Multi-Armed Bandit for Energy-Efficient and Delay-Sensitive Edge Computing in Dynamic Networks with Uncertainty

Saeed Ghoorchian and Setareh Maghsudi

Abstract

In the emerging edge-computing paradigm, mobile devices offload the computational tasks to an edge server by routing the required data over the wireless network. The full potential of edge-computing becomes realized only if the devices select the most appropriate server in terms of the latency and energy consumption, among many available ones. This problem is, however, challenging due to the randomness of the environment and lack of prior information about the environment. Therefore, a smart device, which sequentially chooses a server under uncertainty, attempts to improve its decision based on the historical time- and energy consumption. The problem becomes more complicated in a dynamic environment, where key variables might undergo abrupt changes. To deal with the aforementioned problem, we first analyze the required time and energy to data transmission and processing. We then use the analysis to cast the problem as a budget-constrained multi-armed bandit problem, where each arm is associated with a reward and cost, with time-variant statistical characteristics. We propose a policy to solve the formulated bandit problem and prove a regret bound. The numerical results demonstrate the superiority of the proposed method compared to a number of existing solutions.

Keywords: Computation offloading, edge computing, multi-armed bandits, uncertainty.

I. Introduction

The popularity of mobile applications has significantly increased among users over the past years. Some apps, for example, those based on face and/or voice recognition, produce an excessive amount of data and require heavy computations. Even if a hand-held device is capable of performing the computations using its own internal hardware, local data...
processing and computation usually yield long delay as well as excessive power consumption, thereby resulting in a low Quality of Service (QoS). Moreover, in a long run, repetitive local computation might affect the lifetime of the battery or other components of a mobile device.

Edge computing is an emerging field of research with various applications in the area of Internet of Things (IoT) [1]. In the next-generation wireless networks, edge servers (for example, small base stations) are foreseen to offer computational services, meaning that the devices have the possibility to offload their computational data through a wireless network to the edge servers so that the data is processed remotely. Compared to the cloud servers [2], edge servers are located at close proximity to the users, which guarantees a shorter data transmission time and thereby a lower energy consumption [3], [4]. Needless to say, edge computing becomes more efficient if the offloading devices are autonomous, i.e., able to choose when and to which server to offload and which resources to use. Implementing an autonomous behavior is, however, not a trivial task. One reason is that unlike cloud servers, there might be multiple edge servers available to the user’s device at the time of offloading. Moreover, often the users’ devices are not given any prior information about the servers and network. In addition, the environment might be dynamic and time-variant, i.e., some statistical characteristics of the network and servers might change over time.

To deal with the aforementioned challenge, an autonomous device interacts with the network, by sequentially choosing a server under uncertainty, and gathers some information about the environment in each offloading round. The goal is to improve the decisions for the next offloading rounds based on the previously consumed time and energy. This problem is an instant of online decision-making, where the decisions are taken sequentially based on the historical observations to optimize some objective function.

Multi-armed bandit (MAB) problem is a subclass of online decision-making problems which involves a gambling machine with several arms and a gambler [5], [6]. The random reward generating processes of arms are a priori unknown. At each of the consecutive rounds, the gambler pulls down one arm and receives a reward. The gambler attempts to maximize its accumulated reward over a finite time horizon by selecting the best arm in terms of utility. Alternatively, the gambler would try to minimize its accumulated regret, which is defined as the difference between the reward that could be achieved if the gambler played the optimal arm and the one that it has indeed received in each round by following the applied policy [7]. With its various settings [8], MAB is capable of modeling a variety of
real-world problems. In this paper, we use an MAB formulation to deal with the optimal server selection in the edge computing paradigm.

A. Related Works

Similar to any other networking paradigm, resource management is a key challenge in computation offloading due to the scarcity of resources such as the computational power, environmental and hardware constraints such as the number of available servers, and the dynamic status of the network or servers such as the task arrival rate. In [9], the authors take advantage of supervised learning to solve a computation offloading problem in a dynamic environment, where a single user decides which components of the application to execute locally and which ones to offload. They jointly optimize the local execution cost and the offloading cost using a deep neural network framework. In [10], the authors study the CPU task allocation problem by formulating optimization problems based on the execution time and energy consumption. They consider two cases for the mobile device, namely fixed CPU frequency and elastic CPU frequency, and solve the proposed optimization problems using different approximation approaches. In [11], authors formulate a non-convex optimization problem to optimize both the latency and reliability (offloading failure probability) in computation offloading of a single user. They design three algorithms to optimize edge node candidate selection, offloading ordering, and task allocation. In [12], the authors investigate the partial offloading of some components of the computational task of a single device. They propose an algorithm which uses a Lyapunov optimization with a given time delay constraint to reduce the energy consumption. Similarly, [13] studies a partial computation offloading of a single user where multiple antennas are available at the mobile terminal and the femto-access point. The authors propose a numerical optimization technique to optimize latency and energy consumption. Further, in [14], the authors consider the partial offloading problem. They assume the availability of a small cell cloud manager, which determines whether to offload or not and which portion is needed to be offloaded. They propose different algorithms to separately optimize the latency and consumed energy. [15] considers the offloading problem of a single user in a multi-cloudlet environment and proposes an application-specific cloudlet selection strategy, where different cloudlets are able to execute different application types, to optimize the execution latency and energy consumption of the device. In [16], the authors consider an ad-hoc mobile network and formulate the
partial offloading problem of a single user as a Markov decision process. They use a deep reinforcement learning algorithm to solve the formulated problem. The goal is to find the optimal number of tasks which should be locally executed in the device or offloaded to each cloudlet so that the user’s utility is maximized whereas the energy consumption, processing delay, required payment, and task loss probability are minimized.

As mentioned previously, we model the computation offloading problem in the MAB framework. Our approach is perhaps most closely related to [17], where a budget-constrained MAB problem is considered with a reward and a discrete cost which are independent and identically distributed (i.i.d) random variables. Similarly, [18] studies a budgeted MAB problem with i.i.d reward and cost variables. Nevertheless, extending the developed decision-making policies to dynamic (non i.i.d) environments is not straightforward. Our approach is also related to [19], where the authors investigate a non-stationary MAB problem. They consider a tunable window length $\tau$ and compute the empirical estimate of the mean reward of each arm using only the past $\tau$ observations. However, in their formulation, pulling arms does not result in any cost. In [20], the authors study a budget-constrained MAB problem, where each arm is associated with a reward and a cost variable. The reward generating processes of arms are piece-wise stationary and the cost of pulling each arm is fixed but may be different for different arms. In [21], the authors study a stationary MAB problem with a reward variable and a continuous cost variable. Authors in [22] consider three mechanisms to solve a stationary MAB problem, which aim at selecting the arm with the maximum value of expected reward to expected cost after a finite number of rounds.

B. Our Contribution

The contribution of this paper is two-fold, as follows: Theoretically, we investigate an MAB problem, where as a result of pulling each arm, the player receives some reward and incurs some cost. Reward and cost are independent random variables with time-variant statistical characteristics. Assuming that the player is given a finite budget, we develop a decision-making policy and prove a regret bound. Our work stands in contrast to many previous works that assume stationary arms or do not include the cost of pulling arms into account. Application-wise, we use an MAB model to solve the distributed server selection problem in edge computing in a dynamic wireless network, given no prior information. Therefore, our work extends state-of-the-art works, which are mostly centralized, and/or do not take the
network’s dynamic into account, and/or require heavy information at users or servers.

In summary, the novelty in this paper is as follows.

• Our work enjoys a more realistic model compared to the state-of-the-art by including the dynamic and time-varying nature, as well as the inhomogeneity of wireless networks, into account.

• We analyze the statistical characteristics of the required time for transmitting the data from a user’s device to a server, without assuming any specific routing protocol. In addition, we analyze the required time for data processing at a server.

• We define the reward and cost in terms of the required time and energy in each offloading round, respectively, and we derive the corresponding probability distributions. The results show their dependence on specific piece-wise constant system parameters.

• We propose BPRPC-SWUCB, a novel MAB algorithm, to minimize the player’s expected cumulative regret. BPRPC-SWUCB can be used to solve a variety of dynamic decision-making problems where taking actions yields non i.i.d. reward and cost.

• We analyze BPRPC-SWUCB by proving a regret upper-bound. We also compare its performance with several existing MAB-based algorithms through intensive simulation.

• The proposed model and solution do not require heavy information acquisition and do not result in excessive computational complexity.

C. Organization

Section II describes the system model. In Section III, we introduce the concept of reward and cost in the context of the computation offloading problem, and we derive the statistical characteristics of the aforementioned variables. In Section IV, we describe and theoretically analyze an MAB algorithm, named BPRPC-SWUCB. In Section VI, we present the results of numerical analysis. Section VII concludes the paper.

