Task Class Partitioning for Mobile Computation Offloading

Hong Chen*, Terence D. Todd*, Dongmei Zhao* and George Karakostas†
*Department of Electrical and Computer Engineering
†Department of Computing and Software
McMaster University
Hamilton, Ontario, CANADA
Email: {chenh151,todd,dzhao,karakos}@mcmaster.ca

***Abstract*—This paper introduces algorithms for static task class partitioning in mobile computation offloading (MCO). The objective is to partition a given set of task classes into two sets that are either executed locally by the mobile device (MD) or those classes that are permitted to contend for remote edge server (ES) execution. The goal is to find the task class partition that gives the minimum mean MD power consumption subject to task completion deadlines. The paper generates these partitions for both soft and hard task completion deadlines. Two variations of the problem are considered. The first assumes that the wireless and computational capacities are given and the second generates both capacity assignments subject to an additional resource cost budget constraint. The proposed partitioning algorithms are based on heuristic class ordering methods. The paper introduces two class ordering methods, a simpler one based on a task latency criterion, and an hierarchical version that first sorts and groups classes based on a mean power consumption criterion and then orders the task classes within each group based on a task completion time criterion. A variety of simulation results are presented that demonstrate the excellent performance of the proposed solutions for both given and optimized network resource assignments.**

***Index Terms*—Edge computing, mobile computation offloading, task completion deadlines, cost budget constraints, power efficiency.**

I. Introduction

Mobile computation offloading (MCO) is a way of improving mobile device (MD) performance by offloading certain task executions to a more resourceful cloud server [1]. Using MCO, a reduction in MD energy consumption can often be obtained since the energy consumed for task execution is incurred by the remote cloud server. Since wireless communication is used during the offload, the MD will also experience additional wireless energy usage and network latency that would not happen if the tasks were executed locally. To help mitigate this added latency and energy consumption, the remote task execution for MCO is often performed at an edge server (ES) that is located close to the wireless base station (BS) [2], [3].

A key issue in MCO deals with the question of whether a given task should be executed locally or offloaded. This offloading decision is not a trivial one for many reasons [4]. For example, the increased use of MCO may create competition for the limited ES computational capacity that is used for the remote task execution [5], [6]. Offloading decisions are also more problematic when the MD interacts with the ES over wireless transmission channels that may change randomly during the computation offload. For these reasons, it is important to incorporate the temporal evolution of the system into the offloading decision making process [7], [8]. This includes the latencies incurred due to the queueing delays experienced by offloaded tasks as they wait for ES execution.

The acceptable delay for task execution is often tightly constrained for many applications, such as those involving gaming, virtual reality, and object recognition [9]. Task execution deadline constraints significantly increase the difficulty of the problem compared to prior work without completion time requirements or that uses mean delay alone as the performance criterion [10], [11]. The variability in acceptable delay for different classes of tasks further complicates MCO decisions. This is especially true when offloaded tasks contend for wireless channel and computational resources in an uncontrolled fashion, and may impair the advantages of MCO, since it is easy to see that minimizing the used energy and satisfying the task delay constraints are competing objectives, i.e., in an effort to satisfy tight latency constraints, costly energy-wise MCO decisions may be necessary. Thus, the problem of task scheduling so that delay constraints are satisfied and energy consumption is minimized, becomes a difficult problem for MCO.

Our paper studies the use of *task class partitioning* (TCP) to address this problem. Class partitioning can be used in cases where MD tasks belong to one of a given set of task classes that can each be executed locally or offloaded. Each traffic class definition includes the data upload and execution requirements of the class and its execution deadline, i.e., the time by which task execution should be completed. Class partitioning is motivated by the following simple example: An offloaded traffic class with high computational needs and a loose execution delay constraint may unduly occupy the ES, thus preventing more time-constrained tasks from using MCO. In this case, it may be beneficial to pre-assign certain task classes to local execution so that others can more successfully exploit remote offloading. In TCP, the task classes are therefore partitioned into two sets, those task classes to be executed

locally and those that may be offloaded for remote execution. Depending on the adaptability of class definition, TCP can be either *static*, when the task classes are pre-defined and their definition does not change throughout the lifetime of the system, or *dynamic*, when the class definition can adapt to a changing environment. It is clear that the dynamic TCP is a sequence of static TCP instances, each corresponding to a period of stable environment parameters. In this paper, we study the static version of TCP, i.e., STCP.

In STCP, the number of possible task class partitions is clearly exponential in the number of task classes. A way to overcome this is to prioritize the classes based on the advantages that they provide and their MCO compatibility with other classes. The paper proposes two class partitioning approaches. In the first, task classes are prioritized according to their delay constraints. The second approach uses a hierarchical ordering that first prioritizes task classes according to power consumption, and then prioritizes task classes within groups of similar power demand according to task delays. We show that the hierarchical algorithm produces significantly better solutions than the first one. Both are easily implementable and computationally efficient. As a result, making the offloading decisions requires little communication and computation overhead and the delay introduced by waiting for the offloading decisions is low, which is advantageous when task deadlines are tight.

The paper considers two different design problems. In the *Basic Static Task Class Partitioning (BSTCP)* problem, the wireless channel capacities at each BS and the ES computational capacity are given beforehand. The problem in this case is to create the task class partition that minimizes the average MD power consumption while satisfying the task execution deadline constraints. We show that the proposed static class partitioning algorithms can significantly reduce the MD power consumption without violating task delay constraints. The second problem is where the channel and execution resources are not given, and the algorithms not only partition the task classes as before, but determine the channel and execution resource assignments, subject to a cost budget constraint, as well. This is referred to as the *Joint Static Task Class Partitioning and Network Resource Allocation* (JSTCP) problem.

Due to the static nature of the partitioning, the proposed algorithms are intended for use in situations where the set of task classes remain relatively stable over the time periods of interest. This is not a strong assumption, since although the mixture of tasks may continuously change, the *classes* in which these tasks belong may not do so. When task class partitioning is combined with resource allocation in JSTCP, the problem is inherently one of static resource allocation. The static assignment of wireless channel bandwidth and computational capacity is subject to a cost budget and resources are not re-allocated during network operation. In a design scenario using our algorithms for example, a network designer could assign resources based on their knowledge (or prediction) of the characteristics of the traffic flow that would be impinging on the resulting network. In a practical design, this would likely be done using worst-case traffic assumptions (e.g., anticipated "busy hour" statistics.). The algorithms in our paper could easily be adapted to more quasi-static situations, where resource assignments are updated periodically based on evolving worst-case (busy-hour) traffic conditions. Overall, static TCP is a natural first step towards more sophisticated settings, e.g., dynamic TCP, which we leave for future work.

The main contributions of the paper are summarized below.

- STCP is proposed as a way of improving the performance of computation offloading under task execution time constraints.
- Two main task class partitioning algorithms are proposed. The simpler algorithm uses a latency-based task class ordering. The more sophisticated hierarchical algorithm first sorts the task classes into sets based on their mean power consumption, and within each set, the task classes are further ordered using a task completion time criterion.
- The design algorithms consider both soft and hard task deadlines. In soft deadlines (SDs), the probability of task completion deadline violation is upper bounded by a class-specific value. Since the soft delay constraints are based on satisfying statistical bounds, the paper models the detailed delay distribution of the tasks. The models permit arbitrary task upload and execution time distributions with any set of Markovian channel models. In the hard deadline (HD) case, task deadlines must *always* be respected. This is done by including Concurrent Local Execution (CLE) [8] in the algorithms. The paper provides the only known mechanism for class partitioning that always achieves this goal.
- Both BSTCP and JSTCP problems are introduced. In the former, STCP is applied for a network with preallocated resources, while in the latter STCP is jointly studied with network resource allocation. Both BSTCP and JSTCP are studied in the soft and hard class deadline constraint cases.
- A variety of simulation results are presented that demonstrate the good performance of the proposed algorithms. In the JSTCP case, our algorithms perform well compared to both a state-of-the-art no-partitioning algorithm and the optimal no-partitioning algorithm.

The remainder of the paper is organized as follows. In Section II, the prior work most related to our paper is reviewed. The system model is then described in Section III. In Section IV, the general task offloading decision problems for both SD and HD are formulated. Following this, in Section V, the partitioning algorithms are proposed. In Section VI, the partitioning algorithms are used to obtain solutions of the BSTCP problem. In Section VII, the JSTCP problem is formulated under a resource cost budget for both SD and HD cases. In Section VIII, simulation results that demonstrate the proposed designs are given. Finally, we present the conclusions of the work in Section IX.

II. Related Work

This section reviews the most recent literature that relates to our paper. In MCO, task completion time and MD energy consumption are two important performance metrics. In

terms of task completion time, some work considers tasks with hard completion deadlines under static wireless channel conditions [12]–[14], in which case the fixed channel gains make it possible to predict the wireless transmission time for an offloaded task. When the channel conditions vary with time, the uncertainty in future channel condition results in uncertainty in the completion time of offloaded tasks. In this case, an offloaded task may be discarded if it is not completed before the required deadline [7]. Given the difficulty to satisfy hard completion deadlines for offloaded tasks, some work considers mean delay performance of tasks in MCO, such as [15], [16], and other work makes offloading decisions to achieve a certain statistical delay bound [17], [18]. In addition to the time-varying wireless channel conditions, the random task arrival process also causes uncertainty in the future network conditions, which affects the MCO performance and makes it difficult to support tasks with strict completion time requirements. In [10], [11], offloading decisions for tasks are made by considering the statistic information of task arrivals and the mean completion time of the tasks. Different from the existing work, we study MCO with both time-varying wireless transmission channels and random task arrivals, and consider both hard and soft task completion deadlines.

Offloading decisions in MCO are often jointly considered with network resource allocations to improve the offloading performance. For example, in [19], [20], no constraints are specified for task completion time. Instead, the objective is to optimize a cost or utility function that is defined as a weighted sum of the MD energy consumption and average task completion time by jointly coordinating the co-channel interference and the task offloading decisions. In [14], orthogonal channel allocations are jointly optimized with the task offloading decisions in order to minimize the MD energy consumption subject to hard task completion time under static channel conditions.

For tasks with strict completion time requirements, making the offloading decisions should take the minimal amount of time upon the task arrival. To make the offloading decisions, communication time may be required to collect input information needed for making the decisions and deliver the decisions, depending on the specific algorithms, and computation time is needed to run the decision making algorithms [15], [16]. However, most existing work ignores the decision making time upon the task arrivals. In our work, the class partitioning is performed offline; and upon a task arrival, if the task class is in the set that is allowed to offload, the MD only needs to check if a channel is available in order to decide whether the task can be offloaded. The response time to make the offloading decision is low.

An offloading decision can be binary, which means that a given task is either entirely executed at the MD or offloaded to an ES. Instead, in partial offloading, different portions of a task are allowed to be executed simultaneously at both the MD and the ES [21] or at multiple ESs [22]. The additional flexibility of partial offloading helps reduce the task completion time, while saving energy for the MDs. In networks with highly dynamic topology, such as vehicular networks, partial offloading allows an MD to offload different portions of a task to different computing devices in order to take advantage of the short connection time to these computing devices [23], [24]. Also, a task can be divided continuously [21], [22], [25], or it can be modelled as mutually related subtasks and a binary offloading decision is made for each subtask [23], [24], [26]. Different from partial offloading and individual task partitioning, our work considers *task class* partitioning (TCP), which partitions task classes into two sets for either remote or local execution. TCP can be combined with partial task offloading. However, in order to focus on the benefit of TCP, we do not apply partial offloading for individual tasks, i.e., after we partition task classes into locally executed or offloaded, each task belonging to the latter is either entirely offloaded (if a channel is available for data uploading) or executed locally (when no channel is available). We leave the study of the combination of TCP with partial task offloading to future work.

