increases transmission load for task uploading, and computational load at edge nodes, resulting in a possible degradation of system performance under heavy computing loads. Malik et al. [43] investigated a computation offloading problem in massive-MIMO wireless edge computing networks, where multiple users simultaneously offload computational tasks to multiple edge nodes, thereby shortening the data offloading time. The weighted sum of the energy consumption at both users and edge nodes is optimized. In that work, a user can offload a computational task to only one selected edge node. To offload computational tasks directly from a terminal to multiple edge nodes, the terminal must be covered by service ranges of the edge nodes, which can limit the number of edge nodes available to collaborate the processing of the task.

The computation offloading problems where an edge node can further offload tasks to collaborative edge nodes for processing with acceptable transmission delays/costs has received some attentions from researchers [44]–[48]. Zhang et al. [44] proposed a hierarchical computation offloading framework in vehicle edge computing, where a backup computational server in the neighborhood was introduced to compensate for the computing resource limitation of edge servers at the roadside. Hu et al. [45] investigated a computation offloading problem in a two-tier computation heterogeneous cellular network (HetNet), equipped with edge clouds at small base stations (SBSs) and central cloud connected by a macro base station (MBS) via optical fiber; the central cloud helped edge clouds with offloaded task processing when necessary. Ebrahimzadeh et al. [46] investigated a computation offloading problem in edge computing and FiWi enhanced HetNets, where optical network units (ONUs) were deployed with computational servers that provide edge computing for end users. A central optical line terminal (OLT) at about 20 km separation from the ONUs at the fiber backhaul was also deployed with computational servers that can assist ONUs to further offload computational tasks. Cao et al. [47] investigated a computation offloading problem to minimize energy consumption in a three-node mobile edge computing system consisting of one user node, one helper node, and one access point node with an integrated computing server. The helper node assists the user node to relay computation data to the access point and to process the computing task with its computational resource. However, all of the above work [44]–[47] are based on the deployment of additional computational resources for extra computation workloads and restricted in allowing an edge node to collaborate with only one other node. Li et al. [49] investigated a trusted computation offloading problem that considers trust risks in collaborative mobile edge computing, where edge nodes can further offload tasks to trusted neighboring edge nodes for a long-term cost minimization. Wang et al. [48] investigated a cooperative computation offloading problem with a sleep mode of edge node (a low power consumption mode for a nonactive device to stand by). They devised a control scheme to minimize energy consumption of the edge computing network, where an edge node can offload its computing tasks to other edge nodes to put an edge node into sleep mode. Xiao et al. [24] worked on a cooperation strategy for computation offloading to further improve service response time. In that work, an edge node can forward a part or all of its offloaded workload to multiple neighboring edge nodes for processing. Although these publications investigated cooperative computing among edge nodes to improve the QoS.
provided by edge computing, there is no work that considers computing by both terminals and edge nodes, where terminals may process computational tasks locally, thereby effecting QoS by, for example, delay.

To the best of our knowledge, none of the existing publications consider distributed algorithms for the computation offloading of both terminals and edge nodes in collaborative edge computing, where a task can be processed simultaneously at multiple edge nodes in addition to the local terminal as considered in this paper. Based on Lyapunov optimization, we convert the original problem into an upper bound problem and design a distributed algorithm to optimally solve the upper bound problem.

III. SYSTEM MODEL AND PROBLEM FORMULATION

The scenario considered in this paper consists of a set of terminals \( \mathcal{N} \) and a set of edge nodes \( \mathcal{M} \) as Fig. 1, where terminals can be mobile phones or IoT devices, and edge nodes can be any edge devices/servers with computing ability, such as a router, a switch, or a base station. We investigate the computation offloading problem for terminals to multiple collaborative edge nodes in this scenario, where the energy efficiencies of terminals are taken into account. Each terminal \( i \in \mathcal{N} \) that connects to an edge node \( \tau_i \) has computational tasks to be processed sequentially. Accordingly, a computational task is defined as \( \text{task}_i = \{D_i(t), Z_i(t)\} \), where \( D_i(t) \) denotes the computation requirement (number of uniform CPU cycles) to accomplish this task and \( Z_i(t) \) denotes the data size associated with the computation requirement in time slot \( t \). We assume that the computation requirement is proportional to the data size which is same as [50]. In the problem formulation, we also assume that every task is served in a single time slot; this entails that tasks with higher computational requirements will use more computational resources. We define \( \alpha_i(t) \) as the proportion of a task that is offloaded to an edge node in time slot \( t \) at terminal \( i \). The proportion of the task being processed by edge nodes is \( \alpha_i(t)D_i(t) \), and the rest of the computational task \((1-\alpha_i(t))D_i(t)\) is processed at the terminal, then there is \( \alpha_i(t)Z_i(t) \) data needed to be transmitted to edge nodes if there is \( \alpha_i(t)D_i(t) \) task offloaded to edge nodes. With computation offloading, a computational task may be processed by the terminal with local computing, or by edge nodes with edge computing, or by a combination of the two. In this problem, we allow different tasks to have different completion time deadlines \( T_{i}^{max}(t) \) to guarantee QoS of applications. This makes the computation offloading problem more challenging. We will discuss the task processing from two viewpoints, local computing and edge computing. The notations used in this paper are defined in Table I.

### A. Local Computing

The computational capacity of a terminal is limited, forcing the computational task to be offloaded to edge nodes for faster processing. However, the limited computational resources of the edge nodes are shared by competing terminals that also rely on the offloading option. It is necessary, then, for the local computing of these terminals to be optimized for a better performance of the overall system, e.g. a specific terminal does not offload because then other terminals can offload to a greater overall advantage. We use \( F_{i1}^{l} \) to denote the number of local computing resources of terminal \( i \) in terms of CPU cycles/s, and assume that all local computing resources are used if there are tasks to be processed. Then, the local computing time of the unoffloaded task is expressed as

\[
DU_{i1}^{l}(t) = \frac{(1-\alpha_{i}(t))D_{i}(t)}{F_{i1}^{l}}.
\]

When the unoffloaded workload is processed by terminal \( i \), following [41], [51], [52], we express the energy consumption of terminal for the computational task as

\[
e_{i1}^{l}(t) = \kappa(F_{i1}^{l})^{3}DU_{i1}^{l}(t) = \kappa(F_{i1}^{l})^{2}D_{i}(t)(1-\alpha_{i}(t)),\]

where \( \kappa \) is the effective energy consumption coefficient.

To offload computational tasks to edge nodes, the data associated with the computational task is transmitted to the edge node. The data transmitted is proportional to the offloaded computational task, so that \( \alpha_i(t)Z_i(t) \) data needs to

### TABLE I

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{N} )</td>
<td>Set of terminals</td>
</tr>
<tr>
<td>( \mathcal{M} )</td>
<td>Set of edge nodes</td>
</tr>
<tr>
<td>( N )</td>
<td>Number of terminals</td>
</tr>
<tr>
<td>( M )</td>
<td>Number of edge nodes</td>
</tr>
<tr>
<td>( \tau_i )</td>
<td>Edge node that serves terminal ( i )</td>
</tr>
<tr>
<td>( D_i(t) )</td>
<td>Computational requirement of terminal ( i )</td>
</tr>
<tr>
<td>( Z_i(t) )</td>
<td>Data size associated with the computation of terminal ( i )</td>
</tr>
<tr>
<td>( T_{i}^{max} )</td>
<td>Completion time deadline of terminal ( i )</td>
</tr>
<tr>
<td>( F_{i1}^{l} )</td>
<td>Number of computing resources of terminal ( i )</td>
</tr>
<tr>
<td>( F_{i0} )</td>
<td>Number of computing resources of edge node ( m )</td>
</tr>
<tr>
<td>( \kappa )</td>
<td>Effective energy consumption coefficient</td>
</tr>
<tr>
<td>( e_{i}^{l}(t) )</td>
<td>Energy consumption for local computing of terminal ( i ) in slot ( t )</td>
</tr>
<tr>
<td>( e_{i}^{t}(t) )</td>
<td>Energy consumption for transmission by terminal ( i ) in slot ( t )</td>
</tr>
<tr>
<td>( e_{i}^{e}(t) )</td>
<td>Entire energy consumption of terminal ( i ) in time slot ( t )</td>
</tr>
<tr>
<td>( \alpha_{i}(t) )</td>
<td>Battery energy of terminal ( i ) in time slot ( t )</td>
</tr>
<tr>
<td>( C_{\alpha_{i}m}(t) )</td>
<td>Transmission rate of channel from terminal ( i ) to edge node ( \tau_i )</td>
</tr>
<tr>
<td>( \lambda_{\alpha_{i}m}(t) )</td>
<td>Energy usage to transmit a unit data from terminal ( i ) to node ( \tau_i )</td>
</tr>
<tr>
<td>( e_{h_{i}}(t) )</td>
<td>Harvested energy of terminal ( i ) in time slot ( t )</td>
</tr>
<tr>
<td>( e_{h_{max}} )</td>
<td>Maximum energy value that can be harvested at terminal ( i )</td>
</tr>
<tr>
<td>( E_{i}^{b} )</td>
<td>Battery capacity of terminal ( i )</td>
</tr>
<tr>
<td>( V )</td>
<td>Penalty factor of energy consumption</td>
</tr>
<tr>
<td>( T )</td>
<td>Number of time slots</td>
</tr>
<tr>
<td>( \alpha_{i}(t) )</td>
<td>Offloading proportion of terminal ( i ) to node ( \tau_i ) in slot ( t )</td>
</tr>
<tr>
<td>( \beta_{\alpha_{i}m}(t) )</td>
<td>Offloaded proportion from terminal ( i ) to ( \tau_i ) for tasks of terminal ( i )</td>
</tr>
<tr>
<td>( f_{r_{i}m}(t) )</td>
<td>Computing resources allocated at node ( m ) for task of terminal ( i )</td>
</tr>
</tbody>
</table>