II. System Model

We consider a network consisting of a set of servers located at the network’s edge and a set of users that might be willing to offload their computational job to one of the edge servers. We gather the servers in the set $\mathcal{S} = \{1, \ldots, S\}$ so that any offloading user may select one of the $|\mathcal{S}| = S$ to offload its computational task. Throughout the paper, we may use device and offloading user interchangeably.
A general computation offloading procedure consists of four elements: (i) selection of the server, (ii) sending the data to the server, (iii) processing the data and accomplishing the task at the server, and (iv) sending the results back to the user’s device. We consider the time to be slotted and denote one time instance by $t$. Moreover, we use the term *round* to refer to the time period required to accomplish a computational offloading process entirely, i.e., to succeed in all of the aforementioned sections. We denote the rounds by $\theta = 1, 2, \ldots$. Note that each round $\theta$ includes some time instants $t$.

Each computational job consists of some analysis of the offloaded data. We assume that each computational job can be divided to some homogeneous tasks with respect to the time required to process each task. Without loss of generality, we assume that each device offloads the same amount of the data at each round $\theta$. If a large amount of data is to be offloaded, we model it as multiple rounds of offloading, each with the same amount of data.

As mentioned above, in order to offload a computational task, any user transfers the required data to the server. The transfer takes place via some intermediate helper nodes, which act as transmitters and receivers. This could be, for example, other devices in the network or fixedly deployed micro- or femto small base stations. At each time, every node can act either as a transmitter or as a receiver. In the following, we discuss the network’s model from the perspective of one exemplary user.

As it is conventional [23], [24], we assume that the intermediate nodes (devices, relays, small base stations, and the like), located between the source (the user’s device) and the sink (a server), are distributed according to a homogeneous Poisson Point Process (PPP). Since the servers are located at different geographical areas, the density of the aforementioned PPP varies over servers. Therefore, we use $\Lambda_s$ to show the network’s intensity between the user and each server $s \in \mathcal{S}$. Similar to [23] and [25], to take the transmission impairments of the link between every two nodes into account, we model the links by a Bernoulli random variable with success probability $p_{s,\theta}$. In other words, the transmission is successful (non-outage) with probability $p_{s,\theta}$ and fails (outage) with probability $1 - p_{s,\theta}$. Note that the outage probability depends on the server, since the outage probability is affected by factors such as shadowing, fading, and other similar variables which depend strongly on the geographical area as well as the network density. Moreover, the dependency of the outage parameter on the round (time period) $\theta$ accommodates the time-variation of the channel quality. In brief, the network between each server $s \in \mathcal{S}$ and the offloading user is modeled by a graph,
where the vertices are distributed according to a PPP with intensity \( \Lambda_s \) and there is an edge between every two vertexes with the probability \( p_{s,\theta} \). Throughout the paper, we use the terms, user’s device and source, as well as the terms, server and sink, interchangeably.

As mentioned before, in our problem, we analyze the smart decision-making of a single offloading user, when given a number of choices with respect to the server; nonetheless, it is natural that in every network, there are many of such users, each offloading some tasks to some server. To model the collective behavior of the network mathematically, we assume that the arrived jobs at a server \( s \in S \) follow a Poisson distribution with the rate \( \lambda_{s,\theta} \). The arrival rate depends on the server \( s \) and the offloading round \( \theta \), implying that on average, the intensity of the job arrival changes with respect to the servers and time.

In the following assumption, we describe the mathematical model of time-variations of the characteristics of the random variables.

**Assumption 1.** For any server \( s \in S \) the parameters \( p_{s,\theta} \) and \( \lambda_{s,\theta} \) are piece-wise constant with respect to the round \( \theta \); in other words, they remain constant unless they experience a change at some specific round(s), referred to as change point(s). For parameters \( p_{s,\theta} \) and \( \lambda_{s,\theta} \), we denote the change points by the series \( \theta^{(p,s)}_1, \theta^{(p,s)}_2, \ldots \) and \( \theta^{(\lambda,s)}_1, \theta^{(\lambda,s)}_2, \ldots \), respectively. Naturally, the change points are not necessarily identical for two aforementioned series.

Consider a random process whose instantaneous outcomes are drawn from some probability distribution with parameter \( p_{s,\theta} \) and/or \( \lambda_{s,\theta} \). Then, by the discussion above, the process is piece-wise stationery, as the distribution of the outcomes remains time-invariant over disjoint time intervals, but changes from one interval to the other.

Moreover, we assume that the transmission range of each node (including the source and any sink node \( s \)) is the same and denote it by \( R \). That is to say, a node can only transmit to the nodes inside the circle of radius \( R \) around that node.

In Figure 1, we illustrate an exemplary system model consisting of an offloading user (black disk) and four edge servers (blue squares) at some specific time \( t \). Geographically, the network is divided into four disjoint areas (4 disjoint quarters of a 200 \( \times \) 200 plane in \( \mathbb{R}^2 \)) and the nodes in each area are distributed according to a homogeneous PPP. Naturally, in the areas with higher intensity, a larger number of intermediate nodes are available. The transparent cyan circle around each server represents the corresponding job arrival rate, where a larger radius corresponds to a larger Poisson rate.
Fig. 1: An exemplary illustration of a communication network consisting of an offloading user, four computational servers, and the intermediate transmitters and receivers. The intensity of the intermediate nodes varies with respect to each server. The transparent cyan circle around each server represents its corresponding job arrival rate, where a bigger radius corresponds to a greater rate.

Table I summarizes most important system’s parameters together with a brief description.

| Parameter | Description |
|-----------|-------------|
| $p_{s,\theta}$ | Outage parameter of the network between the user and server $s$ at round $\theta$ |
| $\lambda_{s,\theta}$ | Job arrival rate to the server $s$ at the round $\theta$ |
| $\Lambda_s$ | Network’s intensity between the user and server $s$ |
| $\rho_s$ | Service rate corresponding to the server $s$ |
| $R$ | Transmission range |
| $r_s$ | Distance between the user and the server $s$ |

III. Statistical Characteristics of the System Variables

Conventionally, in wireless networks, each user has some strict constraints (or requirements) on the delay and the energy. Therefore, given multiple choices, it is natural that an offloading user aims at selecting the server that guarantees minimum delay as well as the minimum energy consumption. Selecting the best server is however not a trivial task, in particular under uncertainty, i.e., when the required information is not available at the
offloading user. The problem becomes more challenging in a dynamic environment, where
the characteristics of the network and servers vary over time.

In order to mathematically formulate the server selection problem, in the following, we
first define and analyze the reward and cost of selecting each server.

A. Reward

As mentioned earlier, in computation offloading, an important performance metric is the
total time required for an offloading round, referred to as the delay time and denoted by
d_{s,\theta}. The delay time at round $\theta$ consists of the processing time $f_{s,\theta}$ at the server and the
transmission time $g_{s,\theta}$ between the source and a sink node $s$. Therefore, at round $\theta$ we have

$$d_{s,\theta} = f_{s,\theta} + g_{s,\theta}.$$  \hspace{1cm} (1)

For the user’s quality of service (QoS) satisfaction, we require that the delay time $d_{s,\theta}$ remains
below a pre-specified threshold, namely, $\delta$. In other words, the QoS is satisfied if $d_{s,\theta} \leq \delta$, and is not satisfied otherwise. Therefore, we define the reward, gained by the offloading user
at round $\theta$ upon choosing the server $s \in S$, as

$$r_{s,\theta} = \begin{cases} 1 & d_{s,\theta} \leq \delta \\ 0 & d_{s,\theta} > \delta. \end{cases}$$  \hspace{1cm} (2)

In the rest of this section, our goal is to find the distribution of the reward $r_{s,\theta}$, which is
determined based on the distribution of the delay time $d_{s,\theta}$. Consequently, in the following,
we determine the distribution of the processing time $f_{s,\theta}$ and the transmission time $g_{s,\theta}$.

1) Processing Time

For every server, we define the service rate as the number of tasks which can be processed
by that server per unit of time. In the following, we assume that the servers are inhomogeneous in terms of service rate, meaning that each server $s \in S$ has some service rate
$\rho_s > \lambda_{s,\theta}, \forall \theta$. We use $z_{s,\theta}$ to denote the service time required by the server $s \in S$.