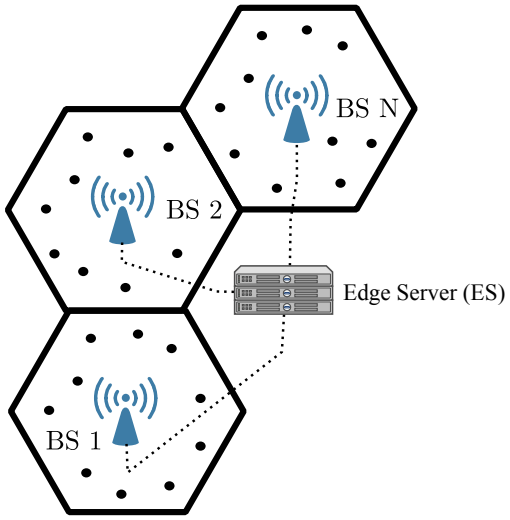


Fig. 1: System Model

III. System Model

As shown in fig. 1, we consider the problem where wireless channels at a set of base stations and ES capacity is leased for computation offloading. Let N be the total number of BSs, $\mathcal{N} = \{1, 2, \ldots, N\}$ be the set of BSs and x_n be the number of wireless channels that are used in BS n. Since the ES is located at the edge of the network, we focus on the dominant sources of delay, i.e., wireless access at the BSs and task execution at the ES [2]. Tasks generated by an MD can be offloaded through the wireless network and executed on the ES. Let f^{C} be ES capacity (in the number of CPU cycles per second) and a fraction y, $0 \leq y \leq 1$, of the capacity is used.

There are J classes of tasks generated by the MDs. Let $\mathcal{J} = \{1, 2, \ldots, J\}$ be the set of task classes. A class j task is defined by parameters s_j, q_j, and d_j, where s_j (in bits) is the input data size needed for processing the task, q_j (in number of CPU cycles) is the computation load, and d_j (in seconds) is the task deadline. A class j task may be executed locally, in which case $T_j^{\mathrm{L}} = q_j / f^{\mathrm{L}}$ and $p^{\mathrm{L}} T_j^{\mathrm{L}}$, respectively, are the amounts of time and energy needed to process the task, where f^{L} is the local processing speed in number of

CPU cycles per second and p^{L} is the local processing power in Watts. We assume that $T_j^{\mathrm{L}} < d_j$ for all $j \in \mathcal{J}$ so that the delay requirement can always be satisfied if a task is executed locally.

A task may be processed remotely at the ES. Let $t_{n,j}^{\mathrm{OFF}}$ be the total amount of time for offloading a class j task generated by an MD in BS n, then $t_{n,j}^{\mathrm{OFF}} = t_{n,j}^{\mathrm{W}} + t_j^{\mathrm{C}}$, where $t_{n,j}^{\mathrm{W}}$ and t_j^{C}, respectively, are the amount of time for uploading the task to the ES and the amount of time the task experiences at the ES before the completion of its execution. Note that $t_{n,j}^{\mathrm{W}}$ is a function of s_j and the wireless data transmission rate. We consider fixed transmission power at each MD and it adapts its transmission rate according to the current channel conditions. In this case, the amount of time needed to upload the s_j bits in order to offload a class j task is determined by the channel conditions.

All the offloaded tasks from different BSs and in different classes are processed at the ES according to a first-come-first-served queueing discipline. For a class j task, the execution time at the ES is $T_j^{\mathrm{C}} = q_j/(yf^{\mathrm{C}})$. In addition to execution time, queueing at the ES may incur additional latency. Let λ^{ES} be the mean aggregate task arrival rate at the ES of all traffic classes from all BSs. The queueing delay of the offloaded tasks at the ES w^{C} is a function of λ^{ES}, and its statistics are the same for all tasks belonging to all classes and generated at all BSs. Since $t_j^{\mathrm{C}} = w^{\mathrm{C}} + T_j^{\mathrm{C}}$, we have $t_{n,j}^{\mathrm{OFF}} = t_{n,j}^{\mathrm{W}} + w^{\mathrm{C}} + T_j^{\mathrm{C}}$. Both $t_{n,j}^{\mathrm{W}}$ and w^{C} are random due to the randomness of the link gain of the wireless channel and the task arrival process at the ES, respectively. Therefore, $t_{n,j}^{\mathrm{OFF}}$ is also random.

IV. Problem Formulation

The objective of this work is to minimize the mean power consumption of the MDs used for processing their tasks, when the latter can be either executed locally by their MD or be offloaded to the ES for execution, and under task delay constraints.

Both soft and hard delay constraints are considered. In the soft delay constraints case, the delay requirements of tasks are achieved with a predefined probability, i.e., $1 - \varepsilon_j$ with $\varepsilon_j \in (0, 1)$. Hence, our goal is to minimize the mean mobile power consumption such that the probability that task execution deadline violation is bounded, i.e., the deadline constraints *can be violated*, albeit rarely.

In the hard delay constraints case, the latency requirements of all the tasks should be *always satisfied*. Due to the random wireless channel conditions and task arrival process, guaranteeing the hard delay constraints is difficult for offloaded tasks. In [8] we proposed to use concurrent local execution (CLE) when it is uncertain that an offloaded task can be completed before its deadline. More specifically, for an offloaded task, if it is not completed before time $t_j^{\mathrm{CLE}} = d_j - T_j^{\mathrm{L}}$, the MD starts to execute the task locally and in parallel to the MCO. CLE aborts if the ES completes the task before d_j in order to save the energy consumption of the MD. Again, the goal is to make power efficient offloading decisions which respect all task deadlines.

We extend the notion of a task class to the notion of a *BS-class* tuple as follows: $(n, j) \in \mathcal{N} \times \mathcal{J}$ is the set of class j tasks arriving at BS n. Note that this allows us to distinguish between the same task classes submitted to different BSs, so that the same task class may be treated differently at different BSs. In both soft and hard delay constraints cases, we adopt a two-step process in making the offloading decisions, as shown in fig. 2:

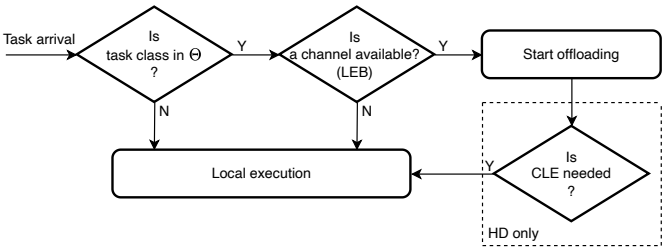


Fig. 2: Offloading Decision Making Process

1) **Class set partitioning:** In this step, we partition the set $\mathcal{N} \times \mathcal{J}$ into subsets Θ and $(\mathcal{N} \times \mathcal{J}) \setminus \Theta$. Tasks in BS-classes $(\mathcal{N} \times \mathcal{J}) \setminus \Theta$ are always executed locally at the MDs. Offloading decisions for tasks from the BS-classes of Θ are made in the second step.
2) **Local execution on blocking (LEB):** This step applies only to tasks belonging in BS-classes of Θ. Upon the arrival of a class j task at BS n with $(n, j) \in \Theta$, if at least one of the x_n channels is available for immediate use, the MD reserves a free channel and starts uploading the task; otherwise the task is blocked from offloading and is executed locally at the MD. Note that LEB treats tasks from different task classes the same, and, therefore, within the same BS, the same blocking probability applies to tasks of any class in $\mathcal{J}$.

Observations:

- The class set partitioning of the first step is performed offline, as preprocessing to MCO, and does not cause any additional delays to task processing. On the other hand, the LEB decision for a task is made immediately upon the task arrival. LEB requires minimal communication and computation overhead, and can be made with nearly zero additional delay. Overall, the extra delay introduced to the offloading decisions is negligible.
- Class set partitioning introduces a pre-offloading stage of 'task filtering', which will 'filter-out' tasks that would burden the offloading process due to their combination of parameters. For example, if tasks of class j have large data size s_j and very tight deadline d_j, then the tasks should not be offloaded, since they require a long channel uploading time and have a high probability of missing their deadlines.
- With this two-step process for making the offloading decisions, either local execution or remote offloading is initiated *immediately at task release time*, which may be advantageous when task deadlines are tight. It also provides a simple mechanism for assessing the ES workload at the MD. That is, when more wireless channels are used for uploading tasks, a higher computation load is at the ES, in which case executing tasks locally at the MDs is

beneficial.

The problem formulation is presented first for soft completion deadlines, and then for hard completion deadlines.

A. Soft deadlines

With the soft delay constraints of the tasks, the power consumption of the MDs includes

- E^{DL}, which is the mean power consumption of the MDs for executing tasks belonging to classes in $(\mathcal{N}\times\mathcal{J})\setminus\Theta$,
- E^{L}, which is the mean power consumption of the MDs for executing tasks belonging to classes in Θ locally (these tasks are blocked for offloading due to no wireless channels upon their arrivals), and
- E^{T}, which is the mean power consumption of the MDs for uploading tasks belonging to classes in Θ.

The objective of class partitioning, together with the allocations of wireless and computation resources in case the latter are not given beforehand, is to minimize the mean power consumption of the MDs, i.e.,

$$\min E^{\mathrm{MD}} = E^{\mathrm{DL}} + E^{\mathrm{L}} + E^{\mathrm{T}} = \sum_{n=1}^{N} \left(E_n^{\mathrm{DL}} + E_n^{\mathrm{L}} + E_n^{\mathrm{T}}\right), \tag{1}$$

where E_n^{DL} is the portion of E^{DL} consumed by the MDs in the nth BS, E_n^{L} is the portion of E^{L} consumed by the MDs in the nth BS, and E_n^{T} is the portion of E^{T} consumed by the MDs in the nth BS. These "BS n portions" are given below.

Let $\lambda_{n,j}$ be the mean arrival rate of class j tasks in BS n. The local energy consumption of the MDs in BS n for processing class j tasks with $(n,j)\notin\Theta$ generated in one second is

$$E_n^{\mathrm{DL}} = \sum_{j:(n,j)\in(\mathcal{N}\times\mathcal{J})\setminus\Theta} \lambda_{n,j} T_j^{\mathrm{L}} p^{\mathrm{L}}. \tag{2}$$

Since this is the mean energy consumption for tasks generated in one second, it is equivalent to the mean power consumption of the MDs for processing the tasks over a long term. Similarly, E_n^{L} and E_n^{T} given below are also the mean power consumption of the MDs.

For BS n, let P_n^{B} be the task blocking probability for all tasks with $(n,j)\in\Theta$ due to no channel being available upon their arrival. Recall that all tasks are treated the same at a BS for accessing a channel, and therefore, the blocking probability does not depend on task class j. How to find P_n^{B} is given in Section VI-B. The mean power consumption of all the MDs at BS n for processing these tasks is

$$E_n^{\mathrm{L}} = \sum_{j:(n,j)\in\Theta} P_n^{\mathrm{B}} \lambda_{n,j} T_j^{\mathrm{L}} p^{\mathrm{L}}. \tag{3}$$

The unblocked tasks in Θ are offloaded through the wireless channels to the ES. The mean power consumption of the MDs in BS n for uploading these tasks is

$$E_n^{\mathrm{T}} = \sum_{j:(n,j)\in\Theta} (1-P_n^{\mathrm{B}}) \lambda_{n,j}\ \bar{t}_{n,j}^{\mathrm{W}}\ p^{\mathrm{T}}, \tag{4}$$

where $\bar{t}_{n,j}^{\mathrm{W}}$ is the mean of $t_{n,j}^{\mathrm{W}}$, and can be calculated based on the statistics of the channel conditions as shown in Section VI-A.

In the soft task deadlines case, the objective of minimizing the mean MD power consumption is subject to the delay constraints

$$\Pr[t_{n,j}^{\mathrm{OFF}} \le d_j] = \Pr[t_{n,j}^{\mathrm{W}} + w^{\mathrm{C}} + T_j^{\mathrm{C}} \le d_j] \ge 1-\varepsilon_j \tag{5}$$

for all n and j. Finding the distribution of $t_{n,j}^{\mathrm{OFF}}$ is given in Section VI-D.

B. Hard deadlines

The objective of minimizing the mean power consumption of the MDs can be written as

$$\min E^{\mathrm{MD}} = E^{\mathrm{DL}} + E^{\mathrm{L}} + E^{\mathrm{T}} + E^{\mathrm{CLE}}, \tag{6}$$

where the expressions for E^{DL}, E^{L}, and E^{T} are same as in the soft deadline case. When the task deadlines are *hard* and must be respected, CLE [8] is used, i.e., the local execution of a task belonging to class j is initiated at $t_j^{\mathrm{CLE}} = d_j - T_j^{\mathrm{L}}$, if offloading is still ongoing.

The additional power consumption of the MDs incurred by CLE is

$$E^{\mathrm{CLE}} = \sum_{(n,j)\in\Theta} \left(E_{n,j}^{\mathrm{O}} + E_{n,j}^{\mathrm{B}}\right), \tag{7}$$

where $E_{n,j}^{\mathrm{O}}$ is the amount of local execution power when offloading is completed before the task completion deadline and $E_{n,j}^{\mathrm{B}}$ is the amount of local execution power when offloading continues beyond the task deadline.