![Fig. 1. Collaborative edge nodes with terminals.](image-url)
be transmitted through the uplink channel from terminal $i$ to edge node $\tau_i$. For simplicity, we assume that the transmission rate of the channel is a constant $C_{i\tau_i}(t)$ in time slot $t$. This is justified by the fact that the transmission rate between terminal and network usually does not vary significantly, and it is provided and guaranteed by the internet service provider. Then the transmission time of the offloaded data is
\[
DU_i^{tr}(t) = \alpha_i(t)Z_i(t)C_{i\tau_i}(t).
\]

For simplicity, we do not take account of the time needed to download the result of the computation from an edge node to a terminal. This assumption is justified as follows. First, usually the result of the computation has a small amount of data, so it requires a short transmission time. For example, a large data set required for machine learning is transmitted to edge nodes by uplink, and the derived model parameters, involving much less data, are sent back to the terminal through downlink. Second, the downlink transmission rate is usually much larger than that of the uplink, e.g. in 5G, where download transmission time of the computation result is further reduced. To further simplify the problem investigated, we assume that the energy required to transmit a unit of data from terminal $i$ to edge node $\tau_i$ is $l_{i\tau_i}(t)$; this can be seen as follows. For a given transmission rate, the time to transmit one unit of data can be obtained; then the transmission power multiplied by the transmission time gives the energy required to transmit one unit of data. Note that the transmission power can be estimated from Shannon’s formula, with constant bandwidth and no interference between channels assumed. Such assumptions may be realistic because modern technologies, e.g. 5G, provide sufficiently large communication resources.

The energy consumption of the data transmission is
\[
ed_i^{tr}(t) = l_{i\tau_i}(t)\alpha_i(t)Z_i(t).
\]

The entire energy consumption of the terminal includes the energy consumption of local computing and of the data transmission of offloaded data to the edge node. It is expressed as
\[
ed_i(t) = ed_i(t) + ed_i^{tr}(t)
= \kappa(F_i^t)^2D_i(t)(1 - \alpha_i(t)) + \alpha_i(t)Z_i(t)l_{i\tau_i}(t).
\]

The energy available to a terminal is constrained by battery limitations. The entire energy consumption, then, cannot exceed the remaining battery energy $e_i(t)$. Thus we have
\[
ed_i(t) \leq e_i(t).
\]

We consider energy harvesting in this paper, specifically, the terminals have the abilities to harvest energy, such as solar or wind energy. The energy harvested by terminal $i$ in time slot $t$ is denoted as $eh_i(t)$, and the maximum harvested energy is $eh_i^{max}$. We have
\[
0 \leq eh_i(t) \leq eh_i^{max}.
\]

The battery installed in the terminal has an energy capacity limitation, denoted by $E_i^{max}$, i.e., the battery can store energy up to $E_i^{max}$. The energy harvested in the current time slot is stored in the battery to be used in later time slots. Consequently, the battery energy update of terminal $i$ is as follows
\[
ed_i(t+1) = \min\{e_i(t) + eh_i(t) - e_i^{tr}(t), E_i^{max}\}, \quad (7)
\]
where $e_i(t) + eh_i(t) - e_i^{tr}(t)$ denotes the rest energy in a battery after energy harvesting and energy consumption, and is no less than $eh_i(t)$, as is guaranteed by (6).

### B. Edge Computing

Edge nodes can be base stations or WiFi access points with limited computing resources, and these resources are shared among terminals. Efficient allocation of resources to terminals is vital for the performance of the edge computing system. In addition, terminals may generate computation hungry tasks with short delay tolerances, resulting in the computation capacity of a single edge node being insufficient to complete the task before the deadline. In this paper, we allow edge nodes to work cooperatively to process computational tasks, so that distributed computing resources can be efficiently used for the processing of each computational task. In like vein, computational tasks offloaded to edge nodes may be further offloaded to other edge nodes for processing. The variable of further offloading is $\beta_{\tau_m}(t)$, denoting the proportion of the computational task offloaded from node $\tau_i$ to node $m$ where the task originally arises in terminal $i$. If $m$ is equal to $\tau_i$, it signifies the proportion of the task that remains with node $\tau_i$ after the second offloading, so that we have
\[
\sum_{m=1}^M \beta_{\tau_m}(t) = 1. \quad (8)
\]

At edge nodes, we assume computing resources are allocated for each task to guarantee the uninterrupted processing of the computing task. This means that the processing time of the offloaded task at node $m$ is $\frac{\alpha_i(t)D_i(t)\beta_{\tau_m}(t)}{f_{\tau_m}(t)}$, where $f_{\tau_m}(t)$ constitutes the computational resources (in terms of CPU cycles/s) allocated at node $m$ for the task of terminal $i$ that is offloaded from node $\tau_i$. In addition to the processing time, the transmission time from node $\tau_i$ to node $m$ should be included in the delay time of this part of task. The connections among edge nodes are usually wired, e.g. optical fiber, which can provide a guaranteed transmission rate between edge nodes. Then the transmission time of one unit of data between edge nodes is calculated as the division of a unit data length by the transmission rate, so that the delay time is
\[
DF_{\tau_m}(t) = \frac{\alpha_i(t)D_i(t)\beta_{\tau_m}(t)}{f_{\tau_m}(t)} + \alpha_i(t)Z_i(t)\beta_{\tau_m}(t)\Lambda_{\tau_m}(t), \quad (9)
\]
where $\Lambda_{\tau_m}(t)$ represents the transmission time of a unit of data from node $\tau_i$ to node $m$, and if $m = \tau_i$, $\Lambda_{\tau_m}(t) = 0$, signifying no data transmission from the node to itself.

With computational task offloading in a cooperative edge computing network, a computational task may be divided into multiple parts (subtasks) and processed with local computing and edge computing. All subtasks are processed in parallel, and the entire task completion time is decided by the subtask that is finished last:
\[
DT_i(t) = \max_m\{DU_i^t(t), DU_i^{tr}(t), DF_{\tau_m}(t)\}. \quad (10)
\]
In (10), the subtask processed by local computing incurs only computation processing time. The subtask processed with edge computing incurs transmission time from terminal $i$ to node $\tau_i$, transmission time from node $\tau_i$ to node $m$ at the second offloading (if used), and the processing time at node $m$.