Moreover, to be processed, each computational job arrived at a server $s \in S$ has to wait
in a queue for some time depending on the job arrival rate. Consider a time instance $t$ inside
a round (time period) $\theta$. We denote the waiting time at time instance $t$ by $w_{s,t}$. Similarly,
we use $f_{s,t}$ and $z_{s,t}$ to denote the processing time and the service time at time instance $t$,
respectively. Thus, at server $s \in S$, the processing time at time $t$ is given by

$$f_{s,t} = z_{s,t} + w_{s,t}.$$  

We consider an $M/M/1$ queue model, by which $z_{s,t}$ and $f_{s,t}$ follow an exponential distribution with parameter $\rho_s$ and $\rho_s - \lambda_{s,t}$, respectively \[26, 27\]. By Assumption 1, the job arrival rate remains fixed at least during a specific round $\theta$. Therefore, for any time instance $t$ inside a round (time period) $\theta$, it holds $\lambda_{s,\theta} = \lambda_{s,t}$. In words, this implies that the expected value of the waiting time, and consequently of the processing time, remains constant for the entire time period of an offloading round $\theta$. Therefore, throughout the paper, we use $f_{s,\theta}$ to denote the processing time at the server for round $\theta$, regardless of the specific time instant $t$ inside the offloading round $\theta$. Moreover, we note that by Assumption 1, $\lambda_{s,\theta}$ is assumed to be piece-wise constant, which implies that $f_{s,\theta}$ follows an exponential distribution with piece-wise constant mean $\frac{1}{\rho_s - \lambda_{s,\theta}}$. Formally,

$$P(f_{s,\theta} = x) = \begin{cases} (\rho_s - \lambda_{s,\theta})e^{-(\rho_s - \lambda_{s,\theta})x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$  \hspace{1cm} (3)  

2) Transmission Time

Now, let us consider the transmission time between the source and a sink $s$. We use $g_{s,\theta}$ to denote the transmission time between the source and a sink $s$ at the round $\theta$. A path of length $N$ is an $N$-hop connection between the source and the sink node. We represent such path by a sequence $o = x_1, x_2, \ldots, x_{N+1} = s$, where $x_i$ denotes $i$-th node in the path and $x_1$ and $x_{N+1}$ stand for the source and the sink, respectively. In the following, we derive the probability distribution of the transmission time $g_{s,\theta}$ between the source and a sink $s$.

Similar to \[28\] and \[29\], we define the concept of progress. Assume a transmitter node located at $x_i$. The progress of a node $x_{i+1}$ is defined as the projection of the link between $x_i$ and $x_{i+1}$ onto the straight line connecting the node $x_i$ and the sink node $s$. Additionally, we say a progress is positive if the projection happens towards the sink node $s$ and it is negative otherwise. We define the maximum number of hops $h_{s,\text{max}}$ between the source $o$ and the sink $s$ as the maximum $N$ for which a path exists between $o$ and $s$ and all the nodes $x_i$, $i = 2, \ldots, N + 1$ have positive progress. We assume that $h_{s,\text{max}}$ between a source $o$ and any sink $s$ is known.
In Fig. 2, a source node $o$ transmits a data packet to the sink $s$. For the pair $(o, s)$, we define the distance as the length of the straight line connecting the source $o$ and sink $s$. According to our system model, the distance is known, which we denote by $r_s$. If the sink $s$ is not located within the transmission range of the source $o$, the data should be transmitted using the intermediate nodes of the PPP. Therefore, several hops might be needed to transmit the data from the source to the sink. Let $H_s$ denote the random variable representing the number of hops between the source and a sink $s$. The probability of connecting the source $o$ and the sink $s$ with $h$ number of hops, $h = 1, 2, \ldots$, is computed in [30] as

$$P(H_s = h) = C_{r_s}[1 - e^{-\Lambda_s|A_s|}]^{h-1},$$

(4)

where $C_{r_s}$ is a constant which depends on the distance $r_s$ between the source and the sink node $s$ and $0 \leq C_{r_s} \leq 1$. Moreover, in (4), $A_s$ denotes the intersection area between the transmission range of a node and its next node in a path which can be calculated as [30]

$$|A_s| = R^2[2 \cos^{-1}(\frac{r_s}{2R})] - \sin(2 \cos^{-1}(\frac{r_s}{2R})).$$

The expected value of the number of hops $H_s$ will therefore be calculated as

$$E[H_s] = \sum_{H_s=1}^{h_{s,\text{max}}} H_s P(H_s) = C_{r_s} \sum_{H_s=1}^{h_{s,\text{max}}} H_s[1 - e^{-\Lambda_s|A_s|}]^{(H_s-1)}.$$  

(5)
However, in our setting, there is a possibility of outage for a transmission between any pair of nodes \( x_i \) and \( x_{i+1} \); this means that the transmitter might require several attempts until a successful reception at the receiver is achieved. Let \( K_i, i = 1, 2, \ldots \), denote the random variable representing the number of Bernoulli trials (time instances) needed for the first successful connection between the transmitter-receiver pair \( x_i \) and \( x_{i+1} \). Then we have

\[
P(K_i = k_i) = p_{s,\theta}(1 - p_{s,\theta})^{k_i-1}.
\]  

(6)

In words, the number of time instances (attempts) needed to achieve the first successful connection follows a geometric distribution.

The total transmission time \( g_{s,\theta} \) between the source and a sink \( s \) at round \( \theta \) is given by

\[
g_{s,\theta} = \sum_{i=1}^{H_s} K_i.
\]

(7)

The following proposition states the statistical characteristics of \( g_{s,\theta} \).

**Proposition 1.** The transmission time \( g_{s,\theta} \) is a random variable with the probability distribution

\[
P(g_{s,\theta} = k) = C_{r_s} \sum_{h=1}^{h_{s,\max}} \binom{k-1}{h-1} p_{s,\theta}^h (1 - p_{s,\theta})^{k-h} [1 - e^{-\Lambda_s |A_s|}]^{h-1}, \quad h \leq k = 1, 2, \ldots
\]

and the expected value

\[
\mathbb{E}[g_{s,\theta}] = \frac{C_{r_s} \sum_{h=1}^{h_{s,\max}} H_s [1 - e^{-\Lambda_s |A_s|}]^{(H_s-1)}}{p_{s,\theta}}.
\]

(8)

(9)

**Proof.** See Appendix VIII-A

3) **Delay Time and Reward**

Finally, the following proposition characterizes the statistics of the variable reward.

**Proposition 2.** Reward \( r_{s,\theta} \) is a random variable with Bernoulli distribution. Moreover, it
has a piece-wise constant expected value as

\[
\mu_{s,\theta} = C_{rs} \sum_{k=1}^{\infty} \left[ \left( 1 - e^{-(\rho_s - \lambda_{s,\theta})(\delta - k)} \right) \sum_{h=1}^{h_{s,max}} \left( \frac{k - 1}{h - 1} \right) p_{s,\theta}^h (1 - p_{s,\theta})^{k-h} [1 - e^{-\Lambda_{s,\theta} \delta}]^{h-1} \right].
\] (10)

Proof. See Appendix VIII-B.

\[\Box\]

B. Cost

Naturally, every offloading round results in some energy consumption due to data transmission to the server as well as data processing at the server. Consider an offloading round \(\theta\) in which the computational task is offloaded to a server \(s\). We denote the total required energy by \(c_{s,\theta}\). Due to the energy scarcity, we define the cost in terms of the consumed energy. In general, the consumed energy, i.e., the cost \(c_{s,\theta}\), is a function of the duration of the data transmission and data processing. More precisely, it consists of the following parts:

\begin{itemize}
  \item The energy required for data transmission, denoted by \(v_g(g_{s,\theta})p_g\), where \(p_g\) is the energy consumption rate for data transmission. Note that \(g_{s,\theta}\) represents the time required for sending the data from the user to the server \(s\) at round \(\theta\). However, we need to consider the time required for sending the data from the server \(s\) back to the user at the same round \(\theta\). We do so by considering that the function \(v_g(\cdot)\) takes into account this round trip, for instance, via additionally multiplying the random variable \(g_{s,\theta}\) by 2.
  \item The energy required for the computational job (data processing) at the server, denoted by \(v_f(f_{s,\theta})p_f\), where \(p_f\) is the energy consumption rate for accomplishing the job at the specific server \(s\).
\end{itemize}

Note that \(p_g\) and \(p_f\) are known system parameters. Generally, \(v_g(\cdot)\) and \(v_f(\cdot)\), can be any invertible function; in this paper, for the sake of computation, we consider linear functions. Consequently, we have

\[
c_{s,\theta} = a_s f_{s,\theta} + a'_s g_{s,\theta} + a''_s,
\] (11)

where \(a_s, a'_s > 0\), and \(a''_s \geq 0\). Therefore, we have \(\min_{s,\theta} c_{s,\theta} = a'_s + a''_s\). Note that the cost \(c_{s,\theta}\) takes its minimum when the data is successfully transmitted via only one hop and in the first attempt and also when the process time \(f_{s,\theta} = 0\).