When $t_{n,j}^{\mathrm{OFF}} < d_j$, the local execution time incurred by CLE is $t_{n,j}^{\mathrm{OFF}} - t_j^{\mathrm{CLE}}$ because the CLE stops as soon as the offloading is completed; and when $t_{n,j}^{\mathrm{OFF}} \ge d_j$, the local execution time incurred by CLE is T_j^{L}. Based on these two cases, $E_{n,j}^{\mathrm{O}}$ and $E_{n,j}^{\mathrm{B}}$ are

$$E_{n,j}^{\mathrm{O}} = (1-P_n^{\mathrm{B}})\lambda_{n,j} p^{\mathrm{L}} \int_{t=t_j^{\mathrm{CLE}}}^{d_j} (t - t_j^{\mathrm{CLE}}) \Pr\left[t_{n,j}^{\mathrm{OFF}} = t\right] dt, \tag{8}$$

$$E_{n,j}^{\mathrm{B}} = (1-P_n^{\mathrm{B}})\lambda_{n,j} p^{\mathrm{L}} T_j^{\mathrm{L}} \int_{d_j}^{\infty} \Pr\left[t_{n,j}^{\mathrm{OFF}} = t\right] dt. \tag{9}$$

V. Class Partitioning

One of the main contributions of this work is the development of class partitioning algorithms that lead to the efficient implementation of the two-step computational offloading decisions described in Section IV. Specifically, the algorithms determine to partition the set $(\mathcal{N}\times\mathcal{J})$ into sets Θ and $(\mathcal{N}\times\mathcal{J})\setminus\Theta$, so that tasks belonging to class j in BS n proceed to the second step of offloading decisions if $(n,j)\in\Theta$, while other tasks are automatically executed locally.

Instead of going through all $2^{NJ}+1$ subsets of $\mathcal{N}\times\mathcal{J}$ in order to find the best Θ, we first sort the NJ (n,j)-tuples in $\mathcal{N}\times\mathcal{J}$ into an ordered list $\eta_1,\eta_2,\ldots,\eta_{NJ}$ according to a certain criterion, and then partition this list into subsets $\Theta = \{\eta_1,\eta_2,\ldots,\eta_\theta\}$ and $(\mathcal{N}\times\mathcal{J})\setminus\Theta = \{\eta_{\theta+1},\eta_{\theta+2},\ldots,\eta_{NJ}\}$, where $\theta = |\Theta|$. The rest of the section introduces sorting criteria and methods to determine θ.

A. Partitioning based on delay constraints (PDC)

The first (and simpler) method orders task classes according to the delay constraints (5). The intuition behind it is that if there is a class of tasks with extremely tight deadline, they should not be allowed to offload, since it will most likely require too many resources to satisfy their deadline constraint, assuming the deadline can indeed be satisfied by offloading. As a result, the system will offload fewer tasks (with the rest executed locally on MDs). Even if such tasks are offloaded (possibly at great cost), there is still a high probability that offloading will fail to satisfy their tight delay constraints due to the stochastic nature of the service system.

The PDC method includes two algorithms, PDC-S for SDs given in Algorithm 1 and PDC-H for HDs given in Algorithm 2.

Soft deadlines: Let λ^{ES} be the aggregate mean task arrival rate at the ES. We notice that $\Pr[t^{\mathrm{W}}_{n,j} + w^{\mathrm{C}} + T^{\mathrm{C}}_j \leq d_j]$ in (5) is a decreasing function of λ^{ES}. Let μ^{C}_j be the number of class j tasks that the ES can process in one second, i.e., $\mu^{\mathrm{C}}_j = \frac{1}{T^{\mathrm{C}}_j}$. If one assumed that all offloaded tasks arriving at the ES come from a single BS n and belong to a single task class j, binary search in the range $[0, \mu^{\mathrm{C}}_j]$ can be used to determine the maximum λ^{ES} that would satisfy delay constraint (5) in this case. Namely,

$$\hat{\lambda}^{\mathrm{ES}}_{n,j} = \max\{\lambda^{\mathrm{ES}} : \Pr\left[t^{\mathrm{W}}_{n,j} + t^{\mathrm{C}}_j \leq d_j\right] \geq 1 - \varepsilon_j\}. \quad (10)$$

Note that although the values $\hat{\lambda}^{\mathrm{ES}}_{n,j}$, calculated by (10), correspond to the simple scenario of one BS and one task class, they can be used as an indication about the possibility that tasks belonging to different classes can meet their delay constraints if offloaded. Therefore, we use them in the general setting below when making class partitioning decisions.

If $\hat{\lambda}^{\mathrm{ES}}_{n,j} = 0$ for a given (n,j)-tuple, tasks in class j from BS n are executed locally, i.e., these (n,j)-tuples will not be included in Θ. This direct assignment to local execution makes sense, since the delay constraints for these tasks cannot be satisfied, even if they were the only kind of tasks offloaded to the ES. Next, a set $\mathcal{R}$ is formed to include all (n,j)'s with $\hat{\lambda}^{\mathrm{ES}}_{n,j} > 0$ as shown in line 2 of Algorithm 1, where $R = |\mathcal{R}|$. The task classes in set $\mathcal{R}$ are ordered in decreasing values of $\hat{\lambda}^{\mathrm{ES}}_{n,j}$ (line 3), since the smaller $\hat{\lambda}^{\mathrm{ES}}_{n,j}$ is, the tighter the corresponding delay constraint is. The algorithm employs a linear search process for the best θ, starting from 0 and going up to R. For a given θ, Θ is the set of the first θ (n,j)-tuples in the ordering of $\mathcal{R}$. Note that when $\theta = 0$, no tasks are allowed to offload; and when $\theta = R$, all the tasks with $\hat{\lambda}^{\mathrm{ES}}_{n,j} > 0$ are allowed to offload. The algorithm compares the resulting mean MD power consumption for each θ, and returns the Θ corresponding to the θ that achieved the minimum mean MD power consumption.

Hard deadlines: While (5) is an explicit constraint in the problem formulation for soft deadlines, the hard deadlines formulation does not contain such a constraint, since CLE guarantees the task completion before its deadline. However, (5) (and (10)) can still be used to obtain an ordering of the (n,j)-tuples. The basic procedure is as follows. Starting from $\Theta = \mathcal{N} \times \mathcal{J}$ and $\varepsilon_j = \varepsilon = 1$ for all j, the common ε is reduced in steps of size ω. For every reduction, we calculate values $\hat{\lambda}^{\mathrm{ES}}_{n,j}$ exactly as in the SD case by using the current ε in constraints (10). For each (n,j) with $\hat{\lambda}^{\mathrm{ES}}_{n,j} = 0$, we remove (n,j) from Θ. This process continues, until θ classes remain in Θ. Algorithm PDC-H (Algorithm 2) implements this process. Note that in Algorithm 2, θ decreases from NJ down to 0 (instead of increasing from 0 to NJ), because this allows us to calculate our solutions for different θ with only one swiping decrease of ε down from 1.

Algorithm 1 PDC-S

1: Find $\hat{\lambda}^{\mathrm{ES}}_{n,j}$, for all n and j, from (10)
2: $\mathcal{R} = \{(n,j) : \hat{\lambda}^{\mathrm{ES}}_{n,j} > 0\}$; $R = |\mathcal{R}|$
3: Sort classes in $\mathcal{R}$ in decreasing order of $\hat{\lambda}^{\mathrm{ES}}_{n,j}$; let $\eta_1, \eta_2, \ldots, \eta_R$ be the resulting ordering
4: $\theta^* = 0$; $E_{\min} = \infty$
5: **for all** $\theta = 0, 1, 2, \ldots, R$ **do**
6: $\quad\Theta = \{\eta_1, \eta_2, \ldots, \eta_\theta\}$
7: $\quad$Find mean MD power consumption E^{MD}
8: $\quad$**if** $E^{\mathrm{MD}} < E_{\min}$ **then**
9: $\quad\quad E_{\min} = E^{\mathrm{MD}}$; $\theta^* = \theta$
10: $\quad$**end if**
11: **end for**
12: **return** $\Theta^* = \{\eta_1, \eta_2, \ldots, \eta_{\theta^*}\}$

Algorithm 2 PDC-H

1: $\Theta = \mathcal{N} \times \mathcal{J}$; $E_{\min} = \infty$
2: $\varepsilon_j = \varepsilon = 1,\ \forall j \in \mathcal{J}$
3: **for all** $\theta = NJ, NJ-1, \ldots, 2, 1, 0$ **do**
4: $\quad$**while** $|\Theta| > \theta$ **do**
5: $\quad\quad$**for all** $(n,j) \in \mathcal{N} \times \mathcal{J}$ **do**
6: $\quad\quad\quad$Find $\hat{\lambda}^{\mathrm{ES}}_{n,j}$ from (10)
7: $\quad\quad\quad$**if** $\hat{\lambda}^{\mathrm{ES}}_{n,j} = 0$ **then**
8: $\quad\quad\quad\quad\Theta = \Theta \setminus \{(n,j)\}$
9: $\quad\quad\quad$**end if**
10: $\quad\quad\quad$**if** $|\Theta| = \theta$ **then**
11: $\quad\quad\quad\quad$break;
12: $\quad\quad\quad$**end if**
13: $\quad\quad$**end for**
14: $\quad\quad\varepsilon = \varepsilon - \omega$; $\varepsilon_j = \varepsilon$ for all $j \in \mathcal{J}$
15: $\quad$**end while**
16: $\quad$Find mean MD power consumption E^{MD}
17: $\quad$**if** $E^{\mathrm{MD}} < E_{\min}$ **then**
18: $\quad\quad E_{\min} = E^{\mathrm{MD}}$; $\Theta^* = \Theta$
19: $\quad$**end if**
20: **end for**
21: **return** Θ^*

B. Hierarchical partitioning (HP)

Unlike the PDC algorithms, the hierarchical partitioning algorithm takes into account both the mean power consumption and the task deadlines when ordering the tasks. This is implemented in two phases. In the first phase, task classes are ordered and grouped according to an estimated power

consumption criterion. In the second phase, classes within each group are further ordered using a second criterion that takes task deadlines into account. This partitioning method is given in Algorithm 3, where g is the index of the current group of (n,j)-tuples and h is the index of the first (n,j)-tuple in the current group.

In the first phase of ordering, we prefer to offload the tasks that can save more power by offloading than by executing locally. Thus, the sorting criterion is the difference between power consumption of the MD for offloading the task and executing it locally, which is defined as

$$\Phi^{\mathrm{E}}_{n,j} = p^{\mathrm{T}} \frac{s_j}{\bar{B}_{n,j}} - p^{\mathrm{L}} T^{\mathrm{L}}_j, \tag{11}$$

where $\bar{B}_{n,j}$ is the expected wireless transmission rate of class j tasks in BS n with the expectation taken over the random channel conditions. An example is provided in Section VIII. The (n,j)-tuples of $\mathcal{N}\times\mathcal{J}$ are ordered in increasing values of $\Phi^{\mathrm{E}}_{n,j}$. After this initial ordering, the (n,j)-tuples whose $\Phi^{\mathrm{E}}_{n,j}$ values differ by at most ξ (for some parameter $0 \le \xi \le 1$) are grouped together (lines 2-9).

Within the same group, the (n,j)-tuples are further ordered according to a second set of values defined as follows:

$$\Phi^{\mathrm{D}}_{n,j} = \frac{s_j/\bar{B}_{n,j} + q_j/(yf^C)}{d_j} = \frac{s_j}{\bar{B}_{n,j} d_j} + \frac{q_j}{yf^C d_j}. \tag{12}$$

That is, $\Phi^{\mathrm{D}}_{n,j}$ is a ratio between the minimum mean offloading time (sum of mean wireless transmission time and ES execution time) over the deadline of class j tasks. Note that $\Phi^{\mathrm{D}}_{n,j}$ does not consider any queueing delay at the ES. This is because the ES service system has a single queue for all offloaded tasks, so that all task classes experience the same mean queueing delay, and as a result, queueing delay does not affect the task class ordering. Within each group, the (n,j)-tuples are sorted in increasing values of $\Phi^{\mathrm{D}}_{n,j}$ (lines 10-13).