C. Problem Formulation

Energy consumption affects computing resource allocation and task offloading, and the battery energy of the terminal can affect the network performance and the end user experience. On the other hand, there is a trend to use green energy: affect the network performance and the end user experience. and task offloading, and the battery energy of the terminal can

\[ \text{min} \quad \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} e_i^\tau(t) \]
\[ \text{s.t.} \quad \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} f_{\tau,m}(t) \leq F^e_m \quad \forall m \in \mathcal{M}, \]
\[ \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} DT_i(t) \leq T_{i,\tau}^{max} \quad \forall i \in \mathcal{N}, \]
\[ \sum_{m=1}^{M} \beta_{\tau,m}(t) = 1 \quad \forall i \in \mathcal{N}, \]
\[ e_i^\tau(t) \leq e_i(t) \quad \forall i \in \mathcal{N}, \]
\[ 0 \leq \alpha_i(t) \leq 1 \quad \forall i \in \mathcal{N}, \]
\[ 0 \leq \beta_{\tau,m}(t) \leq 1 \quad \forall i \in \mathcal{N}, m \in \mathcal{M}, \]
\[ 0 \leq f_{\tau,m}(t) \leq F^e_m \quad \forall i \in \mathcal{N}, m \in \mathcal{M}, \]

where the expressions of $e_i^\tau(t)$ and $DT_i(t)$ are given by (5) and (10), respectively. In Problem 1, the decision variables are $\alpha_i(t), \beta_{\tau,m}(t), f_{\tau,m}(t)$ at each time slot, $i \in \mathcal{N}, m \in \mathcal{M}$. The objective is to minimize the long-term average energy consumption of all terminals while accommodating all computational tasks with guaranteed completion deadline $T_{i,\tau}^{max}$ that is shorter than the time slot. The constraint (12) implies that the long-term average computational resource usage at edge node $m$ does not exceed the resource capacity. This long-term average constraint allows the peak value of the total computational resource usage in some time slots to exceed capacity as long as long-term average resource usage does not exceed capacity. The constraint (13) implies that the long-term average task completion time does not exceed the deadline, which determines user experience at a terminal. In a similar fashion to constraint (12), the task completion time is allowed to be larger than the deadline in some time slots provided the long-term average completion time is no larger than the deadline. In this paper, we formulate our optimization problem based on long-term energy consumption, and computational resources and delay. This is done for tractability and it is also consistent with existing service level agreements in industry [53]. The task completion time $DT_i(t)$ is expressed as the maximum in (10), and constraint (13) can be replaced by two simple constraints as follows

\[ \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \frac{(1 - \alpha_i(t))D_i(t)}{F^e_i} \leq T_{i,\tau}^{max} \quad \forall i \in \mathcal{N}, \]
\[ \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \alpha_i(t)Z_i(t) + \frac{\alpha_i(t)D_i(t)\beta_{\tau,m}(t)}{f_{\tau,m}(t)} \leq T_{i,\tau}^{max} \quad \forall i \in \mathcal{N}, m \in \mathcal{M}. \]

Constraint (14) indicates that the second offloading should ensure that the task is processed completely. It is noted that dummy solutions in which $\alpha_i(t)$ is zero for some $i$, $t$, but the corresponding $\beta_{\tau,m}(t)$ have non-zero values may be generated in some cases, and we can easily set these $\beta_{\tau,m}(t)$ to zero in those cases. Constraint (15) states that the energy consumption used for local computing and data transmission in the slot $t$ cannot exceed the current battery energy of the terminal; recall that the energy harvested in a time slot can only be used in later time slots. The constraint (16) indicates that the offloading proportion $\alpha_i$ can be zero, which means that the entire task of terminal $i$ is processed locally. If it is 1, it means that the entire task is processed at the edge node. Otherwise, the task is divided and processed with local computing and edge computing in parallel. The constraint (17) describes the range of the computational task proportions for edge nodes at the second offloading. Accordingly, in (18), $f_{\tau,m}(t)$ represents the number of computational resources at edge node $m$ to process the task of terminal $i$ offloaded from node $\tau_i$, and this allocation cannot exceed the resource capacity of node $m$. In Problem 1, if $T$ has fixed finite values, the problem becomes an optimization problem that considers exactly $T$ time slots. Then the resource allocation and computation offloading for all $T$ time slots have to be jointly optimized to minimize the overall average energy consumption. This is an intractable non-convex optimization problem because the delay term in the constraint (13) is a non-convex function, even for a single time slot, and decisions have to be optimized jointly over all $T$ time slots. For tractability, we therefore assume that the parameter $T$ approaches infinity in this paper.

IV. LYAPUNOV OPTIMIZATION BASED PROBLEM UPPER BOUND ANALYSIS

In this section, to solve the original Problem 1 with long-term average objective function and long-term average constraints, we use the Lyapunov optimization via the construction of virtual queue to convert the original problem into a new optimization problem, and then we will analyze the performance of this method.

A. Virtual Queue

The Lyapunov optimization can be used to solve the long-term average problem by replacing long-term average constraints by virtual queues, so that the original problem becomes
a drift-plus-penalty problem [32]. In the original Problem P1, there are three long-term average constraints, so that three virtual queues need to be constructed.

For Constraint (12), we introduce a virtual queue $Q_m(t)$ to replace the computational resource constraint of terminal $m$, so that the value of $Q_m(t)$ indicates how much computational resource in excess of capacity is allocated in time slot $t$, where $\forall m \in \mathcal{M}$ as follows

$$Q_m(t + 1) = \max \left\{ Q_m(t) + \sum_{i=1}^{N} f_{\tau,m}(t) - F_{\tau,m}^e, 0 \right\}. \quad (21)$$

For Constraint (19), a virtual queue $U_i(t)$ replaces the local computing time constraint, $\forall i \in \mathcal{N}$, as follows

$$U_i(t + 1) = \max \left\{ U_i(t) + \left(1 - \alpha_i(t)\right) D_i(t) \right\} - T_i^{\max}, 0 \right\}. \quad (22)$$

For Constraint (20), a virtual queue $R_{\tau,m}(t)$ replaces the edge computing time constraint, $\forall i \in \mathcal{N}, m \in \mathcal{M}$, as follows

$$R_{\tau,m}(t + 1) = \max \left\{ R_{\tau,m}(t) + \frac{\alpha_i(t) Z_i(t)}{C_{\tau,m}(t)} + \frac{\alpha_i(t) D_i(t) \beta_{\tau,m}(t)}{f_{\tau,m}(t)} + \alpha_i(t) Z_i(t) \beta_{\tau,m}(t) \Delta_{\tau,m}(t) - T_i^{\max}, 0 \right\}. \quad (23)$$

It can be seen from the virtual queue $Q_m(t)$ that if the total amount of allocated computational resources exceeds the resource capacity, the virtual queue length $Q_m(t)$ will increase, otherwise, the virtual queue length will decrease. The other virtual queues work similarly.

**Lemma 1:** If all three virtual queues are rate stable, i.e. $\lim_{T \to \infty} \frac{Q_m(T)}{T} = 0$, $\lim_{T \to \infty} \frac{U_i(T)}{T} = 0$, and $\lim_{T \to \infty} \frac{R_{\tau,m}(T)}{T} = 0$, $\forall i \in \mathcal{N}, m \in \mathcal{M}$, the long-term average constraints (12), (13) are satisfied.

**Proof:** See Appendix A.

**Remark 1:** Lemma 1 established that if the following optimization problem P2 with queue stability constraints is feasible, the optimal objective function value and the optimal solution will be the same as those of the original Problem P1 with long-term average constraints.

$$\text{P2:} \quad \min \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} e_i^x(t), \quad (24)$$

$s.t.$ $Q_m(t)$ is rate stable, $\forall m \in \mathcal{M}$, $U_i(t)$ is rate stable, $\forall i \in \mathcal{N}$, $R_{\tau,m}(t)$ is rate stable, $\forall i \in \mathcal{N}, m \in \mathcal{M}$, Equations (14)-(18). \quad (25) (26) (27) (28)

### B. Drift-plus-Penalty

In this subsection, we will further convert Problem P2 into an upper bound problem with drift-plus-penalty [32] according to Lyapunov optimization. First, we define vector $\Theta(t)$ that contains all the virtual queues, with the Lyapunov function written as follows

$$L(\Theta(t)) = \frac{1}{2} \left\{ \sum_{m=1}^{M} Q_m(t)^2 + \sum_{i=1}^{N} U_i(t)^2 + \sum_{i=1}^{M} \sum_{m=1}^{N} R_{\tau,m}(t)^2 \right\}. \quad (29)$$

The drift $\Delta \Theta(t)$ is defined by

$$\Delta \Theta(t) = \mathbb{E}\left\{ L(\Theta(t + 1)) - L(\Theta(t)) \big| \Theta(t) \right\}. \quad (30)$$

Via the drift-plus-penalty formalism, Problem P2 with queue stability constraints is converted into the upper bound Problem P3 as follows, where the queue stability constraints are replaced by the drift term in the objective function and the original objective is penalized.

$$\text{P3:} \quad \min \Delta \Theta(t) + V \mathbb{E}\left\{ \sum_{i=1}^{N} e_i^x(t) \right\}, \quad (31)$$