Similar to the reward (see Section III-A), in the following proposition we determine the probability distribution function of the random variable cost \(c_{s,\theta}\).
Proposition 3. The cost $c_{s,\theta} \geq a'_s + a''_s$ for an offloading round $\theta$ between the user’s device and any server $s$ is a random variable with the probability distribution as follows

$$\mathbb{P}(c_{s,\theta} = x) =$$

$$\frac{C_{r_s}}{a_s} \sum_{k=1}^{\lfloor \frac{x-a''_s}{a'_s} \rfloor} \left( (\rho_s - \lambda_{s,\theta}) e^{-(\rho_s - \lambda_{s,\theta})(\frac{x-a''_s}{a'_s})} \right) \sum_{h=1}^{h_{s,\max}} \left( k - 1 \right) p_{s,\theta}^h (1 - p_{s,\theta})^{k-h} \left[ 1 - e^{-\Lambda_s |A_s|} \right]^{h-1}. \quad (12)$$

Moreover, its expected value is equal to

$$\eta_{s,\theta} = \frac{a_s}{\rho_s - \lambda_{s,\theta}} + a'_s C_{r_s} \sum_{h=1}^{h_{s,\max}} \frac{H_s \left[ 1 - e^{-\Lambda_s |A_s|} \right] (H_s-1)}{p_{s,\theta}} + a''_s. \quad (13)$$

Proof. See Appendix VIII-C. \[\square\]

IV. Model and Solution based on Multi-Armed Bandits

To solve the server selection problem, we take advantage of a class of sequential optimization problems with limited information, namely, the Multi-Armed Bandit (MAB) problem. We consider an MAB problem which portrays a gambler (player) facing a number of arms (actions) with unknown cost and reward generating processes. By pulling an arm in each round $\theta = 1, 2, ..., \gamma$, the player pays some cost and receives some reward. Given a limited budget, the goal of the gambler is to maximize its accumulated utility (reward per cost) over the finite gambling horizon. In the rest of this section, we formulate the server selection problem in the MAB framework and propose an algorithm to solve this problem.

A. Budget-Limited Multi-Armed Bandits with Piece-wise Stationary Reward and Cost

In the following, we describe the server selection problem in the context of an MAB problem. We denote the set of arms (servers) of the MAB by $S = \{1, 2, \ldots, S\}$. At each round $\theta$, a player chooses an arm $i \in S$, incurs a cost $c_{i,\theta}$ and receives a reward $r_{i,\theta}$, which have the following characteristics:

- The random process of reward is piece-wise stationary and reward variables follow a probability distribution with mean $\mu_{i,\theta}$ at round $\theta$. The rewards are bounded from above, i.e., there exists a constant $r_{\text{max}} > 0$ such that $r_{i,\theta} \leq r_{\text{max}} \forall i, \theta$.
- Similar to the reward, the random process of cost is piece-wise stationary and cost variables follow a probability distribution with mean $\eta_{i,\theta}$ at round $\theta$. The costs are
bounded from below, i.e., there exists a constant $c_{\min}$ such that $c_{\min} \leq c_{i, \theta} \forall i, \theta$.

The player can continue gambling as long as its accumulated cost remains below the given budget $B$. Ideally, the player’s goal is to maximize its accumulated reward until the last round, which we denote by $T(B)$. Formally, the problem can be formulated as

$$\max_{I_{\theta} \in A} \sum_{\theta=1}^{T(B)} r_{I_{\theta}, \theta}$$

s.t. $$\sum_{\theta=1}^{T(B)} c_{I_{\theta}, \theta} \leq B,$$

where $I_{\theta}$ denotes the played arm at round $\theta$.

The problem stated by (14) is infeasible to solve since the instantaneous outcome of the random variables reward and cost are not known a priori. Moreover, $T(B)$ is a random variable because it depends on the summation of some random variable cost, which by itself depends on the choice of the arm. Therefore, we suggest an alternative problem formulation, as described in the following. First, we define the utility in a way that it includes both reward and cost revealed by an arm upon pulling. Such utility can be used to evaluate the efficiency of a choice of arm as it takes both the reward and cost into account. More precisely, we define the utility as reward-per-cost. Formally, the utility gained by playing arm $I_{\theta}$ at the round $\theta$ is given by

$$u_{I_{\theta}, \theta} = \frac{r_{I_{\theta}, \theta}}{c_{I_{\theta}, \theta}}.$$  \hfill (15)

We define the regret of the player upon pulling the arm $I_{\theta}$ at the round $\theta$ as

$$\text{Regret}_{\theta} = \frac{r_{i^*_\theta, \theta}}{c_{i^*_\theta, \theta}} - \frac{r_{I_{\theta}, \theta}}{c_{I_{\theta}, \theta}},$$

where $i^*_\theta = \arg\max_{i \in A} \frac{\mu_{i, \theta}}{\eta_{i, \theta}}$. Therefore, the cumulative regret yields

$$R_{T(B)} = \sum_{\theta=1}^{T(B)} \left[ \frac{r_{i^*_\theta, \theta}}{c_{i^*_\theta, \theta}} - \frac{r_{I_{\theta}, \theta}}{c_{I_{\theta}, \theta}} \right].$$

Finally, the agent’s goal is to minimize the expected cumulative regret. Formally,

$$\min_{I_{\theta} \in S} \mathbb{E}[R_{T(B)}]$$

Following the work by [17] and [19], we propose the Algorithm 1 to solve the problem [18].
In this algorithm, we define the average reward and average cost as

\[
\bar{r}_{\theta}(\tau, i) = \frac{\sum_{k=\theta-\tau+1}^{\theta} r_{i,k} \mathbb{I}\{I_k = i\}}{N_{\theta}(\tau, i)},
\]

(19)

and

\[
\bar{c}_{\theta}(\tau, i) = \frac{\sum_{k=\theta-\tau+1}^{\theta} c_{i,k} \mathbb{I}\{I_k = i\}}{N_{\theta}(\tau, i)},
\]

(20)

respectively, where

\[
N_{\theta}(\tau, i) = \sum_{k=\theta-\tau+1}^{\theta} \mathbb{I}\{I_k = i\}.
\]

We also define

\[
E_{\theta}(\tau, i) = \frac{(1 + \frac{r_{\max}}{c_{\min}}) r_{\max} \sqrt{\frac{\xi \log (\min(\theta, \tau))}{N_{\theta}(\tau, i)}}}{c_{\min} - r_{\max}} \sqrt{\frac{\xi \log (\min(\theta, \tau))}{N_{\theta}(\tau, i)}}.
\]

(21)

Moreover, \(\xi\) and \(\tau\) are tunable parameters. We will elaborate on the choice of these parameters later in Section VII.

**Algorithm 1** Budgeted Piece-wise stationary Reward with Piece-wise stationary Cost Sliding Window Upper Confidence Bound (BPRPC-SWUCB)

Input: Window length \(\tau\), parameters \(\xi\), \(r_{\max}\), and \(c_{\min}\)

for \(\theta = 1, \ldots, S\) do

Play arm \(I_{\theta} = \theta\)

observe the reward \(r_{I_{\theta}, \theta}\) and the cost \(c_{I_{\theta}, \theta}\)

end for

while \(\sum_{k=1}^{\theta} c_{I_{\theta}, k} \leq B\) do

Play with arm \(I_{\theta}\) which solves

\[
I_{\theta} = \arg \max_{i \in A} \frac{\bar{r}_{\theta}(\tau, i)}{\bar{c}_{\theta}(\tau, i)} + E_{\theta}(\tau, i),
\]

where \(\bar{r}_{\theta}\) and \(\bar{c}_{\theta}\) are defined by (19) and (20), respectively. Moreover, \(E_{\theta}\) is defined by (21).