Parameter ξ regulates the trade-off between the effects of delay constraints and power consumption on class ordering. When $\xi = 0$, there is one and only one BS-class in each group, and the sorting is based only on $\Phi^{\mathrm{E}}_{n,j}$, i.e., the second phase does not have any effect. When $\xi = 1$, all the BS-classes belong in one group after the first phase, and the sorting is solely based on $\Phi^{\mathrm{D}}_{n,j}$ in the second phase.

Finally, lines 15-22 find the partition that achieves the minimum MD power consumption.

Running times: Let $\mathcal{T}$ be the running time of finding the mean MD power consumption in line 7 of Algorithm 1, line 16 in Algorithm 2, and line 18 in Algorithm 3.

- For PDC-S given in Algorithm 1, line 1 takes time $O(NJ\log\frac{\mu^{\mathrm{C}}}{\epsilon})$, where μ^{C} is the maximum of μ^{C}_j and ϵ is the desired accuracy of the binary search, and line 3 takes time $O(NJ\log NJ)$. The running time of PDC-S is $O(NJ\log\frac{\mu^{\mathrm{C}}}{\epsilon} + NJ\log NJ + NJ\mathcal{T})$.
- For PDC-H given in Algorithm 2, within the for loop between lines 3 and 20, the running time of lines 4-15 is $O(\frac{1}{\omega}NJ\log\frac{\mu^{\mathrm{C}}}{\epsilon})$. The running time of PDC-H is $O(\frac{1}{\omega}(NJ)^2\log\frac{\mu^{\mathrm{C}}}{\epsilon} + NJ\mathcal{T})$.
- For HP, the running time of Algorithm 3 is $O(NJ\log(NJ) + NJ\mathcal{T})$.

Algorithm 3 HP

1: Sort the (n,j)-tuples in $\mathcal{N}\times\mathcal{J}$ in increasing order of $\Phi^{\mathrm{E}}_{n,j}$; let $\eta_1,\eta_2,\cdots,\eta_{NJ}$ be the resulting ordering
2: $g=1$; $h=1$; $\mathcal{G}_1=\{\eta_1\}$
3: **for** $j=2$ **to** NJ **do**
4: **if** $\left|\frac{\Phi^{\mathrm{E}}_{\eta_j}-\Phi^{\mathrm{E}}_{\eta_h}}{\Phi^{\mathrm{E}}_{\eta_h}}\right| < \xi$ **then**
5: $\mathcal{G}_g = \mathcal{G}_g \cup \{\eta_j\}$
6: **else**
7: $g=g+1$; $h=j$; $\mathcal{G}_g=\{\eta_h\}$
8: **end if**
9: **end for**
10: $\mathcal{L}=$ null {$\mathcal{L}$ is a list of (n,j)-tuples}
11: **for** $g=1$ **to** G **do**
12: Sort the (n,j)-tuples in group g in increasing order of $\Phi^{\mathrm{D}}_{n,j}$ and append the ordered tuples to $\mathcal{L}$
13: **end for**
14: Let $\mathcal{L}=\eta'_1,\eta'_2,\cdots,\eta'_{NJ}$ be the ordered (n,j) list
15: $\theta^*=0$; $E_{\min}=\infty$
16: **for** $\theta=0$ **to** NJ **do**
17: $\Theta=\{\eta'_1,\eta'_2,\ldots,\eta'_\theta\}$
18: Find mean MD power consumption E^{MD}
19: **if** $E^{\mathrm{MD}} < E_{\min}$ **then**
20: $E_{\min}=E^{\mathrm{MD}}$; $\theta^*=\theta$
21: **end if**
22: **end for**
23: **return** $\Theta^*=\{\eta'_1,\eta'_2,\ldots,\eta'_{\theta^*}\}$

Observation: In line 7 of Algorithm 1, line 16 of Algorithm 2, and line 18 of Algorithm 3, finding the mean MD power consumption E^{MD} requires specific information regarding the task classes, wireless channel statistics, resource availability of the network, etc. In the following two sections, the mean MD power consumption is found for two scenarios, one with fixed x_n's and y, and the other with a cost budget that limits the total cost of the wireless channels and ES computing resources.

VI. Preallocated Network Resources

In this section, we consider the BSTCP problem, i.e., the case where the available resources for the network have been preallocated, i.e., x_n's and y are fixed and given. We assume that the arrival process of class j tasks at BS n follows a Poisson process with mean arrival rate $\lambda_{n,j}$. The assumption is a common one, and justified by the large number of MDs that generate the tasks [27]. As a result, and because of the PASTA rule [28], the ES service system can be modeled as an $M/G/1$ queue. We model the wireless channels between the MDs and the BSs as discrete-time Markov processes. The time slot duration is denoted by τ in seconds. The Markovian transition probabilities are defined in the usual way, i.e., given the channel state in the current time slot, there is a probability associated to its transition to another state in the next time slot. These probabilities are functions of the radio propagation environment that the MDs experience at the BS. Let $\mathcal{K}_n=\{1,2,\ldots,K_n\}$ be the set of different

possible Markovian wireless channel models of BS n, and let $K_n = |\mathcal{K}_n|$.

A. Distributions of $t^W_{n,j}$

Define $P^{\mathrm{G}}_{n,j,k}$ as the probability that a class j task, offloaded by an MD in BS n, encounters channel model k. The distribution of $t^{\mathrm{W}}_{n,j}$ is given as

$$\Pr[t^{\mathrm{W}}_{n,j} = l] = \sum_{k \in \mathcal{K}_n} \Pr[t^{\mathrm{W}}_{n,j,k} = l] P^{\mathrm{G}}_{n,j,k}, \tag{13}$$

where $t^{\mathrm{W}}_{n,j,k}$ (in number of time slots) is the amount of time for uploading a class j task in BS n when the channel follows propagation model k. Given the Markov channel state transition probabilities, the distribution of $t^{\mathrm{W}}_{n,j,k}$ is given in [8]. The mean of $t^{\mathrm{W}}_{n,j}$ can be found as

$$\bar{t}^{\mathrm{W}}_{n,j} = \sum_{l=0}^{\infty} l \Pr[t^{\mathrm{W}}_{n,j} = l]. \tag{14}$$

B. Computing P^{B}_n

P^{B}_n is channel blocking probability for tasks in BS n with $(n,j) \in \Theta$. For BS n, the mean task arrival rate for all classes with $(n,j) \in \Theta$ is $\sum_{j:(n,j)\in\Theta} \lambda_{n,j}$, and the mean service time (uploading time through wireless transmissions) of the tasks is

$$\bar{t}^{\mathrm{W}}_n = \frac{\sum_{j:(n,j)\in\Theta} \lambda_{n,j} \bar{t}^{\mathrm{W}}_{n,j}}{\sum_{j:(n,j)\in\Theta} \lambda_{n,j}}. \tag{15}$$

Given that the task arrivals follow a Poisson process, the Erlang-B formula can be used to find P^B_n even for non-exponentially distributed task uploads, as in [29]. This is referred to as the *insensitivity property* of the formula [30]. Thanks to this property, the Erlang-B result holds for any service time distribution with the same mean.

C. Distribution of w^{C}

The aggregate task arrival process at ES is Poisson [31], and the service system at the ES is an $M/G/1$ queue. As arriving tasks sample the asymptotic equilibrium state distribution of the ES, the statistics of the queueing time experienced at the ES by tasks in different classes and from different BSs are the same. The queueing time distribution depends on the mean aggregate task arrival rate at the ES and the distribution of the execution time of the tasks.

The mean task arrival rate at the ES is

$$\lambda^{\mathrm{ES}} = \sum_{(n,j)\in\Theta} \lambda_{n,j}(1 - P^{\mathrm{B}}_n). \tag{16}$$

Let T^{C} be a random variable representing the execution time of a task arriving at the ES. The probability that $T^{\mathrm{C}} = T^{\mathrm{C}}_j$ is equal to the probability that a class j task arrives at the ES. Since

$$\Pr[T^{\mathrm{C}} = T^{\mathrm{C}}_j] = \frac{\sum_{n:(n,j)\in\Theta}(1 - P^{\mathrm{B}}_n)\lambda_{n,j}}{\lambda^{ES}}, \tag{17}$$

the probability density function of T^{C} is

$$f_{T^{\mathrm{C}}}(\tilde{b}) = \sum_{j:(n,j)\in\Theta} \Pr\left[T^{\mathrm{C}} = T^{\mathrm{C}}_j\right] \delta\left(\tilde{b} - T^{\mathrm{C}}_j\right), \tag{18}$$

and the Laplace-Stieltjes transform of $f_{T^{\mathrm{C}}}(\tilde{b})$ is given by

$$g(s) = \sum_{j:(n,j)\in\Theta} \Pr[T^{\mathrm{C}} = T^{\mathrm{C}}_j] e^{-T^{\mathrm{C}}_j s}. \tag{19}$$

For the queuing time w^{C}, the Laplace-Stieltjes transform of its probability density function is given by the Pollaczek-Khinchine transform [27] as

$$W^*(s) = \frac{(1 - \lambda^{\mathrm{ES}}\bar{T}^{\mathrm{C}})s}{s - \lambda^{\mathrm{ES}}(1 - g(s))}, \tag{20}$$

where $\bar{T}^{\mathrm{C}}$ is the mean of T^{C}. The distribution of w^{C} can be obtained by numerical inversion of (20).

D. Distribution of $t^{\mathrm{OFF}}_{n,j}$

Given the distribution of $t^{\mathrm{W}}_{n,j}$ and w^{C}, we can find the distribution of $t^{\mathrm{OFF}}_{n,j} = t^{\mathrm{W}}_{n,j} + w^{\mathrm{C}} + T^{\mathrm{C}}_j$. Note that $t^{\mathrm{W}}_{n,j}$ and w^{C} are independent since the wireless channel uploading time is independent of the ES service time.

In the soft deadline case, the joint probability distribution of total offloading delay $t^{\mathrm{OFF}}_{n,j}$ is

$$\begin{aligned}\Pr[t^{\mathrm{OFF}}_{n,j} \le d_j] &= \Pr[t^{\mathrm{W}}_{n,j} + w^{\mathrm{C}} + T^{\mathrm{C}}_j \le d_j] \\ &= \Pr[t^{\mathrm{W}}_{n,j} + w^{\mathrm{C}} \le d_j - T^{\mathrm{C}}_j] \\ &= \sum_{l=1}^{\lfloor (d_j - T^{\mathrm{C}}_j)/\tau \rfloor} \Pr[t^{\mathrm{W}}_{n,j} = l] \Pr[w^{\mathrm{C}} \le d_j - T^{\mathrm{C}}_j - l\tau].\end{aligned} \tag{21}$$

In the hard deadline case, in order to calculate the power consumption of MDs based on CLE, time-related variables are discretized into multiples of τ with $\tilde{a}$ be the discretized version of a. For d_j and T^{C}_j, their discretized versions are given as $\tilde{d}_j = \lfloor d_j/\tau \rfloor$ and $\tilde{T}^{\mathrm{C}}_j = \left\lceil \frac{T^{\mathrm{C}}_j}{\tau} \right\rceil$. Note that the discretization of d_j takes the floor of d_j/τ while that of T^{C}_j takes the ceiling of T^{C}_j/τ in order to respect the task delay constraint. With this, the distribution of the discretized version of $t^{\mathrm{OFF}}_{n,j}$ is

$$\Pr\left[\tilde{t}^{\mathrm{OFF}}_{n,j} = t\right] = \sum_{l=1}^{t - \tilde{T}^{\mathrm{C}}_j} \Pr[t^{\mathrm{W}}_{n,j} = l] \Pr[w^{\mathrm{C}} = t - \tilde{T}^{\mathrm{C}}_j - l]. \tag{22}$$

E. Computing $E^{\mathrm{O}}_{n,j}$ and $E^{\mathrm{B}}_{n,j}$

With the discretization of time, the integrals in (8) and (9) become summations as follows:

$$E^{\mathrm{O}}_{n,j} = (1 - P^{\mathrm{B}}_n)\lambda_{n,j}\ p^{\mathrm{L}} \sum_{t=\tilde{t}^{\mathrm{CLE}}_j}^{\tilde{d}_j} (t - \tilde{t}^{\mathrm{CLE}}_j + 1) \Pr\left[\tilde{t}^{\mathrm{OFF}}_{n,j} = t\right], \tag{23}$$

$$E^{\mathrm{B}}_{n,j} = (1 - P^{\mathrm{B}}_n)\lambda_{n,j}\ p^{\mathrm{L}}\ T^{\mathrm{L}}_j \sum_{t=\tilde{d}_j+1}^{\infty} \Pr\left[\tilde{t}^{\mathrm{OFF}}_{n,j} = t\right], \tag{24}$$

where $\tilde{t}_j^{\mathrm{CLE}}$ is the discretized value of t_j^{CLE} in number of time slots, i.e., $\tilde{t}_j^{\mathrm{CLE}} = \lceil t_j^{\mathrm{CLE}}/\tau \rceil$. In (23) and (24), $\Pr\left[\tilde{t}_{n,j}^{\mathrm{OFF}} = t\right]$ is given by (22).