$s.t.$ Equations (14)-(18),

where $V$ is the penalty parameter for energy consumption in the objective function of Problem P3. The combination of energy consumption and drift is minimized, and if this problem is feasible (with feasible solutions and the drift is less than $+\infty$), then the three virtual queues are rate stable (proved later). Next, we will drive the upper bound of the drift, which is composed of drift parts from all virtual queues. The drift part of virtual queue $Q_m(t)$ is

$$Q_m(t + 1)^2 - Q_m(t)^2 = \max \left\{ Q_m(t) + \sum_{i=1}^{N} f_{\tau,m}(t) - F_{\tau,m}^e, 0 \right\}^2 - Q_m(t)^2 \leq \left[ Q_m(t) + \sum_{i=1}^{N} f_{\tau,m}(t) - F_{\tau,m}^e \right]^2 - Q_m(t)^2 \leq 2Q_m(t) \left[ \sum_{i=1}^{N} f_{\tau,m}(t) - F_{\tau,m}^e \right] + \left[ \sum_{i=1}^{N} f_{\tau,m}(t) - F_{\tau,m}^e \right]^2 \leq 2Q_m(t) \sum_{i=1}^{N} f_{\tau,m}(t) - 2Q_m(t) F_{\tau,m}^e + \left( \sum_{i=1}^{N} f_{\tau,m}(t) \right)^2 + (F_{\tau,m}^e)^2 - 2F_{\tau,m}^e \sum_{i=1}^{N} f_{\tau,m}(t) \leq 2(Q_m(t) - F_{\tau,m}^e) \sum_{i=1}^{N} f_{\tau,m}(t) - 2Q_m(t) F_{\tau,m}^e + (N^2 + 1)(F_{\tau,m}^e)^2. \quad (32)$$

This is the upper bound of the drift part of the virtual queue $Q_m(t)$. The first inequality is because $\max\{a, 0\}^2 \leq a^2$. The last inequality arises from $0 \leq f_{\tau,m}(t) \leq F_{\tau,m}^e$. Accordingly, we also have

$$\sum_{m=1}^{M} \left[ Q_m(t + 1)^2 - Q_m(t)^2 \right]$$
\[
\begin{align*}
&\leq \sum_{m=1}^{M} \left[ 2(Q_m(t) - F_m) \sum_{i=1}^{N} f_{\tau,m}(t) - 2Q_m(t)F_m \right. \\
&\quad + (N^2 + 1)(F_m^e)^2 \left. \right] \\
&= G_1 + 2 \sum_{i=1}^{N} \sum_{m=1}^{M} (Q_m(t) - F_m) f_{\tau,m}(t),
\end{align*}
\]

where \(G_1\) is a constant in a fixed time slot which is set by
\[
G_1 = \sum_{m=1}^{M} \left[ (N^2 + 1)(F_m^e)^2 - 2Q_m(t)F_m \right].
\]

Similar derivations can be applied to the other virtual queues. The drift part of virtual queue \(U_i(t)\) is
\[
\begin{align*}
U_i(t + 1)^2 - U_i(t)^2 &= \max \left\{ U_i(t) + \left( \frac{1 - \alpha_i(t)D_i(t)}{F_i} - T_i^{\max} \right), 0 \right\}^2 - U_i(t)^2 \\
&\leq \left[ U_i(t) + \left( \frac{1 - \alpha_i(t)D_i(t)}{F_i} - T_i^{\max} \right) \right]^2 - U_i(t)^2 \\
&= \left( \frac{1 - \alpha_i(t)D_i(t)}{F_i} \right)^2 + (T_i^{\max})^2 - \frac{2D_i(t)(U_i(t) - T_i^{\max})}{F_i} (1 - \alpha_i(t)) - 2U_i(t)T_i^{\max} \\
&\quad + \frac{2D_i(t)(T_i^{\max} - U_i(t))}{F_i} \alpha_i(t) - 2U_i(t)T_i^{\max},
\end{align*}
\]

and we have
\[
\begin{align*}
&\sum_{i=1}^{N} \left[ U_i(t + 1)^2 - U_i(t)^2 \right] \\
&\leq \sum_{i=1}^{N} \left\{ \left( \frac{D_i(t)}{F_i} \right)^2 + (T_i^{\max})^2 + \frac{2D_i(t)(U_i(t) - T_i^{\max})}{F_i} \right. \\
&\quad + \frac{2D_i(t)(T_i^{\max} - U_i(t))}{F_i} \alpha_i(t) - 2U_i(t)T_i^{\max} \left. \right\} \\
&= G_2 + \sum_{i=1}^{N} \frac{2D_i(t)(T_i^{\max} - U_i(t))}{F_i} \alpha_i(t),
\end{align*}
\]

where \(G_2\) is a constant in a fixed time slot, set to be
\[
G_2 = \sum_{i=1}^{N} \left\{ \left( \frac{D_i(t)}{F_i} \right)^2 + (T_i^{\max})^2 \\
+ \frac{2D_i(t)(U_i(t) - T_i^{\max})}{F_i} - 2U_i(t)T_i^{\max} \right\}.
\]

Similarly, the drift part of virtual queue \(R_{\tau,m}(t)\) is
\[
\begin{align*}
R_{\tau,m}(t + 1)^2 - R_{\tau,m}(t)^2 \\
&\leq \left[ \frac{\alpha_i(t)Z_i(t)}{C_{\tau_i}(t)} + \frac{\alpha_i(t)D_i(t)\beta_{\tau,m}(t)}{f_{\tau,m}(t)} \right. \\
&\quad + \frac{\alpha_i(t)Z_i(t)\beta_{\tau,m}(t)\Lambda_{\tau,m}(t) - T_i^{\max}}{C_{\tau_i}(t)} \left. \right] ^2 \\
&+ 2R_{\tau,m}(t) \left[ \frac{\alpha_i(t)Z_i(t)}{C_{\tau_i}(t)} + \frac{\alpha_i(t)D_i(t)\beta_{\tau,m}(t)}{f_{\tau,m}(t)} \right. \\
&\quad + \frac{\alpha_i(t)Z_i(t)\beta_{\tau,m}(t)\Lambda_{\tau,m}(t) - T_i^{\max}}{C_{\tau_i}(t)} \left. \right] \\
&= \left[ \frac{\alpha_i(t)Z_i(t)}{C_{\tau_i}(t)} + \frac{\alpha_i(t)D_i(t)\beta_{\tau,m}(t)}{f_{\tau,m}(t)} \right. \\
&\quad + \frac{\alpha_i(t)Z_i(t)\beta_{\tau,m}(t)\Lambda_{\tau,m}(t) - T_i^{\max}}{C_{\tau_i}(t)} \left. \right] + \left( T_i^{\max} \right)^2 \\
&\quad + 2R_{\tau,m}(t)T_i^{\max} + 2(R_{\tau,m}(t) - T_i^{\max}) \frac{\alpha_i(t)Z_i(t)}{C_{\tau_i}(t)} + \frac{\alpha_i(t)D_i(t)\beta_{\tau,m}(t)}{f_{\tau,m}(t)} + \frac{\alpha_i(t)Z_i(t)\beta_{\tau,m}(t)\Lambda_{\tau,m}(t)}{C_{\tau_i}(t)} \\
&\leq \left[ \frac{Z_i(t)}{C_{\tau_i}(t)} + \frac{Z_i(t)\Lambda_{\tau,m}(t)}{f_{\tau,m}(t)} \right. \\
&\quad + \frac{D_i(t)}{f_{\tau,m}(t)} \left. \right] ^2 + \left( T_i^{\max} \right)^2 \\
&\quad + \left[ \frac{Z_i(t)}{C_{\tau_i}(t)} + \frac{Z_i(t)\Lambda_{\tau,m}(t)}{f_{\tau,m}(t)} \right. \\
&\quad + \frac{D_i(t)}{f_{\tau,m}(t)} \left. \right] + \frac{D_i(t)^2}{f_{\tau,m}(t)^2} + 2(R_{\tau,m}(t) - T_i^{\max}) \frac{\alpha_i(t)Z_i(t)}{C_{\tau_i}(t)} + \frac{\alpha_i(t)D_i(t)\beta_{\tau,m}(t)}{f_{\tau,m}(t)} + \frac{\alpha_i(t)Z_i(t)\beta_{\tau,m}(t)\Lambda_{\tau,m}(t)}{C_{\tau_i}(t)} \\
&\quad + \frac{\alpha_i(t)D_i(t)\beta_{\tau,m}(t)}{f_{\tau,m}(t)} + \frac{\alpha_i(t)Z_i(t)\beta_{\tau,m}(t)\Lambda_{\tau,m}(t)}{C_{\tau_i}(t)} + \frac{D_i(t)^2}{f_{\tau,m}(t)^2} + 2(R_{\tau,m}(t) - T_i^{\max}) \frac{\alpha_i(t)Z_i(t)}{C_{\tau_i}(t)} + \frac{\alpha_i(t)D_i(t)\beta_{\tau,m}(t)}{f_{\tau,m}(t)} + \frac{\alpha_i(t)Z_i(t)\beta_{\tau,m}(t)\Lambda_{\tau,m}(t)}{C_{\tau_i}(t)} \right],
\end{align*}
\]