Observe the reward \(r_{I_{\theta}, \theta}\) and the cost \(c_{I_{\theta}, \theta}\)

end while

V. Analysis of BPRPC-SWUCB

In this section, we prove an upper bound on the expected cumulative regret of the BPRPC-SWUCB by upper bounding the expected number of times an arm was chosen where it
was not the optimal arm. As mentioned earlier, our work is inspired by the work done in [19] and [17] and there are many similarities between our proof and the ones explained in the mentioned references. However, the regret bound is slightly different than SW-UCB algorithm in [19], as expected. We use the following definition in the rest of this paper.

\[
\Delta_{\pi_{T(B)}(i)} = \min \left\{ \frac{\mu_{i}^{*} - \theta}{\eta_{i}^{*}} - \frac{\mu_{i} - \theta}{\eta_{i}} \mid \forall \theta \in \{1, \ldots, T(B)\} \right\}
\]

(22)

Let us denote by \(\mathbb{P}\) and \(\mathbb{E}\) the probability and expectation under the policy of our algorithm, respectively. Moreover, we denote by \(\tilde{N}_{T(B)}(i)\) the number of rounds arm \(i\) has been played when it was not the optimal arm. With this definition, as shown in [19], the upper bound on the expected cumulative regret will be as follows

\[
\mathbb{E}[R_{T(B)}] = \mathbb{E} \left[ \sum_{\theta=1}^{T(B)} \sum_{i: \mu_{i}^{*} - \theta < \mu_{i}^{*} - \eta_{i}^{*} \theta} \frac{r_{i}^{*} \sum_{\theta} r_{i,\theta} 1_{\{I_{\theta} = i\}}}{c_{i,\theta}} - \frac{r_{i,\theta} 1_{\{I_{\theta} = i\}}}{c_{i,\theta}} \right] \leq r_{\max} c_{\min} \sum_{i=1}^{S} \mathbb{E}[\tilde{N}_{T(B)}(i)].
\]

(23)

**Theorem 1.** Let us denote by \(\Upsilon_{T(B)}\) the number of change points before time \(T(B)\) corresponding to both the reward and cost distribution. For \(\xi > \frac{1}{2}\), any integer \(\tau\), and any arm \(i \in S\)

\[
\mathbb{E}[\tilde{N}_{T(B)}(i)] \leq C(\tau, i) T(B) \frac{\log(\tau)}{\tau} + \tau \Upsilon_{T(B)} + 2 \log^{2}(\tau),
\]

(24)

where

\[
C(\tau, i) = \left( \frac{2(1 + r_{\max}) + \Delta \mu_{T(B)}(i)}{c_{\min} \Delta \mu_{T(B)}(i)} \right)^{2} r_{\max}^{2} \xi \left[ \frac{T(B)}{\tau} \right] + 4 \frac{\log(\tau)}{\log(1 + 4 \sqrt{1 - (2\xi)^{-1}})}.
\]

(25)

**Proof.** See Appendix VIII-D.

\[\blacksquare\]

**Remark 1.** Note that, as suggested in [20], we have \(T(B) \leq \frac{B}{c_{\min}}\). Considering this upper bound for the stopping time yields the following upper bound on the expected regret

\[
\mathbb{E}[R_{T(B)}] \leq \frac{r_{\max}}{c_{\min}} \sum_{i=1}^{S} \left( C(\tau, i) \frac{B}{c_{\min}} \frac{\log(\tau)}{\tau} + \tau \Upsilon_{B} c_{\min} + 2 \log^{2}(\tau) \right),
\]

(26)

where \(C(\tau, i)\) is the same as equation (25) with \(T(B) = \frac{B}{c_{\min}}\).

**Remark 2.** Computational Complexity
The computational complexity of BPRPC-SWUCB is linear with respect to the time horizon $T(B)$. It is worth noting that BPRPC-SWUCB only stores the action and reward/cost history of the last $\tau$ rounds, hence it is more space-efficient compared to the algorithms that rely on the full history. It has a linear computational complexity with respect to the window length $\tau$. Finally, depending on the search algorithm used to find the highest UCB index, the computational complexity can vary with respect to the number of servers (arms) $S$. For example, if we use the merge sort to sort the UCB indices of $S$ arms, BPRPC-SWUCB will have a complexity (with respect to the number of arms) of order $O(S \log S)$. 

Remark 3. By choosing $\tau = \sqrt{\frac{B}{\min c_{min}}} \frac{\log(B c_{min})}{\min c_{min}}$, we achieve the following

$$
\mathbb{E} \left[ \tilde{N}_{\frac{B}{c_{min}}(i)} \right] = O \left( \sqrt{\frac{Y}{\min c_{min}} B c_{min} \log \left( \frac{B}{c_{min}} \right)} \right).
$$

If we assume that the growth rate of the number of change points is independent of $T(B) = \frac{B}{c_{min}}$, we achieve an upper bound $O \left( \sqrt{\frac{B}{\min c_{min}} \log \left( \frac{B}{c_{min}} \right)} \right)$ for the regret.

VI. Numerical Analysis

We consider a network consisting of one offloading user and three servers. To better demonstrate the results in our simulation, we chose a reasonably small number of servers, $|S| = 3$. Moreover, we consider a maximum total of 6 change points in the reward or cost distribution (including the one corresponding to the initial round). As demonstrated in Section III-A we consider a Bernoulli distribution with the piece-wise constant mean for the reward. Hence, $r_{s,\theta} \in \{0, 1\}$. The distribution for the cost is derived in Section III-B. Note that we can rewrite the probability distribution (12) for the cost as follows

$$
\mathbb{P}(c_{s,\theta} = x) = \begin{cases} 
C_x (\frac{\rho_s - \lambda_{s,\theta}}{a_s}) e^{-(\frac{\rho_s - \lambda_{s,\theta}}{a_s})x}, & x \geq a'_s + a''_s \\
0, & x < a'_s + a''_s 
\end{cases}
$$

where $C_x$ is a constant which depends on $x$. However, for a fixed $x$, this constant is finite due to the summations being finite. Note that (27) is similar to an exponential distribution with the support $[a'_s + a''_s, \infty]$. In our simulation, for simplicity, we consider an exponential distribution with $a_s, a'_s = 1$ and $a''_s = 0$, $\forall s \in S$. Table II summarizes the change points in the mean reward and mean cost for each server together with their values.
We evaluate the performance of our algorithm by comparing it with a policy, called oracle, which chooses the arm (server) with the highest mean reward per mean cost at each round. As explained in our analysis in Section IV, we assess the performance of BPRPC-SWUCB by investigating its regret over a time horizon $T(B)$. We depict several figures in this section and illustrate the growth in the cumulative regret as a function of a growth in the budget $B$, or equivalently, $T(B)$. Moreover, we compare our methods with the following MAB-based policies:

- **$\varepsilon$-Greedy**: At each round $\theta$, $\varepsilon$-Greedy chooses with probability $1-\varepsilon$ the arm which has the highest empirically computed average reward per average cost, and with probability $\varepsilon$ a randomly chosen arm among the others. To be more efficient, several choices for $\varepsilon$ have been proposed in the literature. Here we choose $\varepsilon = \frac{dS}{d^2}$ at each round $\theta$, where $d > 0$ and $0 < d' \leq \min_{s} \frac{\Delta \mu_T(s)}{\eta_T(s)}$ [32], where $\Delta \mu_T(s)$ is defined in (22). Choosing such parameter has the advantage that as the round of play increases, the $\varepsilon$-greedy algorithm explores less and exploits more the arm which has shown a higher empirically computed average reward per average cost.

- **UCB1**: One of the most used UCB-based algorithms [32], which calculates an index for
each arm $s$ at round $\theta$ as 

$$\sum_{k=1}^{\theta} \frac{r_{k,s} + I_{k,s}}{N_{\theta}(s)} + r_{\max} \sqrt{\frac{\sum \log(\theta)}{N_{\theta}(s)}},$$

where $\xi'$ is a tunable parameter, $N_{\theta}(s) = \sum_{k=1}^{\theta} \mathbb{1}_{I_k = s}$, and $I_k$ denotes the played arm under the applied policy at round $k$.

- **UCB-based algorithm**: We define a policy which explores the arms similar to UCB1 but exploits similar to our proposed algorithm. By implementing this policy, we can compare the performance of our algorithm with a general UCB-based algorithm in a non-stationary environment. At a given round $\theta$, the policy described above calculates an index for an arm $s$ as

$$\bar{r}_\theta(s) + \frac{\sum \log(\theta)}{N_{\theta}(s)} \bar{c}_\theta(s),$$

where $\xi''$ is a tunable parameter. Moreover, we have $N_{\theta}(s) = \sum_{k=1}^{\theta} \mathbb{1}_{I_k = s}$, $\bar{r}_\theta(s) = \frac{1}{N_{\theta}(s)} \sum_{k=1}^{\theta} r_{s,k} \mathbb{1}_{I_k = s}$, and $\bar{c}_\theta(s) = \frac{1}{N_{\theta}(s)} \sum_{k=1}^{\theta} c_{s,k} \mathbb{1}_{I_k = s}$.

- **Uniformly at Random (UaR)**: As suggested in the name, this algorithm chooses an arm at each round $\theta$ according to a uniform distribution over the arms regardless of their past outcomes of the reward and cost.

- **UCB-BV1**: This algorithm is different from the aforementioned ones with respect to its setting. It is specifically designed for stationary MABs where each arm has a reward and a cost variable associated with it [17]. UCB-BV1 calculates an index for an arm $s$ at each round $\theta$ as

$$\bar{r}_\theta(s) + \frac{\sum \log(\theta-1)}{c_{\min}} \bar{c}_\theta(s) - \frac{\sum \log(\theta-1)}{c_{\min}},$$

where $N_{\theta}(s)$, $\bar{r}_\theta(s)$, and $\bar{c}_\theta(s)$ are defined as above.