VII. Joint Task Class Partitioning and Network Resource Allocation

In this section we consider the JSTCP problem, i.e., solving TCP in a system similar to [29]. Instead of having a fixed number of channels from each BS and a fixed amount of computing resources at the ES, the considered network can lease channels from different BSs and computing resources from the ES, provided the total cost is within a predefined budget $B_{\max}$. Note that this budget is normalized to the monetary cost for a given time period, such as a day or a month. There are up to M_n channels at BS n, that can be selected. The cost of renting a channel from BS n is α_n. In order to use the computing resources at the ES, CPU resources must also be leased at the ES. The cost (based on the number of CPU cycles per second) for leasing on the CPU resource is denoted by β. The fraction of available CPU speed for rental is $y \in [0,1]$, i.e., the CPU speed available is yf^{C}. Both x_n's and y are normalized to their respective monetary cost for the same time period as $B_{\max}$. In this case, the following constraint should be satisfied:

$$\textstyle\sum_{n=1}^{N} \alpha_n x_n + \beta y f^{\mathrm{C}} \leq B^{\max}. \tag{25}$$

In this system, $\mathbf{x} = [x_1, x_2, \ldots, x_N]$ and y are decision variables, P_n^{B}, w^{C}, and $t_{n,j}^{\mathrm{OFF}}$ all become functions of $\mathbf{x}$, y, and the task class partitioning, which complicates the calculations and the problem becomes joint task class partitioning and resource allocations.

Algorithms PDC-S and PDC-H require the computation of values $\hat{\lambda}_{n,j}^{\mathrm{ES}}$, given by (10). We make the simplifying assumption that the task distributions in each BS are all the same, i.e., the probability that a task arriving at a BS belongs to a certain class task $j \in \mathcal{J}$ is the same for all BSs. We also assume that this probability is known, e.g., by observing the past history of offloading requests.

Under these assumptions, criterion (10) is simplified as follows. We first calculate the values $\hat{\lambda}_{n,j}^{\mathrm{ES}}$ as in (10), and then define

$$\hat{\lambda}_j^{\mathrm{ES}} = \min_n \hat{\lambda}_{n,j}^{\mathrm{ES}}, \tag{26}$$

for all $j \in \mathcal{J}$, as the values used for ordering the task classes. In addition, the criteria used in the HP algorithms will be changed from $\Phi_{n,j}^{\mathrm{E}}$ and $\Phi_{n,j}^{\mathrm{D}}$ to Φ_j^{E} and Φ_j^{D}, respectively. As a result, in what follows we will be considering $\mathcal{J}$ instead of $\mathcal{N} \times \mathcal{J}$, and $\Theta \subseteq \mathcal{J}$.

A. Soft Deadlines

For the soft task deadline case, we rewrite the mean power consumption of the MDs in (1) as

$$E^{\mathrm{MD}} = E_{\Theta}^{\mathrm{MD}} + E_{\mathcal{J}\setminus\Theta}^{\mathrm{DL}}, \tag{27}$$

where E_{Θ}^{MD} is the MD power consumption for the tasks belonging to classes in Θ, i.e., $E_{\Theta}^{\mathrm{MD}} = E^{\mathrm{L}} + E^{\mathrm{T}}$, and $E_{\mathcal{J}\setminus\Theta}^{\mathrm{DL}}$ is the local processing power consumption of the MDs for the tasks belonging to classes in $\mathcal{J} \setminus \Theta$, given by

$$E_{\mathcal{J}\setminus\Theta}^{\mathrm{DL}} = \sum_{n=1}^{N} \sum_{j \in \mathcal{J}\setminus\Theta} \lambda_{n,j} T_j^{\mathrm{L}} p^{\mathrm{L}}. \tag{28}$$

The general formulation of the joint class partitioning and resource allocation problem with soft deadlines is the following:

$$\min_{\Theta,\mathbf{x},y} E_{\Theta}^{\mathrm{MD}} + E_{\mathcal{J}\setminus\Theta}^{\mathrm{DL}} \text{ s.t.} \tag{29}$$

$$\sum_{n=1}^{N} \alpha_n x_n + \beta f^{\mathrm{C}} y \leq B^{\max} \tag{30}$$

$$\Pr[t_{n,j}^{\mathrm{OFF}} \leq d_j] \geq 1 - \varepsilon_j, \forall n, j \tag{31}$$

$$x_n \in \{0, 1, \ldots, M_n\}, \forall n = 1, 2, \ldots, N \tag{32}$$

$$0 \leq y \leq 1 \tag{33}$$

$$\Theta \subseteq \mathcal{J} \tag{34}$$

Note that $\mathbf{x}$ and y do not affect $E_{\mathcal{J}\setminus\Theta}^{\mathrm{DL}}$. In general, problem (29)-(34) is computationally hard to solve exactly over all possible values of $\mathbf{x}, y, \Theta$. Instead, we would like to obtain an approximate solution, based on limiting Θ to the $J+1$ subsets for a single task classes ordering.

Different y values may result in different orderings of task classes. Therefore, Algorithm PDC-S (Algorithm 1) and Algorithm HP (Algorithm 3) need to be incorporated into the approximation algorithm proposed in [29] for computing a good solution $\mathbf{x}, y$. More specifically, the algorithm of [29] discretizes variable $y \in [0,1]$ by breaking $[0,1]$ into equal segments. For each y, we use Algorithm PDC-S or Algorithm HP to obtain an ordered task class list. For each such Θ, line 7 of Algorithm PDC-S (Algorithm 1) and line 18 of Algorithm HP (Algorithm 3) become $E^{\mathrm{MD}} = E_{\mathcal{J}\setminus\Theta}^{\mathrm{DL}} + E_{\Theta}^{\mathrm{MD}*}$, where $E_{\Theta}^{\mathrm{MD}*}$ is the minimum mean power consumption of all MDs for task classes in Θ. The algorithm of [29] is used to find an $\mathbf{x}$ close to the minimum power consumption $E_{\Theta}^{\mathrm{MD}*}$, i.e., for every y and every Θ of the ordering induced by y, a good $\mathbf{x}$ is computed. The details are briefly summarized as follows: The original problem (29)-(34) can be relaxed, by relaxing variables $0 \leq x_n \leq M_n$. With y and Θ fixed, the original relaxation is transformed into a convex program, by noticing that $\Pr[t_{n,j}^{\mathrm{OFF}} \leq d_j]$ is a monotonically decreasing function of the aggregate mean task arrival rate λ^{ES}. Therefore, binary search in the range $[0, yf^{\mathrm{C}}/\bar{q}]$, where $\bar{q}$ is the average computation load of a task that can be easily computed, can be used to approximately calculate the maximum possible value λ^* that satisfies constraints (31) for all n, j. Using (16), constraints (31) can be replaced by constraint

$$\sum_{(n,j)\in\Theta} \lambda_{n,j}(1 - P_n^{\mathrm{B}}) \leq \lambda^*. \tag{35}$$

Next, we note that the blocking probability P_n^{B} is monotonically decreasing in x_n. Let $P_{n,\min}^{\mathrm{B}}$ be the blocking probability when $x_n = M_n$, then the relaxation constraints $0 \leq x_n \leq M_n$ can be replaced by the equivalent constraints

$$P_{n,\min}^{\mathrm{B}} \leq P_n^{\mathrm{B}} \leq 1, \ \forall n \in \mathcal{N}. \tag{36}$$

Finally, x_n in constraint (30) can be replaced with any convex upper bound $F(P_n^{\mathrm{B}})$ under the assumptions of Section VI, e.g., the one in [32]: $x_n \leq (\sum_{j:(n,j)\in\Theta} \lambda_{n,j})\bar{t}_n^{\mathrm{W}}(1-P_n^{\mathrm{B}}) + 1/P_n^{\mathrm{B}},\ \forall n$. After solving the new convex optimization problem on variables P_n^{B} that approximates the original one when y and Θ are fixed, and obtaining the P_n^{B}'s, we can compute the largest integral x_n^* which achieves a blocking probability equal to or bigger than P_n^{B}, for all $n \in \mathcal{N}$. In the end, the algorithm outputs the combination $\mathbf{x}^*, y^*, \Theta^*$ that achieves the minimum mean power consumption.

B. Hard Deadlines

The mean MD power consumption in the hard deadline case can be written as in (27), and the general joint resource allocation and class partitioning problem with hard deadlines is the following:

$$\min_{\Theta,\mathbf{x},y} E^{\mathrm{MD}} = E_{\Theta}^{\mathrm{MD}} + E_{\mathcal{J}\setminus\Theta}^{\mathrm{DL}} \text{ s.t.} \tag{37}$$

$$\sum_{n=1}^{N} \alpha_n x_n + \beta f^{\mathrm{C}} y \leq B^{\max} \tag{38}$$

$$x_n \in \{0,1,\ldots,M_n\}, \forall n = 1,2,\ldots,N \tag{39}$$

$$0 \leq y \leq 1 \tag{40}$$

$$\Theta \subseteq \mathcal{J} \tag{41}$$

where $E_{\mathcal{J}\setminus\Theta}^{\mathrm{DL}}$ is the same as in (28), and $E_{\Theta}^{\mathrm{MD}} = E^{\mathrm{L}} + E^{\mathrm{T}} + E^{\mathrm{CLE}}$.

Algorithms 2 or 3, together with the solution method of [29] are used in this case in the similar way as in the soft deadline case, to produce good approximate solutions. More specifically, in addition to discretizing y we also discretize $\lambda^{\mathrm{ES}} \in [0, yf^{\mathrm{C}}/\bar{q}]$ by breaking interval $[0, yf^{\mathrm{C}}/\bar{q}]$ into Λ equal segments. For each fixed y, λ^{ES} and Θ, probability $\Pr\left[\tilde{t}_{n,j}^{\mathrm{OFF}} = t\right]$ can be calculated directly for any t. By using (36) and $F(P_n^{\mathrm{B}})$, the resulting convex program can be solved efficiently. Hence, we can obtain the optimal blocking probabilities $P_n^{\mathrm{B}*}$ and compute the largest integral x_n^* which achieves blocking probabilities no smaller than $P_n^{\mathrm{B}*}$, for all $n \in \mathcal{N}$, based on the fact that the P_n^{B}'s are decreasing functions of the x_n's. After collecting the solutions for all subproblems, we output the minimum average power consumption with $\mathbf{x}^*, y^*, \Theta^*$.

VIII. Simulation Results

In this section, we present simulation results that demonstrate the performance of our algorithms. Although our algorithms can be applied to any set of Markovian channel models, for our results we use a two-state Gilbert-Elliot channel model [33], i.e., the channel states change by following a Markov chain with two states, "Good" (G) and "Bad" (B). This model is commonly used to characterize the effects of burst noise in wireless channels, where the channel can abruptly transition between good and bad conditions [34]. Since the channel can make these abrupt transitions in quality, computation offloading decisions may be more difficult compared to cases where the channel states are more consistent as the channel offloading progresses. Let B_g and B_b be the data transmission rates when the channel is in the G and B states, respectively. It is assumed that all channels have the same B_g and B_b values but differ in their state transition probabilities, which result in different propagation models. The transition probabilities for propagation model k in BS n are denoted as $P_{n,k}^{\mathrm{GG}}, P_{n,k}^{\mathrm{GB}}$, $P_{n,k}^{\mathrm{BG}}$, and $P_{n,k}^{\mathrm{BB}}$. In each time slot, the channel state Markov chain transitions in accordance with these probabilities. Denote $\pi_{n,k}^{\mathrm{G}}$ and $\pi_{n,k}^{\mathrm{B}}$, respectively, as the stationary probabilities of being in the G and B states for a channel following propagation model k in BS n. The average wireless transmission rate of jth class tasks in BS n as

$$\bar{B}_{n,j} = \sum_{k=1}^{K_n} P_{n,j,k}^{\mathrm{G}}(\pi_{n,k}^{\mathrm{G}} B_g + \pi_{n,k}^{\mathrm{B}} B_b). \tag{42}$$

Given the channel state transition probabilities, the distribution of wireless transmission time $t_{n,j,k}^{\mathrm{W}}$ for uploading a class j task in BS n through a channel with propagation model k can be calculated from [8].