so that
\[
\begin{align*}
\sum_{i=1}^{N} \sum_{m=1}^{M} \left[ R_{\tau,m}(t + 1)^2 - R_{\tau,m}(t)^2 \right] \\
&\leq \sum_{i=1}^{N} \sum_{m=1}^{M} \left\{ \left[ \frac{Z_i(t)}{C_{\tau_i}(t)} + \frac{Z_i(t)\Lambda_{\tau,m}(t)}{f_{\tau,m}(t)} \right. \\
&\quad + \frac{D_i(t)}{f_{\tau,m}(t)} \left. \right] ^2 + \left( T_i^{\max} \right)^2 \\
&\quad + \left[ \frac{Z_i(t)}{C_{\tau_i}(t)} + \frac{Z_i(t)\Lambda_{\tau,m}(t)}{f_{\tau,m}(t)} \right. \\
&\quad + \frac{D_i(t)}{f_{\tau,m}(t)} \left. \right] + \frac{D_i(t)^2}{f_{\tau,m}(t)^2} + 2(R_{\tau,m}(t) - T_i^{\max}) \frac{\alpha_i(t)Z_i(t)}{C_{\tau_i}(t)} + \frac{\alpha_i(t)D_i(t)\beta_{\tau,m}(t)}{f_{\tau,m}(t)} + \frac{\alpha_i(t)Z_i(t)\beta_{\tau,m}(t)\Lambda_{\tau,m}(t)}{C_{\tau_i}(t)} \right] \\
&\quad + \frac{\alpha_i(t)D_i(t)\beta_{\tau,m}(t)}{f_{\tau,m}(t)} + \frac{\alpha_i(t)Z_i(t)\beta_{\tau,m}(t)\Lambda_{\tau,m}(t)}{C_{\tau_i}(t)} + \frac{D_i(t)^2}{f_{\tau,m}(t)^2} + 2(R_{\tau,m}(t) - T_i^{\max}) \frac{\alpha_i(t)Z_i(t)}{C_{\tau_i}(t)} + \frac{\alpha_i(t)D_i(t)\beta_{\tau,m}(t)}{f_{\tau,m}(t)} + \frac{\alpha_i(t)Z_i(t)\beta_{\tau,m}(t)\Lambda_{\tau,m}(t)}{C_{\tau_i}(t)} \right],
\end{align*}
\]

where \(G_3\) is a constant that is set as
\[
G_3 = \sum_{i=1}^{N} \sum_{m=1}^{M} \left\{ \left[ \frac{Z_i(t)}{C_{\tau_i}(t)} + \frac{Z_i(t)\Lambda_{\tau,m}(t)}{f_{\tau,m}(t)} \right. \\
+ \left. \right] ^2 + \left( T_i^{\max} \right)^2 - 2R_{\tau,m}(t)T_i^{\max} \right\}.
\]
Finally, the upper bound of the objective function of Problem P3 is expressed as

$$\Delta \Theta(t) + V \mathbb{E} \left\{ \sum_{i=1}^{N} e_i^2(t) | \Theta(t) \right\}$$

$$= \mathbb{E} \left\{ L(\Theta(t+1)) - L(\Theta(t)) | \Theta(t) \right\} + V \mathbb{E} \left\{ \sum_{i=1}^{N} e_i^2(t) | \Theta(t) \right\}$$

$$\leq \frac{1}{2} (G_1 + G_2 + G_3) + V \sum_{i=1}^{M} \left\{ \sum_{m=1}^{N} (Q_m(t) - F_m^c) f_{\tau,m}(t) \right\}$$

$$+ \mathbb{E} \left\{ \sum_{i=1}^{N} D_i(t) \left( T_i^{max} - U_i(t) \right) \alpha_i(t) \right\}$$

$$+ \mathbb{E} \left\{ \sum_{i=1}^{N} \sum_{m=1}^{M} \left\{ \frac{Z_i(t)}{C_{\tau,m}(t)} + Z_i(t) \Lambda_{\tau,m}(t) \right\} \frac{D_i(t)}{f_{\tau,m}(t)} \right\}$$

$$+ \frac{D_i(t)^2}{2 f_{\tau,m}(t)^2} \right\} + \mathbb{E} \left\{ \sum_{i=1}^{N} \left\{ V[Z_i(t) l_{\tau,i}(t) - \kappa(F_i^f)^2 D_i(t) \right\} \right\}$$

$$= \frac{1}{2} (G_1 + G_2 + G_3) + V \sum_{i=1}^{N} \kappa(F_i^f)^2 D_i(t)$$

$$+ \mathbb{E} \left\{ \sum_{i=1}^{N} \left\{ V[Z_i(t) l_{\tau,i}(t) - \kappa(F_i^f)^2 D_i(t) \right\} \right\}$$

$$+ \frac{D_i(t)(T_i^{max} - U_i(t))}{F_i^f}$$

$$+ \sum_{m=1}^{M} \left\{ \frac{R_{\tau,m}(t) - T_{\tau}^{max}}{C_{\tau,m}(t)} \right\} \alpha_i(t)$$

$$+ \sum_{i=1}^{N} \sum_{m=1}^{M} \left\{ \frac{D_i(t)^2}{2 f_{\tau,m}(t)^2} + (Q_m(t) - F_m^c) f_{\tau,m}(t) \right\}$$

$$+ \left\{ \frac{Z_i(t)}{C_{\tau,m}(t)} + Z_i(t) \Lambda_{\tau,m}(t) \right\} \frac{D_i(t)}{f_{\tau,m}(t)}$$

$$+ \left\{ \frac{Z_i(t)}{C_{\tau,m}(t)} + Z_i(t) \Lambda_{\tau,m}(t) \right\} \frac{D_i(t)}{f_{\tau,m}(t)}$$

$$- T_{\tau}^{max} D_i(t) \alpha_i(t) \beta_{\tau,m}(t)$$

$$- T_{\tau}^{max} Z_i(t) \Lambda_{\tau,m}(t) \alpha_i(t) \beta_{\tau,m}(t) \right\}$$

subject to Equations (14)-(18).

(38)

In this way, the original Problem P1 can be approximated by Problem P4. We will show the relationship between the solutions of the original Problem P1 and the approximate Problem P4.

**Lemma 2:** If Problem P4 is feasible and the optimal objective function value plus the constant $\frac{1}{2} (G_1 + G_2 + G_3) + V \sum_{i=1}^{N} \kappa(F_i^f)^2 D_i(t)$ gives a value of $W$ ($W < \infty$), then the virtual queues must be rate stable for the optimal values of the decision variables, i.e. $\lim_{T \to \infty} \frac{Q_m(t)}{T} = 0$, $\lim_{T \to \infty} \frac{U_i(t)}{T} = 0$, and $\lim_{T \to \infty} \frac{R_{\tau,m}(t)}{T} = 0$, $\forall i \in N$, $m \in M$.  

**Proof:** See Appendix B.

**Lemma 3:** If Problem P4 is feasible, the long-term average constraints (12), (19), (20) can be satisfied.

**Proof:** According to **Lemma 2**, if Problem P4 is feasible, all the virtual queues are rate stable for its optimal solution. From **Lemma 1**, the long-term average constraints (12), (19), (20) are satisfied if the virtual queues are rate stable. This completes the proof.  

**C. Approximation Analysis**

In this section, we describe the mathematical relationship between the solution of the original Problem P1 and the solution of the upper bound Problem P4.

**Theorem 1:** If the original Problem P1 has an optimal value $\phi^*$, the optimal long-term energy consumption value of Problem P4 is constrained as follows

$$\lim_{T \to \infty} \frac{1}{T} \sum_{i=0}^{T-1} \sum_{t=1}^{N} e_i^2(t) \leq \phi^* + \frac{\varepsilon}{T}$$

(39)
where \( e^c_i(t) \) is the energy consumption of terminal \( i \) in time slot \( t \) obtained by Problem P4, and

\[
\begin{align*}
\varepsilon &= \frac{1}{2} \sum_{m=1}^{M} (N^2 + 1)(F^e_m)^2 + \frac{1}{2} \sum_{i=1}^{N} \left\{ \left( \frac{D_i(t)}{F^i_i} \right)^2 + (T^\text{max}_i)^2 \right\} \\
&+ \frac{2D_i(t)(U^\text{max}_i - T^\text{max}_i)}{F^i_i} \\
&+ \sum_{i=1}^{N} \sum_{m=1}^{M} \left\{ \left( \frac{Z_i(t)}{C_{ir}(t)} + Z_i(t)\Lambda_{\tau,m}(t) \right)^2 + (T^\text{max}_i)^2 \right\} \\
&+ \sum_{i=1}^{N} \left\{ \frac{D_i(t)T^\text{max}_i}{F^i_i} + \sum_{m=1}^{M} \frac{R_{\tau,m}Z_i(t)}{C_{ir}(t)} \right\} \\
&+ \sum_{i=1}^{N} \sum_{m=1}^{M} \left[ Q^\text{max}_m F^e_m + R_{\tau,m}^\text{max} Z_i(t)\Lambda_{\tau,m}(t) \right].
\end{align*}
\]

(40)

**Proof:** See Appendix C.

V. DISTRIBUTED PROBLEM AND ALGORITHM DESIGN

In the above section, the original Problem P1 is converted into the upper bound Problem P4 by introducing virtual queues and the drift-plus-penalty strategy. However, Problem P4 is a centralized problem, and it is intractable in a large problem size with a large number of variables. In this section, we provide an efficient distributed algorithm to solve Problem P4.