To be comparable with other algorithms, we chose the system variables so that to fulfill the prerequisites of the other algorithms. The tuned parameters used in our simulation are listed in table III. Note that, we chose a rather small window length $\tau$. As we see shortly, choosing a bigger $\tau$ with $\xi = 0.6$ results in a lower regret but requires a higher storage capacity to store the past taken actions and their corresponding reward and cost. Therefore, we set $\tau$ so that to make a trade-off between the regret optimization and storage efficiency.

Fig. 3 depicts the simulation results of running different policies to solve the computation offloading problem in the aforementioned network with a given budget $B = 12000$. Fig. 3a

| Policy Settings | Policy | UCB1 | BPRPC-SWUCB | $\varepsilon$-Greedy | UCB-BV1 | UCB-based |
|-----------------|--------|------|-------------|------------------|--------|-----------|
| Parameters      | $\xi = 0.6$ | $r_{\max} = 1$ | $\xi = 0.6$ | $d = 0.24$ | $c_{\min} = 1$ | $\xi'' = 0.6$ | $r_{\max} = 1$ | $c_{\min} = 1$ |
| $r_{\max} = 1$  | $c_{\min} = 1$ | $\tau = 2000$ | $d' = 0.24$ | $c_{\min} = 1$ |
Fig. 3: The computational offloading problem in a dynamic environment; (a) Changes in mean reward per mean cost for each server. (b) Regret of different policies for a given same budget. (c) Evolution of the expected value of the reward and cost variables.

shows the evolution of the mean reward per mean cost at each round $\theta$ for the 3 servers and Fig. 3c depicts the variations in the expected value of the reward and cost. Note that, as mentioned before, the change points do not have to be identical; for example, at round $\theta = 1000$, the expected reward for server 1 is changing while its expected cost remains fixed. Fig. 3b shows the trend of regret for each policy. To be comparable, we truncated the graph of all policies at the smallest time horizon $T(B)$ among the different policies. As we see, BPRPC-SWUCB surpasses all other policies and is able to conform faster to abrupt changes in the environment. As a result, BPRPC-SWUCB has a smoother curve where does not exist sudden jumps in the regret, unlike other policies. The regret of other policies grows faster than BPRPC-SWUCB especially close to change points. Note that, there is not any sudden rise in the UaR curve because this algorithm doesn’t calculate any index to approximate the highest average reward per average cost in each round, and it chooses the servers only at random. Finally, note that algorithms other than BPRPC-SWUCB fail in their performance due to their nature; they are designed to perform well in a stationary environment.

Fig. 4 depicts the highest mean reward per mean cost at each round (solid blue line), which is known to the oracle, and the empirically computed average reward per average cost of the chosen server by the other policies at each round. This figure illustrates well why the BPRPC-SWUCB is performing better than other algorithms; it correctly chooses the optimal server in more number of rounds (compared to other policies) due to its ability to detect the changes in the environment.
Fig. 4: The highest mean reward per mean cost at each round chosen by the oracle and the empirically computed average reward per average cost of the chosen server by different policies at each round.

To investigate the choice of servers by BPRPC-SWUCB, Fig. 5 compares the performance of the BPRPC-SWUCB with the oracle in terms of the selection of servers after each change point. As expected, in the first few rounds, mainly before the second change point at $\theta = 500$, the BPRPC-SWUCB is investing more on exploring the servers to approximate the mean.

Fig. 5: Server choice for the oracle vs. BPRPC-SWUCB.
Fig. 6: The effect of parameters on the performance of BPRPC-SWUCB; 6a Regret obtained for different choices of $\xi$ and $\tau$. 6b Regret over a time horizon for $\xi = 0.6$ and different window lengths $\tau$.

As mentioned earlier, the performance of BPRPC-SWUCB highly depends on the choice of parameters. To better demonstrate this, we have shown the effect of parameters in Fig. 6. Fig. 6a depicts an overview of the amount of cumulative regret obtained for different choices of the parameters, namely $\xi$ and the window length $\tau$. We see that for smaller values of $\xi$ and larger values of $\tau$ we have smaller regret. This graph is also obtained for a given budget $B = 12000$. Fig. 6b shows the trend of regret for a slice of the previous figure corresponding to $\xi = 0.6$. It clearly shows that for $\xi = 0.6$, a bigger $\tau$ results in a smaller regret. However, when taking the storage efficiency into account, a smaller $\tau$ would be still beneficial.

**Remark 4. Parameter Selection**

*Fig. 6 might appear different for a problem with different settings, for example, a problem with different change points, number of change points, number of servers, and so on. Therefore, the tunable parameters $\tau$ and $\xi$ should be chosen based on the given problem. Generally, $\xi$ controls the exploration power of the algorithm. A larger $\xi$ results in giving more importance to the*
exploration rather than exploiting the arm which has shown promising results. In problems with more number of servers, a larger $\xi$ can be useful. The window length $\tau$ is chosen based on the number and frequency of change points. In general, selecting a smaller $\tau$ would be more suitable if change points occur often. Moreover, a smaller $\tau$ results in storage efficiency. In an environment where the system variables change seldom, we may choose a larger $\tau$.

VII. Conclusion

In this paper, we mainly focused on the computation offloading problem in a dynamic network under uncertainty; nonetheless, the theoretical results are applicable in a number of contexts. We derived the probability distribution of the required time for data transmission from the user’s device to an edge server. Moreover, we analyzed the probability distribution of the required time for data processing in a server. By leveraging the aforementioned distributions, we derived the probability distribution of total required time and energy for the whole offloading process. We then cast the server selection problem in the MAB framework. We developed a novel UCB-based algorithm, namely BPRPC-SWUCB, to solve the formulated problem. We proved that if the growth rate of the number of change points is independent of the time horizon $T(B)$, BPRPC-SWUCB may achieve a regret of order $O(\sqrt{\frac{B}{c_{\min}} \log \left(\frac{B}{c_{\min}}\right)})$. The numerical results demonstrated that BPRPC-SWUCB performs well in a non-stationary environment.

VIII. Appendix

A. Proof of Proposition 1

Fix a sink node $s$ and a round $\theta$. We will derive the probability distribution of the transmission time $g_{s,\theta}$ by finding the joint distribution of the transmission time $g_{s,\theta}$ and the number of hops $H_s$. From the basics of probability theory we have

$$
\Pr(g_{s,\theta} = k) = \sum_{h=1}^{h_{\text{max}}} \Pr(g_{s,\theta} = k, H_s = h) = \sum_{h=1}^{h_{\text{max}}} \Pr(g_{s,\theta} = k \mid H_s = h) \Pr(H_s = h). \tag{28}
$$

The second term $\Pr(H_s = h)$ is given in (4). As for the first term, we start by calculating it for the first few cases, i.e., $h = 1, 2, 3$.

For $h = 1$, we have $g_{s,\theta} = K_1$. Therefore,

$$
\Pr(g_{s,\theta} = k \mid H_s = 1) = \Pr(K_1 = k) = p_{s,\theta}(1 - p_{s,\theta})^{k-1}.
$$
For $h = 2$, we have $g_{s,\theta} = K_1 + K_2$. It yields

$$
P(g_{s,\theta} = k | H_s = 2) = P(K_1 + K_2 = k)$$

$$= P(K_1 = k - 1, K_2 = 1) + \cdots + P(K_1 = 1, K_2 = k - 1)$$

$$\overset{(a)}{=} P(K_1 = k - 1)P(K_2 = 1) + \cdots + P(K_1 = 1)P(K_2 = k - 1)$$

$$\overset{(b)}{=} (k - 1)p_{s,\theta}^2(1 - p_{s,\theta})^{k-2} = \left(\frac{k - 1}{1}\right)p_{s,\theta}^2(1 - p_{s,\theta})^{k-2},$$

where (a) follows from the fact that $K_i, i = 1, 2, \ldots$ are independent geometric random variables with the same parameter $p_{s,\theta}$ and (b) follows from (6) and simple calculation.