We consider a network with 3 BSs. Within each BS, there are two (Markovian) channel propagation models with transition probabilities $P_{n,1}^{\mathrm{GG}} = 0.9$, $P_{n,2}^{\mathrm{GG}} = 0.6$, $P_{n,1}^{\mathrm{BB}} = 0.1$, and $P_{n,2}^{\mathrm{BB}} = 0.4$ for $n = 1,2,3$. We consider that for given n and k, $P_{n,j,k}^{\mathrm{G}}$ values are the same for all $j \in \mathcal{J}$. That is, the probability of accessing the channels with a given propagation model in a given BS does not depend on the task class. Therefore, $P_{n,j,k}^{\mathrm{G}}$ can be reduced to $P_{n,k}^{\mathrm{G}}$ by dropping the subscript j, and their values are given as $P_{1,1}^{\mathrm{G}} = 0.8$, $P_{1,2}^{\mathrm{G}} = 0.2$, $P_{2,1}^{\mathrm{G}} = 0.5$, $P_{2,2}^{\mathrm{G}} = 0.5$, $P_{3,1}^{\mathrm{G}} = 0.2$, and $P_{3,2}^{\mathrm{G}} = 0.8$. Other default parameter values are summarized in Table I. These parameter values are similar to those used in [7], [35], [36] and [37] and varied during the simulation. We intentionally use a wide range of parameter values based on the referenced ranges so that we can make conclusions that apply in general settings. In addition, we chose different sets of parameters for the task classes in order to examine the performance of the proposed solutions.

TABLE I: Default Parameters

Parameter	Value
τ	10 ms
p^{L}	0.5 W
p^{T}	0.1 W
$\lambda_{n,j}$	0.4 tasks/s
α_n	1 /channel
β	2 /GHz
f^{L}	200M cycles/s
ε	3 %
ξ	40 %
Bg, B_b	3M, 0.5M bits per time slot

TABLE II: Task class parameters, set 1

s_j (M bits)	4	8	12	16	24	28	32	46	50	54
q_j (M CPU cycles)	54	50	46	32	28	24	16	12	8	4
d_j (ms)	300	250	250	160	200	160	250	300	500	500

TABLE III: Task class parameters, set 2

10 classes										
s_j (M bits)	2	3	6	9	30	32	35	70	75	80
q_j (M CPU cycles)	40	42	45	47	21	23	25	2	4	6
d_j (ms)	200	230	250	250	200	230	250	500	500	550

TABLE IV: Task class parameters, set 3

20 classes										
s_j (M bits)	1	2	3	5	6	8	10	31	33	35
q_j (M CPU cycles)	79	72	69	67	65	64	62	39	38	38
d_j (ms)	400	400	400	400	350	350	350	250	250	250
s_j (M bits)	35	37	39	40	61	63	65	65	76	78
q_j (M CPU cycles)	36	35	34	33	10	8	6	5	3	1
d_j (ms)	250	280	280	280	300	300	500	500	600	600

TABLE V: Task class parameters, set 4

s_j (M bits)	3	8	10	16	46	51	59	82	90	99
q_j (M CPU cycles)	86	83	85	92	41	50	56	18	2	11
d_j (ms)	440	500	520	580	320	380	430	520	700	780

TABLE VI: Task class parameters, set 5

10 classes										
s_j (M bits)	2	4	6	7	8	10	32	35	39	75
q_j (M CPU cycles)	78	74	72	68	65	60	39	35	31	2
d_j (ms)	60	80	100	60	120	120	500	500	500	1000

A. Task class ordering criteria

Before looking at performance of the proposed class partitioning algorithms, we first examine the performance of the different class ordering criteria used in these algorithms. For both PDC-S and PDC-H, the ordering of task classes are based on $\hat{\lambda}_{n,j}^{\mathrm{ES}}$'s, which are calculated based on the delay constraints. For HP, the ordering is based on both MD power consumption ($\Phi_{n,j}^{\mathrm{E}}$) and task delay constraints ($\Phi_{n,j}^{\mathrm{D}}$).

We first consider preallcoated network resources with $x_n = 15$ in all the BSs and the ES capacity $yf^{\mathrm{C}} = 25$G CPU cycles/s. There are 10 task classes and their parameter values are given in Table II. In order to reflect the heterogenous nature of the classes, we consider 3 categories of task classes. The first 4 classes have relatively small input data sizes and large computational loads, the next 3 classes have moderate input data sizes and computational loads, and the last 3 classes have relatively large input data sizes and small computational loads. For each of the class ordering methods, after the task classes have been ordered, the set Θ includes the first θ classes in the class list, and the average MD power consumption based on this partitioning is collected and shown in Fig. 3.

Next, we consider that the available amount of network resources is optimized based on task class partitioning, class completion deadlines, and resource cost budget. That is, for each of the class ordering methods, after the classes have been ordered and the first θ classes are included into the set Θ, the average MD power consumption is minimized by optimizing x_n's and y subject to a cost budget. We consider two sets of task classes with the parameter values given in Tables III and IV, respectively. For the case with 10 task classes given in Tables III, the total number of wireless channels in each BS is $M_n = 20$, the total ES capacity is $f^{\mathrm{C}} = 20$ G CPU cycles/s, and the resource cost budget is 100. For the case with 20 task classes given in Tables IV, the total number of wireless channels in each BS is $M_n = 25$, the total ES capacity is $f^{\mathrm{C}} = 40$ G CPU cycles/s, and the resource cost budget is 220. These parameter settings are similar to those in [35], [38] and [39]. The mean MD power consumption is shown in Fig. 4 for different θ values.

Both figures show that when θ is relatively small, increasing θ helps reduce the mean MD power consumption; furthermore, the power consumption decreases much faster using the hierarchical ordering than the delay-based ordering. However, as θ keeps increasing and beyond a certain value, the mean MD power consumption using both ordering criteria starts increasing, and the hierarchical ordering may result in much higher MD power consumption than the delay-based criterion. The figures demonstrate that for both class ordering criteria, allowing too many task classes to offload can increase the mean MD power consumption. This demonstrates the importance of task class partitioning. For example, Fig. 3 shows that, for the delay-based class ordering, when θ is 28 or larger, all tasks are processed locally; and for the hierarchical class ordering, when θ is 19 or larger, all tasks are processed locally. Meanwhile, the figures also show that for each class ordering criterion, there is an optimum value of θ that minimizes the mean MD power consumption; and the minimum achievable MD power consumption using the hierarchical ordering is much less than that using the delay-based ordering.

B. BSTCP

In this subsection, we examine BSTCP, i.e., the proposed task class partitioning algorithms with preallocated network resources. There are 10 task classes with their parameter values given in Table II. Fig. 5 shows the average MD power consumption versus x_n (same for all BSs) when $yf^{\mathrm{C}} = 25$ G CPU cycles/s. For comparison, we include the case of no class partitioning, i.e., all tasks can be offloaded, provided a channel is available upon the task arrival and the delay constraint can be satisfied (for SD only) [29], which is an example of the state of the art that considers the computation offloading problem without class partitioning but with both the same hard and soft deadline definitions. Without class partitioning, the average MD power consumption is a constant for the SD case, since all tasks are executed locally based on the simulated parameter setting. For the HD case and without class partitioning, when x_n increases, the average power consumption decreases first and then increases and finally becomes constant as x_n is sufficiently large.

By partitioning the task classes, both PDC and HP help significantly reduce the average MD power consumption for both SD and HD cases. For all the offloading solutions, provided offloading is possible, the mean MD power consumption decreases with the number of channels and then increases and finally keeps constant. This is because at first, increasing number of channels helps more tasks to offload, which reduces

TABLE VII: Task class parameters, set 6

15 classes								
s_j (M bits)	2	3	6	8	10	31	33	35
q_j (M CPU cycles)	72	69	65	64	62	39	38	38
d_j (ms)	60	80	40	120	120	200	280	280
s_j (M bits)	37	40	62	63	65	76	77	
q_j (M CPU cycles)	35	33	10	10	8	2	1	
d_j (ms)	280	280	400	400	1000	1000	1000	

TABLE VIII: Task class parameters, set 7

20 classes										
s_j (M bits)	2	4	6	30	31	32	33	33	34	35
q_j (M CPU cycles)	80	70	60	40	39	38	38	37	36	35
d_j (ms)	60	60	80	250	500	500	500	500	500	500
s_j (M bits)	35	36	37	38	38	39	40	60	70	80
q_j (M CPU cycles)	35	34	33	33	32	31	30	8	5	2
d_j (ms)	500	500	500	250	500	500	500	500	1000	1200

the MD power consumption; however too many offloaded tasks increases the queueing delay at the ES server, which increases the MD power consumption due to CLE (for HD) or results in more local executions (for SD). However, since the task arrival rates are fixed in the simulation, the maximum amount of traffic load at the ES stops increasing when x_n is sufficiently large, and therefore, the MD power consumption stops increasing with x_n.

Fig. 6 shows the average MD power consumption versus y, where $x_n = 25$ for all n. Both PDC and HP help decrease the MD power consumption, and HP achieves lower power consumption than PDC. When y is relatively small, the average MD power consumption using both PDC and HP decreases with y; and y is sufficiently large, the average MD power consumption becomes constant for both PDC and HP. This is because as y is sufficiently large, the queueing delay at the ES becomes almost zero, and further increasing y does not change the offloading performance. However, the y values at which the power consumption stops decreasing for PDC is much smaller than that for HP. This is an indication that HP allows more tasks to be offloaded, and therefore, achieves lower mean MD power consumption than PDC.

Figs. 5 and 6 together indicate that for a given y value, there is an optimum number of channels to minimize the MD power consumption; and for given x_n, there is a minimum value of y that achieves the minimum MD power consumption. Next, we examine the performance of joint task class partitioning and network resource allocations.

C. JSTCP

We consider two sets of task classes with the parameter values given in Tables III and IV, respectively. Default parameters used in this subsection are as follows. For the case with 10 task classes given in Tables III, the total number of wireless channels in each BS is $M_n = 20$, the total ES capacity is $f^{\mathrm{C}} = 20$ G CPU cycles/s, and the resource cost budget is 100. For the case with 20 task classes given in Tables IV, the total number of wireless channels in each BS is $M_n = 25$, the total ES capacity is $f^{\mathrm{C}} = 40$ G CPU cycles/s, and the resource cost budget is 220.