A. Distributed Problem

The special characters of Problem P4 lead to a distributed algorithm design in this paper. In the objective function of Problem P4, individual terms are summed over all terminals. Straightforward separation of the constraints for each terminal transforms Problem P4 into a distributed problem for each terminal as follows

**DIP:** min \( \{ V[Z_i(t)|_{\tau_i}(t) - \kappa(F^i_i)^2D_i(t)] \}

\[
\begin{align*}
&+ \frac{D_i(t)(T^\text{max}_i - U_i(t))}{F^i_i} \\
&+ \sum_{m=1}^{M} \left( R_{\tau,m}(t) - T^\text{max}_i \right) Z_i(t) \right\} \alpha_i(t) \\
&+ \sum_{m=1}^{M} \left\{ \frac{D_i(t)^2}{2R_{\tau,m}(t)^2} + (Q_m(t) - F^e_m)f_{\tau,m}(m) \right\} \\
&+ \left[ \frac{Z_i(t)}{C_{ir}(t)} + Z_i(t)\Lambda_{\tau,m}(t) \right] \frac{D_i(t)}{f_{\tau,m}(m)} + \left( R_{\tau,m}(t) \\
&- T^\text{max}_i \right) D_i(t) \frac{\alpha_i(t)\beta_{\tau,m}(t)}{f_{\tau,m}(t)} \\
&+ \left( R_{\tau,m}(t) - T^\text{max}_i \right) Z_i(t)\Lambda_{\tau,m}(t)\alpha_i(t)\beta_{\tau,m}(t) \right\},
\end{align*}
\]

(41)

\[
\begin{align*}
s.t. \quad &\sum_{m=1}^{M} \beta_{\tau,m}(t) = 1, \\
&\left( Z_i(t)|_{\tau_i}(t) - \kappa(F^i_i)^2D_i(t) \right) \alpha_i(t) \\
&\leq e_i(t) - \kappa(F^i_i)^2D_i(t),
\end{align*}
\]

(42)

(43)

Problem DIP is a non-convex optimization problem because the objective function has multiplications and divisions of the decision variables. Only approximate solutions can be obtained by the existing methods. However, in this paper, motivated by the specifics of Problem DIP, an algorithm is proposed to solve this non-convex optimization problem, and the optimal solution is obtained.

B. Algorithm Design

To simplify the expression of Problem DIP, new notations \( A_i, B_m, J_{im}, W_i, X_{im}, Y_{im} \), \( \forall i \in \chi, m \in M \) are introduced to replace the coefficients in the objective function. For a specific time slot, these coefficients are constant. The new notations are defined as follows

\[
\begin{align*}
A_i &= V[Z_i(t)|_{\tau_i}(t) - \kappa(F^i_i)^2D_i(t)] + \frac{D_i(t)(T^\text{max}_i - U_i(t))}{F^i_i} \\
&+ \sum_{m=1}^{M} \left( R_{\tau,m}(t) - T^\text{max}_i \right) Z_i(t) \right\} \frac{C_{ir}(t)}{C_{ir}(t)} \\
B_m &= Q_m(t) - F^e_m, \\
J_{im} &= \left( \frac{C_{ir}(t)}{C_{ir}(t)} + \Lambda_{\tau,m}(t) \right) Z_i(t)D_i(t), \\
W_i &= \frac{D_i(t)^2}{2}, \\
X_{im} &= (R_{\tau,m}(t) - T^\text{max}_i)D_i(t), \\
Y_{im} &= (R_{\tau,m}(t) - T^\text{max}_i)Z_i(t)\Lambda_{\tau,m}(t).
\end{align*}
\]

Then, the objective function of Problem DIP is

\[
\begin{align*}
&\min A_i\alpha_i(t) + \sum_{m=1}^{M} \left\{ B_m f_{\tau,m}(m) + J_{im} f_{\tau,m}(m)^{-1} \\
&+ W_i f_{\tau,m}(m)^{-2} + X_{im}\alpha_i(t)\beta_{\tau,m}(t)f_{\tau,m}(m)^{-1} + Y_{im}\alpha_i(t)\beta_{\tau,m}(t) \right\}.
\end{align*}
\]

From the above formula, we find that the objective function has the special character of linearity in the decision variable \( \alpha_i(t) \), and all constraints involving \( \alpha_i(t) \) (constraints (43) and (44)) are also linear. If an optimization problem is linear in variable \( x \), and the constraints related to \( x \) are also linear, the optimal solution of the problem must be obtained at one endpoint of value range of \( x \). Problem DIP is thus converted into two new problems corresponding to the two extreme values of the variable \( \alpha_i(t) \) (two endpoints of the value range of the variable) as follows

\[
\begin{align*}
&\min A_i\alpha^i_i(t) + \sum_{m=1}^{M} \left\{ B_m f_{\tau,m}(m) + J_{im} f_{\tau,m}(m)^{-1} \\
&+ W_i f_{\tau,m}(m)^{-2} + X_{im}\alpha^i_i(t)\beta_{\tau,m}(t)f_{\tau,m}(m)^{-1} + Y_{im}\alpha^i_i(t)\beta_{\tau,m}(t) \right\},
\end{align*}
\]

(47)

where \( \alpha^i_i(t) \) is either the endpoint (\( \alpha^\text{min}_i(t) \) or \( \alpha^\text{max}_i(t) \)) of the value range. These two new problems have eliminated
the decision variable \( \alpha_i(t) \) and only have constraints (42), (45) and (46). The minimal objective value of two new problem solutions has the same value as the optimal solution of Problem DIP. Next we observe that the objective function (47) of the new problems can be rewritten as

\[
\begin{align*}
\min_{m=1}^{M} & \sum_{m=1}^{M} \beta_{\tau,m}(t)(X_{im}\alpha_i^*(t)f_{\tau,m}(t)^{-1} + Y_m\alpha_i^*(t)) \\
& + \sum_{m=1}^{M} (B_mf_{\tau,m}(t) + J_{im}f_{\tau,m}(t))^{-1} + W_i f_{\tau,m}(t)^{-2} + A_i\alpha_i^*(t) \\
& + \sum_{m=1}^{M} \beta_{\tau,m}(t)(X_{im}\alpha_i^*(t)f_{\tau,m}(t)^{-1} + Y_m\alpha_i^*(t)) \\
& + \sum_{m=1}^{M} (B_mf_{\tau,m}(t) + J_{im}f_{\tau,m}(t))^{-1} + W_i f_{\tau,m}(t)^{-2} + A_i\alpha_i^*(t).
\end{align*}
\]

This is also linear in the decision variable \( \beta_{\tau,m}(t) \), and the constraints (42) and (45) associated with \( \beta_{\tau,m}(t) \) are linear. 

**Lemma 4**: The optimization problem with objective function (48) and constraints (42), (45) and (46) has an optimal solution in which only one \( \beta_{\tau,m}(t) \) \( \forall m \in M \) has a non-zero value and since all these variables must be nonnegative and they sum up to 1, this variable must have the value of 1.

*Proof:* Suppose that the optimal solution has more than one \( \beta_{\tau,m}(t) \) variables with positive values. Without loss of generality, we assume that there are two variables \( \beta_{\tau,m1}(t) \) and \( \beta_{\tau,m2}(t) \) with positive values in the optimal solution, and all other \( \beta_{\tau,m}(t) \) \( \forall m \in M \) are zero. Then we have \( \beta_{\tau,m1}(t) + \beta_{\tau,m2}(t) = 1 \) according to constraints (42) and (45). The coefficients \( (X_{im}\alpha_i^*(t)f_{\tau,m}(t)^{-1} + Y_m\alpha_i^*(t)) \) of \( \beta_{\tau,m1}(t) \) and \( \beta_{\tau,m2}(t) \) are different. Accordingly, the optimal value can be decreased by increasing one variable value (with smaller coefficient value) and decreasing the other variable value until they reach the value 1 and 0 respectively. This contradicts the assumption and completes the proof. 