The case $h = 3$ yields $g_{s,\theta} = K_1 + K_2 + K_3$. Hence,

$$P(g_{s,\theta} = k | H_s = 3) = P(K_1 + K_2 + K_3 = k)$$

$$= P(K_1 = k - 2, K_2 = 1, K_3 = 1) + \cdots + P(K_1 = 1, K_2 = 1, K_3 = k - 2)$$

$$\overset{(a)}{=} P(K_1 = k - 2)P(K_2 = 1)P(K_3 = 1) + \cdots + P(K_1 = 1)P(K_2 = 1)P(K_3 = k - 2)$$

$$\overset{(b)}{=} \frac{(k - 1)(k - 2)}{2}p_{s,\theta}^3(1 - p_{s,\theta})^{k-3} = \left(\frac{k - 1}{2}\right)p_{s,\theta}^3(1 - p_{s,\theta})^{k-3},$$

where (a) and (b) follows from the same reasoning as above. Therefore, we find the general form of the conditional probability distribution as follows

$$P(g_{s,\theta} = k | H_s = h) = \left(\frac{k - 1}{h - 1}\right)p_{s,\theta}^h(1 - p_{s,\theta})^{k-h}. \quad (29)$$

Thus, the first part of the proposition, i.e., (8), follows by substituting (4) and (29) in (28).

Since all the random variables $K_i$ are independent and have the same expected value, it holds

$$\mathbb{E}[g_{s,\theta}] = \mathbb{E}[K_i]\mathbb{E}[H_s]. \quad (30)$$

For $K_i, i = 1, 2, \ldots$, we have $\mathbb{E}[K_i] = \frac{1}{p_{s,\theta}}$. Therefore, the second part of the proposition, i.e., (9), follows by substituting (5) in (30).

**B. Proof of Proposition 2**

We have the distribution of the delay time $d_{s,\theta}$ as the convolution of the two probability distributions of process time $f_{s,\theta}$ and the transmission time $g_{s,\theta}$. From the definition of the
reward we have \( r_{s, \theta} \in \{0, 1\} \). Moreover, for any server \( s \in S \) and any round \( \theta \) we have

\[
P_s = P(r_{s, \theta} = 1) = P(d_{s, \theta} \leq \delta) = \sum_{k=1}^{\infty} \mathbb{P}(f_{s, \theta} \leq \delta - k)\mathbb{P}(g_{s, \theta} = k),
\]

\[
P_f = P(r_{s, \theta} = 0) = P(d_{s, \theta} \geq \delta) = 1 - P(d_{s, \theta} \leq \delta) = \sum_{k=1}^{\infty} \mathbb{P}(f_{s, \theta} \leq \delta - k)\mathbb{P}(g_{s, \theta} = k),
\]

where (a) and (b) follow from the following facts; \( d_{s, \theta} \) is a random variable which is the sum of two independent random variables \( f_{s, \theta} \) and \( g_{s, \theta} \). Moreover, \( \delta - k \geq 0 \). Note that, the processing time \( f_{s, \theta} \) is a continuous random variable whereas the transmission time \( g_{s, \theta} \) is a discrete random variable. We get the expressions for \( P_s \) and \( P_f \) using the distributions of \( f_{s, \theta} \) and \( g_{s, \theta} \). Therefore, we have \( P_s + P_f = 1 \). Hence, \( r_{s, \theta} \) is a Bernoulli random variable with expected value (success probability) \( P_s \). Thus the result follows from Assumption 1.

C. Proof of Proposition 3

To prove the distribution, we first start by deriving the Cumulative Distribution Function (CDF) of the random variable cost. This is not a trivial task since the random variable \( c_{s, \theta} \) is the result of linear combination of a continuous random variable \( f_{s, \theta} \) and a discrete random variable \( g_{s, \theta} \). In the following, \( F_Z \) and \( f_Z \) denote the CDF and the PDF of the random variable \( Z \), respectively. Fix a server \( s \) and an offloading round \( \theta \). We have

\[
F_c(c_{s, \theta} = x) = \mathbb{P}(c_{s, \theta} \leq x) = \mathbb{P}(a_s f_{s, \theta} + a'_s g_{s, \theta} + a''_s \leq x)
= \sum_{k=1}^{\infty} \mathbb{P}(a_s f_{s, \theta} + a'_s g_{s, \theta} + a''_s \leq x | g_{s, \theta} = k)\mathbb{P}(g_{s, \theta} = k)
= \sum_{k=1}^{\infty} \mathbb{P}(f_{s, \theta} \leq \frac{x - a''_s - a'_s k}{a_s})\mathbb{P}(g_{s, \theta} = k) = \sum_{k=1}^{\infty} F_f(\frac{x - a''_s - a'_s k}{a_s})\mathbb{P}(g_{s, \theta} = k).
\]

Taking the derivative of the above equation yields

\[
f_c(c_{s, \theta} = x) = \frac{d}{dx} F_c(c_{s, \theta} = x) = \sum_{k=1}^{\infty} \frac{d}{dx} F_f(\frac{x - a''_s - a'_s k}{a_s})\mathbb{P}(g_{s, \theta} = k)
= \sum_{k=1}^{\infty} \frac{1}{a_s} f_f(\frac{x - a''_s - a'_s k}{a_s})\mathbb{P}(g_{s, \theta} = k) = \frac{1}{a_s} \sum_{k=1}^{[\frac{x - a''_s}{a'_s}]} f_f(\frac{x - a''_s - a'_s k}{a_s})\mathbb{P}(g_{s, \theta} = k),
\]

where (*) follows from the fact that \( f_f(\frac{x - a''_s - a'_s k}{a_s}) = 0 \) for \( k > [\frac{x - a''_s}{a'_s}] \). The result follows by substituting the PDF of \( f_{s, \theta} \) and the PMF of \( g_{s, \theta} \), according to (3) and (8), respectively. The expected value (13) can be calculated by taking expectation from (11) and using the
linearity property of the expected value operator.

**D. Proof of Theorem 1**

Let $S(\tau) = \left( \frac{2(1 + \frac{r_{\max}}{c_{\min}} + \Delta \nu_{T(B)}(i))}{c_{\min} \Delta \nu_{T(B)}(i)} \right)^2 r_{\max}^2 \xi \log(\tau)$. Moreover, define $\Gamma(\tau)$ as follows:

$$\Gamma(\tau) = \left\{ \theta \in \{S + 1, \ldots, T(B)\} \right\} \left\{ \mu_{i,j} = \mu_{i,\theta} \& \eta_{i,j} = \eta_{i,\theta}, \forall i \in \{1, \ldots, S\} \& \forall j \text{ s.t. } \theta - \tau < j \leq \theta \right\}.$$ 

We have the following [19]:

$$\tilde{N}_{T(B)}(i) = 1 + \sum_{\theta = S+1}^{T(B)} I_{\theta \neq i^*_{\theta}} \leq 1 + \sum_{\theta = 1}^{T(B)} I_{\theta = i^*_{\theta}, \theta \in \{1, \ldots, S\}} + \sum_{\theta = S+1}^{T(B)} I_{\theta = i^*_{\theta}, \theta \geq S}$$

$$(*) \leq 1 + \left[ \frac{T(B)}{\tau} \right] S(\tau) + \sum_{\theta = S+1}^{T(B)} I_{\theta = i^*_{\theta}, \theta \geq S}$$

$$\leq 1 + \left[ \frac{T(B)}{\tau} \right] S(\tau) + \tau Y_{T(B)} + \sum_{\theta \in \Gamma(\tau)} I_{\theta = i^*_{\theta}, \theta \geq S}.$$ (31)

where $(*)$ follows from the Lemma (25) in [19]. For $\theta \in \Gamma(\tau)$ we have

$$\{I_{\theta \neq i^*_{\theta}, \theta \geq S} \leq 1 + \left[ \frac{T(B)}{\tau} \right] S(\tau) + \tau Y_{T(B)} + \sum_{\theta \in \Gamma(\tau)} I_{\theta = i^*_{\theta}, \theta \geq S}.$$ (31)

For the Event 3 we have

$$E_{\theta}(\tau, i) = \frac{(1 + \frac{r_{\max}}{c_{\min}})\max}{c_{\min} - r_{\max}} \sqrt{\frac{\xi \log(\min(\theta, \tau))}{N_{\theta}(\tau, i)}} \leq \frac{(1 + \frac{r_{\max}}{c_{\min}})\max}{c_{\min} - r_{\max}} \sqrt{\frac{\xi \log(\tau)}{S(\tau)}} = \frac{\Delta \nu_{T(B)}(i)}{2}.$$ (32)

Therefore, the Event 3 never occurs. Upper bound for the Events 1 and 2 are similar and we show only for Event 1. Note that if Event 1 occurs, it implies that at least one of the two following inequalities happens

$$\tilde{r}_{\theta}(\tau, i) > \mu_{i,\theta} + \epsilon_{\theta}(\tau, i),$$ (32)
or

\[ \bar{c}_\theta(\tau, i) < \eta_{i, \theta} - e_\theta(\tau, i), \quad (33) \]

where

\[ e_\theta(\tau, i) = r_{\max} \sqrt{\frac{\xi \log \{\min\{\theta, \tau\}\}}{N_\theta(\tau, i)}}. \quad (34) \]