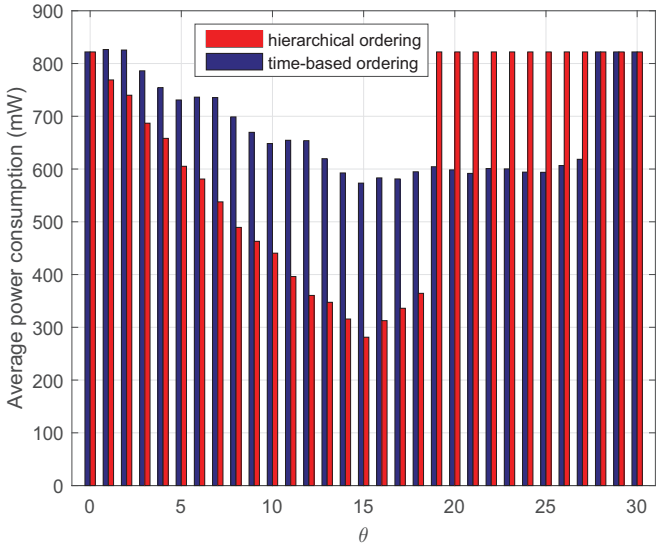
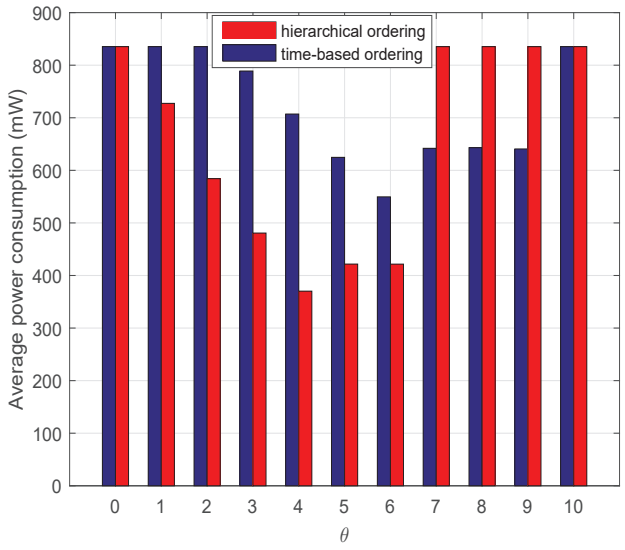


Fig. 3: Average power consumption versus θ

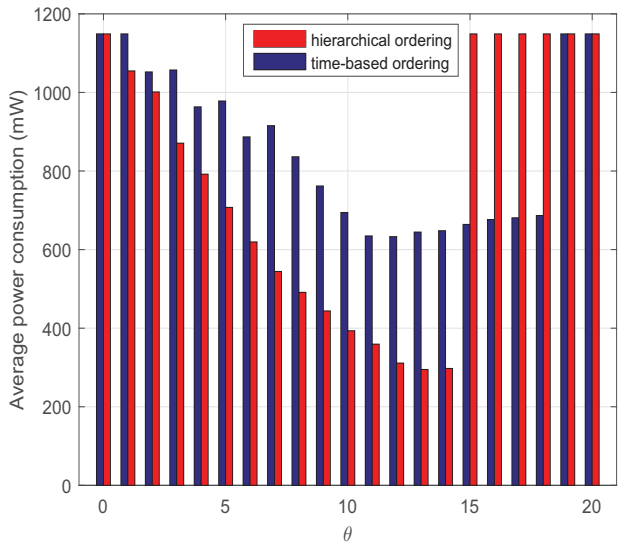
Fig. 7 shows the average MD power consumption versus $B^{\max}$, which shows that the proposed joint class partitioning and resource allocation solutions can greatly reduce the average MD power consumption, compared to the no class partitioning achieved in [29]. When the cost budget is relatively small, the power reduction is small due to the limited amount of resources that limits the number of offloaded tasks. Note that all the tasks are executed locally for the SD case when there is no class partitioning. For all the other offloading cases, the cost budget increases, the average MD power consumption reduces, while using HP results in much lower average MD power consumption. When the cost budget is sufficiently high that results in negligible channel blocking probability and ES queueing delay, further increasing the cost budget does not help reduce the power consumption.

Fig. 8 shows the average MD power consumption versus ES capacity, f^{C}. It is seen that when the ES capacity is relatively low, the difference of average MD power consumption between partitioning and no partitioning [29] and between PDC and HP is small. The small amount of computing resource limits the number of offloaded tasks (without class partitioning) or task classes (with class partitioning), and as a result, most tasks are executed locally. As the ES capacity increases, the joint class partitioning and resource allocation solutions result in much lower average MD power consumption than the solutions without class partitioning. When the ES capacity is larger than a certain value, further increasing it will not reduce the MD power consumption since the offloading performance is limited by other factors such as cost budget. By comparing PDC and HP, it is seen that HP can take better advantage of the ES capacity to reduce the MD power consumption.

Next, we use a different set of the task class parameters as given in Table V. We also add another set of comparisons with the optimum resource allocation obtained by exhaustive search for the no class partitioning case [29]. In this *no partitioning-OPT* case, the optimum resource allocation is obtained by exhaustive search without class partitioning. Figs. 9 and 10

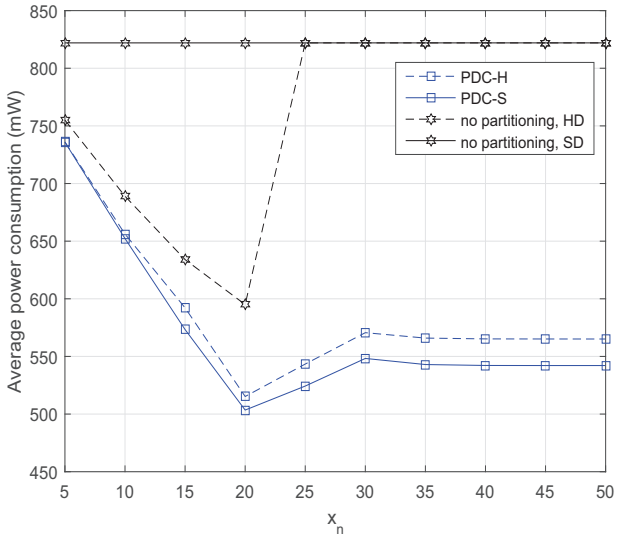


(a) 10 classes

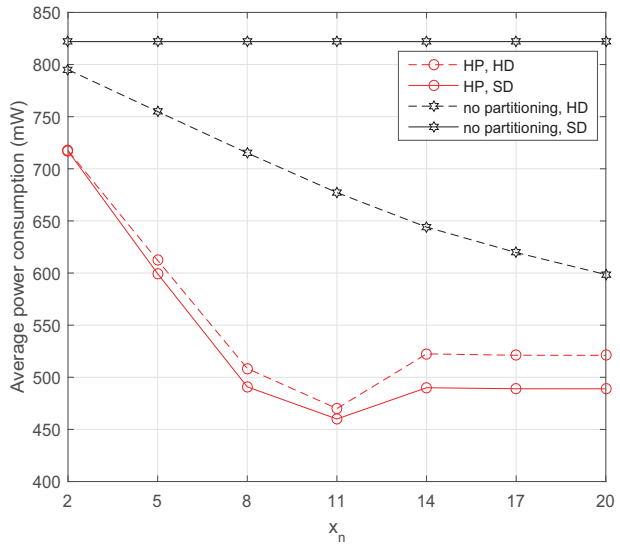


(b) 20 classes

Fig. 4: Average power consumption versus θ



(a) PDC



(b) HP

Fig. 5: Average power consumption versus x_n

show the average MD power consumption. In Fig. 9, the ES capacity is $f^{\mathrm{C}} = 30$ G CPU cycles/s; and in Fig. 10, the cost budget is $B^{\max} = 100$. In both figures, the number of available wireless channels in each BS is $M_n = 20$. Different from the results shown in Figs. 7 and 8, the new task class parameters allow some tasks to be offloaded even for the SD case without class partitioning. However, the observations are consistent with before in terms of the effect of using the proposed joint class partitioning and resource allocation solutions on the average MD power consumption. In addition, our proposed algorithms can achieve significantly lower power consumption for the mobile devices than the optimal power consumption without class partitioning. This further demonstrates the importance of task class partitioning even when it is only static, and verifies the good performance of our proposed algorithms.

Finally, we consider the effect of parameter ξ in Algorithm 3. We consider three sets of task class parameters as given in Tables VI, VII, and VIII. Other parameters used in the simulation include $x_n = 20$ for all BSs and ES capacity $yf^{\mathrm{C}} = 40$ G CPU cycles per second. The results are shown in Fig. 11. For each set of the task classes, the average MD power consumption is higher when ξ is 0 or 1. In another word, in order to reduce the average MD power consumption, the classes should not be ordered by $\Phi^{\mathrm{E}}_{n,j}$ only ($\xi = 0$) or $\Phi^{\mathrm{D}}_{n,j}$ only ($\xi = 1$). Although finding the optimum value of ξ that minimizes the average MD power consumption is not straightforward, we did notice that for each set of task class parameters, there is a relatively large range of ξ values during which the average MD power consumption is the minimum.

IX. Conclusions

This paper has introduced algorithms for static task class partitioning in MCO. The algorithms are given a set of task

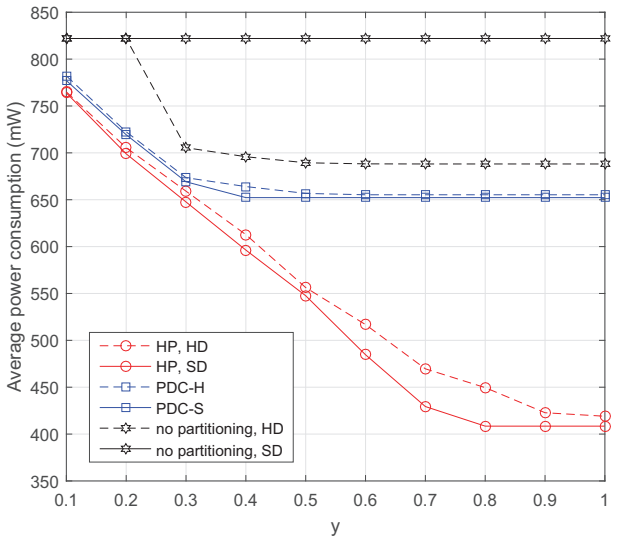
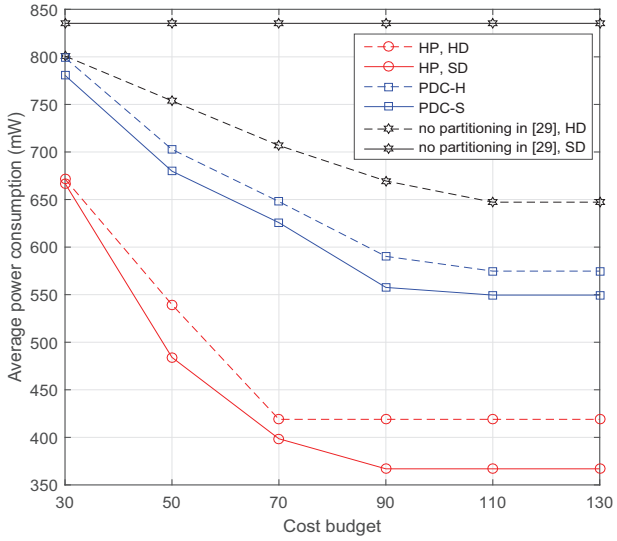
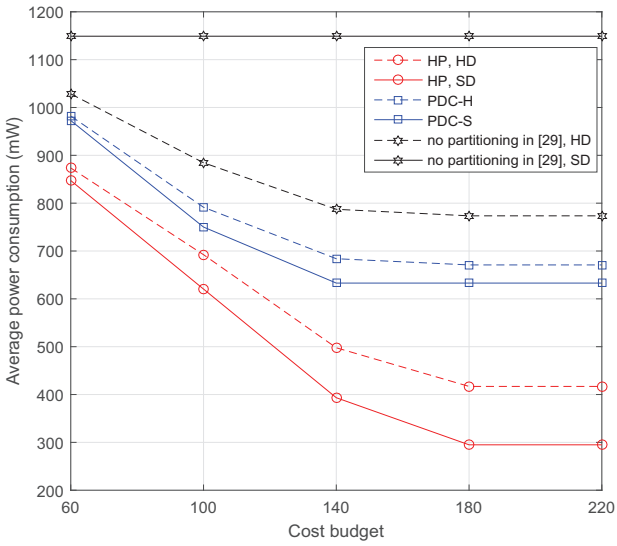


Fig. 6: Average power consumption versus y



(a) 10 classes



(b) 20 classes

Fig. 7: Average power consumption versus cost budget

classes that must be partitioned into two sets, each containing task classes that are either executed locally by the MD or those classes that can contend for remote ES execution. The objective is to find the task class partition that gives the minimum mean mobile device power consumption subject to task completion time constraints. Algorithms for both soft and hard task completion time deadlines were presented. The algorithms consider two different variations of the problem. In the first, the wireless and computational resource capacities are given beforehand and the algorithms determine the task class partitioning. In the second variation, the resource capacities are not given, and the algorithms generate both capacity assignments and the partitioning subject to a resource cost budget constraint. Two basic algorithms were introduced, the first one uses latency based task class ordering, and the second orders the task classes in an hierarchical way by first grouping task classes with similar mean power consumption and then ordering the classes within the same group based on task completion time. A variety of simulation results were presented that demonstrate the excellent performance of the proposed class partitioning algorithms for both given and optimized network resource assignments.

As mentioned above, an obvious future research direction is the study of dynamic TCP and the development of online task class partitioning algorithms. In addition to ideas presented in this work, machine learning and deep learning techniques could also be used, in response to quickly changing environments.

Another future research direction already mentioned is the combination of TCP with partial task offloading. This adds the extra complication of dividing application tasks into sub-tasks and the assignment of subtasks to a set of sub-classes that will have to be partitioned according to TCP, possibly with the requirement that subtask precedence constraints are also satisfied.

REFERENCES

[1] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo, "Computation offloading in mobile edge computing networks: A survey," *J. of Network and Computer Applications*, vol. 202, p. 103366, 2022.