From Lemma 4, we know that only one \( \beta_{\tau,m}(t) \) has the value 1 and all others are 0, but we do not know exactly which one has the value 1. In addition, if \( \beta_{\tau,m}(t) = 0 \), \( f_{\tau,m}(t) \) must have the value 0, then we can convert the optimization problem with objective function (48) and constraints (42), (45) and (46) into \( M \) subproblems, each for an specific \( \beta_{\tau,m}(t) \), and the minimal result of all subproblems is the optimum of the original optimization problem. It is noted that \( M \) subproblems can be further distributively executed because these subproblems are independent apart from the selection of the minimum after the subproblem calculation. The objective function of the subproblem is

\[
\begin{align*}
\min & \quad X_{im}\alpha_i^*(t)f_{\tau,m}(t)^{-1} + Y_m\alpha_i^*(t) \\
& + B_mf_{\tau,m}(t) + J_{im}f_{\tau,m}(t)^{-1} + W_i f_{\tau,m}(t)^{-2} + A_i\alpha_i^*(t).
\end{align*}
\]

This objective function (49) is a convex function with only one variable \( f_{\tau,m}(t) \), and this optimization problem can be solved by convex problem solvers. It is noted that after the variable settings and formulae transformations described above, optimality is maintained, which guarantees the solutions obtained by the proposed algorithm is the optimal solution of the original Problem DIP. When solving the convex problem that has the objective function (49) and constraint (46), if \( \alpha_i^*(t) \) has the value 0, there is no computation offloading at terminal \( i \) at time slot \( t \). Then, \( \beta_{\tau,m}(t) \) and \( f_{\tau,m}(t) \) must have the value 0, and the computational task is processed locally at the terminal.

**Algorithm 1: Collaborative Offloading Algorithm**

**Input:** Computational resource \( F_i^t \), battery energy \( e_i(t) \) of terminal \( i \), edge node computing capacity \( F_e^m \), virtual queue lengths \( Q_{im}(t), U_i(t), R_{\tau,m}(t) \).

**Output:** Optimal solution \( S = (\alpha_i(t), \beta_{\tau,m}(t), f_{\tau,m}(t)) \), objective function value \( H \).

1. \( S \leftarrow 0, H = \infty, m' = 0, f' = 0 \).
2. for \( m = 1 \) to \( M \) do
3. \( \text{Consider constraints (43) and (44), if there is no feasible range of } \alpha_i(t), \text{Continue, otherwise obtain the left endpoint value } \alpha_l \text{ and the right endpoint value } \alpha_h. \)
4. for \( \alpha_i^*(t) \in \{\alpha_l, \alpha_h\} \) do
5. \( \text{If } \alpha_i^*(t) = 0 \text{ then} \)
6. \( \text{The entire task is processed at the terminal, } h = 0. \)
7. \( \text{If } H > h \)
8. \( H = h, \alpha_i(t) = m', f' = 0. \)
9. \( \text{else} \)
10. \( \text{Solve the convex optimization problem with objective function (49) and constraint (46), obtain objective result } h \text{ and solution } f. \)
11. \( S_{\tau,m} = 1, f_{\tau,m}(t) = f'. \)
12. Return \( S, H \).

The specific algorithm is shown as Algorithm 1. The for loop of lines 4–10, performs two optimizations for each of the two endpoint values of the variable \( \alpha_i(t) \). In the loop, at line 5, no optimization is required because the entire task is processed at the terminal. At line 8, the convex optimization problem is solved by solvers. If a better solution is derived, the final result of the algorithm will be updated, which means the solution with the minimal result will be selected as the solution of Problem DIP. As mentioned above, these subproblems (total number is \( M \)) can be distributively executed if the terminal has a distributive computation ability. At line 11, the final solution is provided as the solution of the subproblem that has the minimal objective function value, where corresponding variables \( \beta_{\tau,m}(t) \) and \( f_{\tau,m}(t) \) are set with the values obtained by the subproblem. It is easy to see that the complexity of the algorithm is polynomial because it devolves to \( 2M \) convex optimization problems.

**VI. NUMERICAL RESULTS**

This section evaluates the performance of the proposed distributed algorithm in different experimental scenarios, where multiple edge nodes connect together and provide edge computing service to terminals. In the scenarios, each terminal executes Algorithm 1 and the values of decision variables are derived, then edge nodes collect \( f_{\tau,m}(t), i \in \mathcal{N} \) from
terminals and update the virtual queue $Q_m(t)$ according to (21). Finally, the length of the virtual queue $Q_m(t)$ is returned to terminals, incurring a low communication overhead. The other two virtual queues $U_i(t)$, and $R_{\tau,m}(t)$, and the battery energy $e_i(t)$ are updated at terminals according to (22), (23), and (7), respectively. In the experimental scenarios, terminals are randomly scattered around the coverage area of each edge node. As our algorithm assumes that $T$ approaches infinity, while the simulations must terminate at finite time, we terminate the simulations after a sufficiently long time to achieve stability of the average energy consumption. In particular, in the simulations that were done to produce the results presented in this section, 100 times slot were used for the length of the simulations. The parameter settings in the simulations are listed in Table II.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>10 ~ 50</td>
</tr>
<tr>
<td>$M$</td>
<td>5</td>
</tr>
<tr>
<td>$Z_i(t)$</td>
<td>$\text{unif}[100, 200]$ Kbit</td>
</tr>
<tr>
<td>$D_i(t)$</td>
<td>$\text{unif}[0.5, 3]$ Giga CPU cycles</td>
</tr>
<tr>
<td>$T_i^{\text{max}}$</td>
<td>$\text{unif}[50, 400]$ ms</td>
</tr>
<tr>
<td>$F_i^L$</td>
<td>$\text{unif}[2, 3]$ GHz</td>
</tr>
<tr>
<td>$F_i^H$</td>
<td>$\text{unif}[5, 15]$ GHz</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>$1 \times 10^{-28}$</td>
</tr>
<tr>
<td>$e^{\text{max}}_i$</td>
<td>$\text{unif}[1.6, 2.4]$ J</td>
</tr>
<tr>
<td>$E_i^{\text{max}}$</td>
<td>$\text{unif}[4, 6]$ J</td>
</tr>
<tr>
<td>$C_i(t)$</td>
<td>$\text{unif}[0.5, 2]$ Mbit/s</td>
</tr>
<tr>
<td>$I_{\tau,m}(t)$</td>
<td>$\text{unif}[0.05, 0.2]$ s/Mbit</td>
</tr>
<tr>
<td>$V$</td>
<td>10 ~ 1800</td>
</tr>
</tbody>
</table>

### A. Comparison to Non-Collaborative Algorithm

In this subsection, we show the benefit of the collaborative approach in the proposed distributed algorithm over the computational offloading with non-collaborative edge computing (an edge node does not help other edge nodes with computing services). The non-collaborative algorithm is a simply modification of the proposed distributed algorithm, achieved by making $\beta_{\tau,m}(t) = 0$, $f_{\tau,m}(t) = 0$, $\forall m \neq \tau$, to prevent collaborative computing between edge nodes.

Fig. 2 compares the performances of two algorithms with different penalty factor values $V$, where the numbers of terminals and the number of edge nodes are fixed at $N = 20$ and $M = 5$, respectively. The value of $V$ signifies the impact of the energy consumption on the objective function value. In Fig. 2(a), the proposed collaborative algorithm always has a lower energy consumption than the non-collaborative algorithm with different penalty values ($V = 10, 100, 300, 600, 900, 1200, 1500, 1800$), and in Fig. 2(b), the collaborative algorithm has a higher average offloading proportion than the non-collaborative algorithm. This is because the collaborative algorithm uses the edge computational resources more efficiently and accommodates more offloaded tasks than the non-collaborative algorithm.

Fig. 3 demonstrates the performances of the two algorithms with varying numbers of terminals $N$ in the network, where the number of edge nodes and the penalty value are fixed at $M = 5$ and $V = 300$, respectively. In Fig. 3(a), the average energy consumptions per terminal are compared; the proposed collaborative algorithm always has a lower energy consumption than the non-collaborative algorithm irrespective of the number of terminals. When the number of terminals increases, the average energy consumption increases. This is because more terminals compete for the limited edge computing resources, and more computational tasks are forced to be processed locally, increasing energy consumption at terminals. Fig. 3(b) shows the average offloading proportion per terminal; the proposed collaborative algorithm always has a higher offloading proportion than the non-collaborative algorithm. As the number of terminals increases, the average offloading proportion per terminal decreases, again because more terminals compete for the limited edge computing resources, and more computational tasks are forced to be processed locally. The above comparisons show that the collaborative algorithm uses the edge computational resources more efficiently than the non-collaborative algorithm. It can thus accommodate more offloaded tasks and reduce the energy consumptions at terminals.
B. Collaborative Algorithm Performance

In this subsection, the detailed performance of the proposed collaborative algorithm is investigated in different scenarios. In Fig. 4(a)-(c), the average energy consumption per terminal, average drift per terminal and average task offloading proportion per terminal are investigated with different penalty values $V = 300, 600, 900$, with the numbers of terminals and edge nodes fixed at $N = 20$ and $M = 5$, respectively. Fig. 4(a) shows the average energy consumption of the collaborative algorithm. The average energy consumptions become stable after about 20 time slots, and the energy consumption values are different with different penalty values. Specifically, the energy consumption with $V = 300$ has the highest value, followed by $V = 600$, and $V = 900$. This is because the larger the penalty value, the higher weight of the energy consumption in the optimization function (31), which drives the optimization towards a lower energy consumption solution.