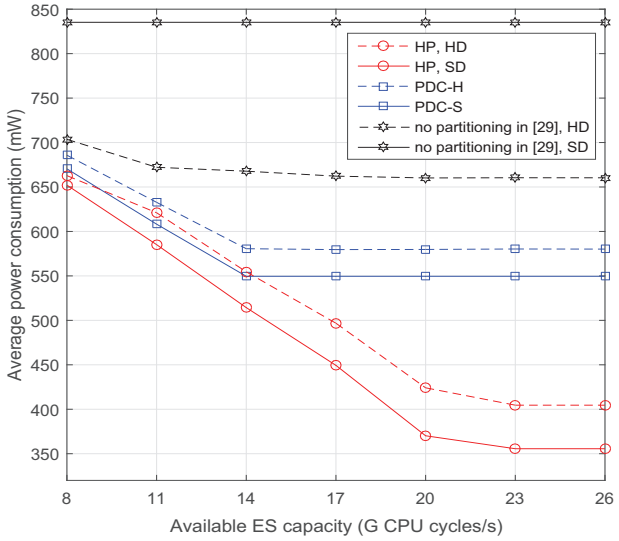
[2] Huawei Inc., "5G network architecture - a high-level perspective," https://www.huawei.com/en/technology-insights/industry-insights/outlook/mobile-broadband/insights-reports/5g-network-architecture, 2016.

[3] C. Tang, C. Zhu, N. Zhang, M. Guizani, and J. J. P. C. Rodrigues, "SDN-assisted mobile edge computing for collaborative computation offloading in industrial internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 253–24 263, 2022.

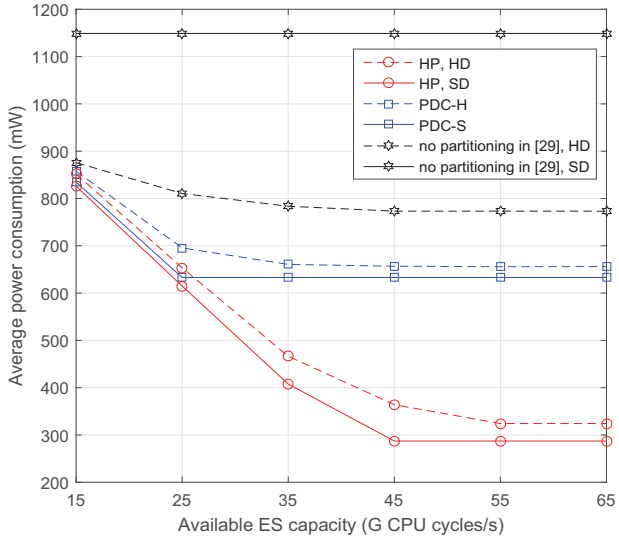
[4] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. Shen, "Energy efficient dynamic offloading in mobile edge computing for internet of things," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 1050–1060, 2021.

[5] O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, 2015.

[6] S. Nath and J. Wu, "Dynamic computation offloading and resource



(a) 10 classes



(b) 20 classes

Fig. 8: Average power consumption versus ES capacity

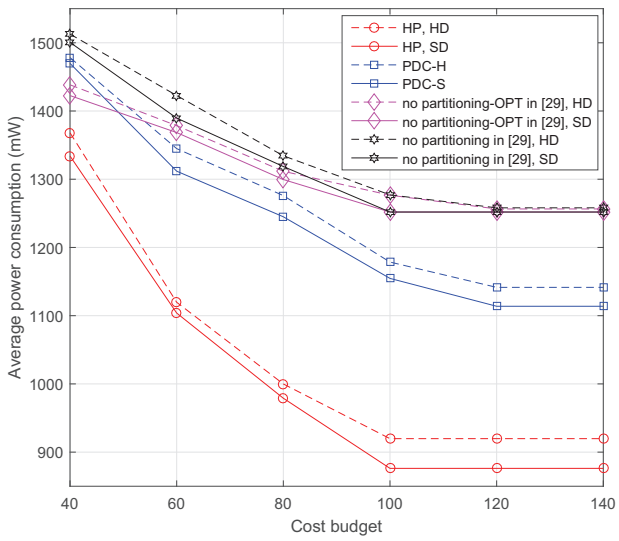


Fig. 9: Average power consumption versus cost budget

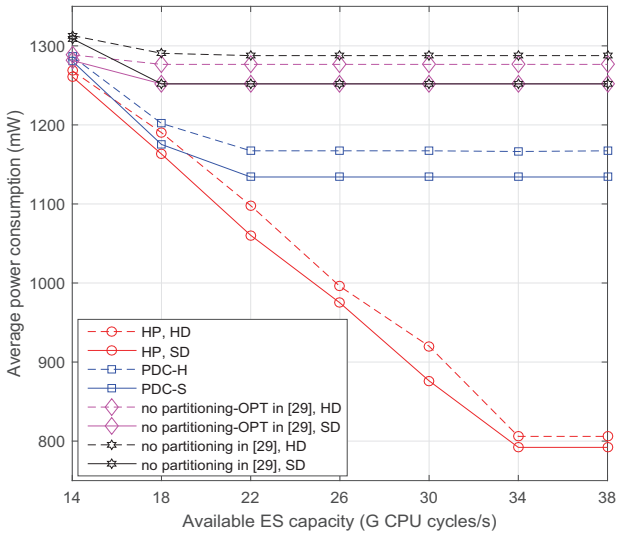


Fig. 10: Average power consumption versus ES capacity

allocation for multi-user mobile edge computing," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[7] S. Yue, J. Ren, N. Qiao, Y. Zhang, H. Jiang, Y. Zhang, and Y. Yang, "TODG: Distributed task offloading with delay guarantees for edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 7, pp. 1650–1665, 2022.

[8] A. Hekmati, P. Teymoori, T. D. Todd, D. Zhao, and G. Karakostas, "Optimal mobile computation offloading with hard deadline constraints," *IEEE Transactions on Mobile Computing*, vol. 19, no. 9, pp. 2160–2173, 2020.

[9] H. Chen, D. Zhao, Q. Chen, and R. Chai, "Joint computation offloading and radio resource allocations in small-cell wireless cellular networks," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 3, pp. 745–758, 2020.

[10] Y. Deng, Z. Chen, and X. Chen, "Resource allocation for multi-user mobile-edge computing systems with delay constraints," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[11] C. W. Zaw, N. H. Tran, Z. Han, and C. S. Hong, "Radio and computing resource allocation in co-located edge computing: A generalized nash equilibrium model," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[12] H. Li, H. Xu, C. Zhou, X. Lü, and Z. Han, "Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10 214–10 226, 2020.

[13] J. Ren and S. Xu, "DDPG based computation offloading and resource allocation for MEC systems with energy harvesting," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021, pp. 1–5.

[14] L. Tan, Z. Kuang, L. Zhao, and A. Liu, "Energy-efficient joint task offloading and resource allocation in OFDMA-based collaborative edge computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 3, pp. 1960–1972, 2022.

[15] H. Wu, Y. Sun, and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," *IEEE Transactions on Cloud Computing*, vol. 8, no. 2, pp. 570–584, 2020.

[16] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 2, pp. 461–474, 2018.

[17] G. Qu, J. Zhou, Z. Sheng, H. Yu, and Y. Ren, "Reliability-guaranteed admission control for mobile computation offloading under nakagami fading channel," *IEEE Wireless Communications Letters*, vol. 10, no. 10, pp. 2195–2199, 2021.

[18] Q. Li, S. Wang, A. Zhou, X. Ma, F. Yang, and A. X. Liu, "QoS driven task offloading with statistical guarantee in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 278–290, 2022.

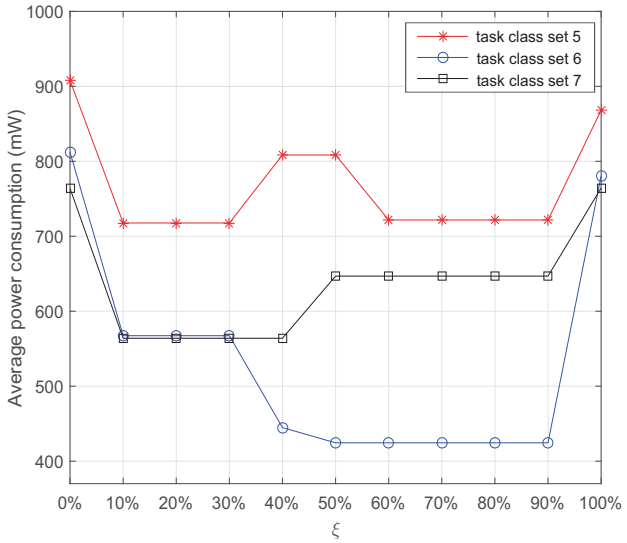


Fig. 11: Average power consumption versus ξ in HP

[19] L. Zeng, W. Wen, and C. Dong, "QoS-aware task offloading with NOMA-based resource allocation for mobile edge computing," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 1242–1247.

[20] T. Fang, F. Yuan, L. Ao, and J. Chen, "Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: A potential game approach," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3226–3237, 2022.

[21] S. Mu, Z. Zhong, and D. Zhao, "Energy-efficient and delay-fair mobile computation offloading," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15 746–15 759, 2020.

[22] Y. Wu, K. Ni, C. Zhang, L. P. Qian, and D. H. K. Tsang, "NOMA-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12 244–12 258, 2018.

[23] Z. Wang, D. Zhao, M. Ni, L. Li, and C. Li, "Collaborative mobile computation offloading to vehicle-based cloudlets," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 768–781, 2021.

[24] Z. Wang, Z. Zhong, D. Zhao, and M. Ni, "Vehicle-based cloudlet relaying for mobile computation offloading," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 11 181–11 191, 2018.

[25] S. Mu, Z. Zhong, D. Zhao, and M. Ni, "Joint job partitioning and collaborative computation offloading for internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1046–1059, 2019.

[26] H. Wu, W. J. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1464–1480, 2019.

[27] A. Y. Khintchine, "Mathematical theory of a stationary queue," *Matematicheskii Sbornik*, vol. 39, no. 4, pp. 73–84, 1932.

[28] R. W. Wolff, "Poisson arrivals see time averages," *Operations Research*, vol. 30, no. 2, pp. 223–414, 1982.

[29] H. Chen, T. D. Todd, D. Zhao, and G. Karakostas, "Wireless and service allocation for mobile computation offloading with task deadlines," *arXiv preprint arXiv:2301.12088*, 2023.

[30] D. Y. Burman, "Insensitivity in queueing systems," *Advances in Applied Probability*, vol. 13, no. 4, pp. 846–859, 1981.

[31] D. N. Shanbhag and D. G. Tambouratzis, "Erlang's formula and some results on the departure process for a loss system," *Journal of Applied Probability*, vol. 10, no. 1, pp. 233–240, 1973.

[32] S. A. Berezner, A. E. Krzesinski, and P. G. Taylor, "On the inverse of erlang's function," *Journal of Applied Probability*, vol. 35, no. 1, pp. 246–252, 1998.

[33] E. N. Gilbert, "Capacity of a burst-noise channel," *The Bell System Technical Journal*, vol. 39, no. 5, pp. 1253–1265, 1960.

[34] T. Blazek and C. F. Mecklenbräuker, "Measurement-based burst-error performance modeling for cooperative intelligent transport systems," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–10, 2018.

[35] X.-Q. Pham, T.-D. Nguyen, V. Nguyen, and E.-N. Huh, "Joint service caching and task offloading in multi-access edge computing: A QoE-based utility optimization approach," *IEEE Communications Letters*, vol. 25, no. 3, pp. 965–969, 2021.

[36] Y. Li, B. Yang, H. Wu, Q. Han, C. Chen, and X. Guan, "Joint offloading decision and resource allocation for vehicular fog-edge computing networks: A contract-stackelberg approach," *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15 969–15 982, 2022.

[37] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668–682, 2019.

[38] D. Liu, A. Hafid, and L. Khoukhi, "Multi-item auction based mechanism for mobile data offloading: A robust optimization approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4155–4168, 2020.

[39] E. Šlapak, J. Gazda, W. Guo, T. Maksymyuk, and M. Dohler, "Cost-effective resource allocation for multitier mobile edge computing in 5G mobile networks," *IEEE Access*, vol. 9, pp. 28 658–28 672, 2021.