Fig. 4(b) shows the average drift per terminal with different penalty values, where the drift value at a time slot for all terminals is calculated according to (30). The drift values are stable and limited, which guarantees satisfaction of the long-term constraints in the optimization problem. In the figure, the drift values become stable after about 20 time slots; the average drift with $V = 900$ has the highest value, followed by $V = 600$, and $V = 300$ (average values are 0.715, 0.62 and 0.426, respectively). The reason is the same as that given for Fig. 4(a), namely that the larger the penalty value, the higher the weight of the energy consumption term, leading the optimization to offload more computational tasks to reduce energy consumption at terminals.

In Fig. 5(a)-(c), the same metrics for the collaborative algorithm are investigated with varying numbers of terminals $N = 10, 30, 50$, and with the number of edge nodes and the penalty value fixed as $M = 5$ and $V = 300$, respectively. In Fig. 5(a), the average energy consumptions are compared for different numbers of terminals; the average energy con-
consumption with $N = 10$ has the lowest value, followed by $N = 30$, and $N = 50$. This is because as the number of terminals increases, more computational tasks compete for the limited computational resources in the fixed set of edge nodes. This leads more tasks to be processed locally and energy consumption to increase at terminals. Fig. 5(b) shows the average drifts with different numbers of terminals. The drift values are stable and limited, with drift values ordered as energy consumptions in Fig. 5(a). This is because more terminals compete for limited computational resources. This in turn imposes pressures on the computational resource and delay constraints, which makes the drift of virtual queues longer. Fig. 5(c) shows the average offloading proportions with different numbers of terminals. The average offloading proportion with $N = 10$ has the highest value, followed by $N = 30$, and $N = 50$. This is for the same reason as we gave for Fig. 5(a): more terminals lead to more computational tasks being processed locally, and the offloading proportion reduces.

Fig. 6. Average energy consumption per terminal with different numbers of edge nodes.

To further demonstrate the performance of the collaborative algorithm when the number of edge nodes varies, in Fig. 6, we show the average energy consumption per terminal when different values for the numbers of edge nodes are considered. In these simulations, the number of terminals and the penalty value are fixed as $N = 20$ and $V = 300$, respectively. As observed from Fig. 6, when the number of edge nodes increases, the energy consumption is reduced. The reason for this is that more edge nodes provide more computational resources that can be used by terminals. Then, more computational tasks can be offloaded from terminals to edge nodes with cooperative edge computing, which reduces the energy consumptions for computation processing at terminals.

VII. CONCLUSION

We have considered the computation offloading problem in collaborative edge computing networks where edge nodes work collaboratively to provide efficient computation services to terminals. The objective of the problem is to minimize the long-term energy consumption of terminals under constraints on long-term computational resource and response time thresholds. We use the drift-plus-penalty Lyapunov optimization approach to convert the original problem into an upper bound problem based on virtual queues to replace long-term constraints. A distributed algorithm is proposed to optimally solve the upper bound problem, and the gap between the proposed distributed algorithm solution and the original optimal solution is analysed. Numerical results have shown that the solution obtained by the algorithm meets the requirements of long-term measures, and keep the long-term average energy consumption of terminals at a low level.

VIII. APPENDIX A

Proof of Lemma 1: From (21), we have $Q_m(t + 1) \geq Q_m(t) + \sum_{i=1}^{N} f_{\tau_i m}(t) - F^e_m$, which yields $Q_m(t + 1) - Q_m(t) \geq \sum_{i=1}^{N} f_{\tau_i m}(t) - F^e_m$. Summing the left and right sides of the
inequality over \( t = 0, 1, 2, ..., T - 1 \), and taking average, we obtain \( \lim_{T \to \infty} \frac{1}{T} (Q_m(T) - Q_m(0)) \geq \lim_{T \to \infty} \frac{1}{T} \sum_{i=0}^{T-1} \sum_{j=1}^{N} f_{r,m}(t) - F'_m \). Combining \( Q_m(0) = 0 \) and rate stability of virtual queue \( Q(t) \) i.e. \( \lim_{T \to \infty} \frac{1}{T} Q_m(T) = 0 \), we then have

\[
\lim_{T \to \infty} \frac{1}{T} \sum_{i=0}^{T-1} \sum_{j=1}^{N} f_{r,m}(t) \leq F'_m.
\]

Similarly, if virtual queues \( U_i \) and \( R_{\tau,m}(t) \) are rate stable, we can obtain satisfactions of long-term constraints (19) and (20). This completes the proof. \[\blacksquare\]

**IX. APPENDIX B**

**Proof of Lemma 2:** From (37), Problem P4 is feasible and, given the objective value \( W \) is obtained, we have

\[
L(\Theta(t + 1)) - L(\Theta(t)) + V \sum_{i=1}^{N} e_i^c(t) \leq W.
\]

From this, we see that \( L(\Theta(t + 1)) - L(\Theta(t)) \leq W - V \sum_{i=1}^{N} e_i^c(t) \leq W \), where

\[
V \sum_{i=1}^{N} e_i^c(t) \geq 0.
\]

Summing up both sides of the inequality over \( t = 0, 1, ..., T - 1 \), we have \( L(\Theta(T)) - L(\Theta(0)) \leq TW \). Because all virtual queues are empty at the beginning, \( L(\Theta(0)) = 0 \). Thus,

\[
L(\Theta(T)) \leq TW.
\]

From (29), \( L(\Theta(T)) = \frac{1}{2} \left\{ \sum_{m=1}^{M} Q_m(T)^2 + \sum_{i=1}^{N} U_i(T)^2 + \sum_{i=1}^{N} \sum_{j=1}^{m} R_{\tau,m}(T)^2 \right\} \), and because the length of any queue is non-negative, for queue \( Q_m(T) \), we have \( 0 \leq Q_m(T) \leq \sqrt{2TW} \) and \( 0 \leq \lim_{T \to \infty} \frac{1}{T} Q_m(T) \leq \lim_{T \to \infty} \frac{1}{T} \sqrt{2TW} = 0 \). Finally, we achieve

\[
\lim_{T \to \infty} \frac{Q_m(T)}{T} = 0.
\]

Similar calculations work for virtual queues \( U_i(t) \) and \( R_{\tau,m}(t) \) yielding the same conclusion, \( \lim_{T \to \infty} \frac{U_i(T)}{T} = 0 \), and \( \lim_{T \to \infty} \frac{R_{\tau,m}(T)}{T} = 0 \). This completes the proof. \[\blacksquare\]

**X. APPENDIX C**

**Proof of Theorem 1:** After solving Problem P4, we obtain the optimal solution with \((\alpha'_i(t), \beta'_{r,m}(t), f'_{r,m}(t), \Delta \Theta(t))\) as the values when variables are assigned the optimal solution. From (37), see that

\[
\Delta \Theta(t) + V \mathbb{E} \left\{ \sum_{i=1}^{N} e_i^c(t) \mid \Theta(t) \right\}
\]

\[
\leq \frac{1}{2} (G_1 + G_2 + G_3) + V \sum_{i=1}^{N} e_i^c(t)
\]

\[
+ \sum_{i=1}^{M} \left\{ D_i(t) \left( T_i^{max} - U_i(t) \right) \right\} \alpha'_i(t) + \sum_{i=1}^{N} \sum_{j=1}^{M} \left\{ \frac{D_i(t)^2}{2 f'_{r,m}(t)^2} \right\}
\]

\[
\leq \frac{1}{2} (G_1 + G_2 + G_3) + \sum_{i=1}^{N} \sum_{j=1}^{M} \left\{ \frac{D_i(t)^2}{2 f'_{r,m}(t)^2} \right\}
\]

\[
(\phi^* - \frac{1}{V} \Delta \Theta(t)) + \sum_{i=1}^{N} \sum_{j=1}^{M} \left\{ \frac{D_i(t)^2}{2 f'_{r,m}(t)^2} \right\} - \frac{1}{V} \Delta \Theta(t).
\]

Summing up both sides of the inequality over \( t =
where $L(\Theta'(0)) = 0$, $f_{\tau,m}(t)$ and $L(\Theta'(T - 1))$ are finite values. This completes the proof.  

REFERENCES