To prove this, assume none of them happens. Therefore, we have [17]

\[ \frac{\bar{r}_\theta(\tau, i)}{\bar{c}_\theta(\tau, i)} - \frac{\mu_{i, \theta}}{\eta_{i, \theta}} = \frac{(\bar{r}_\theta(\tau, i) - \mu_{i, \theta})\eta_{i, \theta} + (\eta_{i, \theta} - \bar{c}_\theta(\tau, i))\mu_{i, \theta}}{\bar{c}_\theta(\tau, i)\eta_{i, \theta}} \]

\[ \leq \frac{e_\theta(\tau, i)}{\bar{c}_\theta(\tau, i)} + \frac{e_\theta(\tau, i)\mu_{i, \theta}}{\bar{c}_\theta(\tau, i)\eta_{i, \theta}} \leq \frac{e_\theta(\tau, i)}{c_{\min} - e_\theta(\tau, i)} + \frac{e_\theta(\tau, i)r_{\max}}{(c_{\min} - e_\theta(\tau, i))c_{\min}} = E_\theta(\tau, i). \]

Hence, we upper bound the probability of (32) and (33). Using Corollary (21) in [19] for any \( \nu > 0 \) we have

\[ P(\bar{r}_\theta(\tau, i) > \mu_{i, \theta} + e_\theta(\tau, i)) \leq \left[ \frac{\log \{\min\{\theta, \tau\}\}}{\log (1 + \nu)} \right] (\min\{\theta, \tau\})^{-2\xi(1 - \frac{\nu^2}{16})}, \quad (35) \]

and

\[ P(\bar{c}_\theta(\tau, i) < \eta_{i, \theta} - e_\theta(\tau, i)) \leq \left[ \frac{\log \{\min\{\theta, \tau\}\}}{\log (1 + \nu)} \right] (\min\{\theta, \tau\})^{-2\xi(1 - \frac{\nu^2}{16})}. \quad (36) \]

For the Event 2 we have similar results as follows

\[ P(\bar{r}_\theta(\tau, i^*_\theta) > \mu_{i^*_\theta, \theta} + e_\theta(\tau, i^*_\theta)) \leq \left[ \frac{\log \{\min\{\theta, \tau\}\}}{\log (1 + \nu)} \right] (\min\{\theta, \tau\})^{-2\xi(1 - \frac{\nu^2}{16})}, \quad (37) \]

and

\[ P(\bar{c}_\theta(\tau, i^*_\theta) < \eta_{i^*_\theta, \theta} - e_\theta(\tau, i^*_\theta)) \leq \left[ \frac{\log \{\min\{\theta, \tau\}\}}{\log (1 + \nu)} \right] (\min\{\theta, \tau\})^{-2\xi(1 - \frac{\nu^2}{16})}. \quad (38) \]

Choosing \( \nu = 4\sqrt{1 - \frac{1}{2\xi}} \) as suggested in [19], equations (31) and (35)-(38) result in

\[ \mathbb{E}[\tilde{N}_{T(B)}(i)] \leq 1 + \left[ \frac{T(B)}{\tau} \right] S(\tau) + \tau Y_{T(B)} + 4 \sum_{\theta=1}^{T(B)} \left[ \frac{\log \{\min\{\theta, \tau\}\}}{\log (1 + \nu)} \right] \min\{\theta, \tau\}. \]
We achieve the equation (24) using the following:

\[
\sum_{\theta=S+1}^{T(B)} \frac{\log (\min\{\theta, \tau\})}{\min\{\theta, \tau\}} \leq \sum_{\theta=2}^{\tau} \frac{\log (\theta)}{\theta} + \sum_{\theta=1}^{T(B)} \frac{\log (\tau)}{\tau} \leq \frac{1}{2} \log^2(\tau) + \frac{T(B) \log (\tau)}{\tau}.
\]

REFERENCES

[1] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, “A survey on the edge computing for the internet of things,” IEEE Access, vol. 6, pp. 6900–6919, 2018.
[2] S. Ješilo and G. Dán, “A game theoretic analysis of selfish mobile computation offloading,” in IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, May 2017, pp. 1–9.
[3] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile edge computing: A survey,” IEEE Internet of Things Journal, vol. 5, no. 1, pp. 450–465, Feb 2018.
[4] A. Ahmed and E. Ahmed, “A survey on mobile edge computing,” in 2016 10th International Conference on Intelligent Systems and Control (ISCO), Jan 2016, pp. 1–8.
[5] Herbert Robbins, “Some aspects of the sequential design of experiments,” Bulletin of the American Mathematical Society, vol. 58, no. 5, pp. 527–535, 1952.
[6] Tze Leung Lai and Herbert Robbins, “Asymptotically efficient adaptive allocation rules,” Advances in applied mathematics, vol. 6, no. 1, pp. 4–22, 1985.
[7] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al., “Regret analysis of stochastic and nonstochastic multi-armed bandit problems,” Foundations and Trends® in Machine Learning, vol. 5, no. 1, pp. 1–122, 2012.
[8] Setareh Maghsudi and Ekram Hossain, “Multi-armed bandits with application to 5g small cells,” CoRR, vol. abs/1510.00627, 2015.
[9] S. Yu, X. Wang, and R. Langar, “Computation offloading for mobile edge computing: A deep learning approach,” in 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), Oct 2017, pp. 1–6.
[10] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, “Offloading in mobile edge computing: Task allocation and computational frequency scaling,” IEEE Transactions on Communications, vol. 65, no. 8, pp. 3571–3584, 2017.
[11] J. Liu and Q. Zhang, “Offloading schemes in mobile edge computing for ultra-reliable low latency communications,” IEEE Access, vol. 6, pp. 12825–12837, 2018.
[12] D. Huang, P. Wang, and D. Niyato, “A dynamic offloading algorithm for mobile computing,” IEEE Transactions on Wireless Communications, vol. 11, no. 6, pp. 1991–1995, June 2012.
[13] Olga Munoz, Antonio Pascual-Iserte, and Josep 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.
[14] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, “Mobile-edge computing: Partial computation offloading using dynamic voltage scaling,” IEEE Transactions on Communications, vol. 64, no. 10, pp. 4268–4282, 2016.
[15] Deepsubhra Guha Roy, Debashis De, Anwesha Mukherjee, and Rajkumar Buyya, “Application-aware cloudlet selection for computation offloading in multi-cloudlet environment,” The Journal of Supercomputing, vol. 73, no. 4, pp. 1672–1690, 2017.
[16] Duc Van Le and Chen-Khong Tham, “A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds,” in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018, pp. 760–765.

[17] Wenkui Ding, Tao Qin, Xu-Dong Zhang, and Tie-Yan Liu, “Multi-armed bandit with budget constraint and variable costs,” in *AAAI*, 2013.

[18] Yingce Xia, Haifang Li, Tao Qin, Nenghai Yu, and Tie-Yan Liu, “Thompson sampling for budgeted multi-armed bandits,” in *IJCAI*, 2015.

[19] Aurélien Garivier and Eric Moulines, “On Upper-Confidence Bound Policies for Non-Stationary Bandit Problems,” *arXiv e-prints*, p. arXiv:0805.3415, May 2008.

[20] S. Maghsudi and D. Niyato, “On power-efficient planning in dynamic small cell networks,” *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 304–307, June 2018.

[21] Yingce Xia, Wenkui Ding, Xu-Dong Zhang, Nenghai Yu, and Tao Qin, “Budgeted bandit problems with continuous random costs,” in *ACML*, 2015.

[22] Yingce Xia, Tao Qin, Nenghai Yu, and Tie-Yan Liu, “Best action selection in a stochastic environment,” in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 758–766.

[23] Radha Krishna Ganti and Martin Haenggi, “Dynamic connectivity and path formation time in poisson networks,” *Wireless Networks*, vol. 20, no. 4, pp. 579–589, May 2014.

[24] François Baccelli and Bartłomiej Blaszczyszyn, *Stochastic Geometry and Wireless Networks, Volume I - Theory*, vol. 1 of *Foundations and Trends in Networking Vol. 3: No 3-4*, pp 249-449, Now Publishers, 2009, Stochastic Geometry and Wireless Networks, Volume II - Applications; see http://hal.inria.fr/inria-00403040.

[25] Martin Haenggi, Jeffrey G Andrews, François Baccelli, Olivier Dousse, and Massimo Franceschetti, “Stochastic geometry and random graphs for the analysis and design of wireless networks,” *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 7, 2009.

[26] János Sztrik, “Basic queueing theory,” 2012.

[27] Shermila Ranadheera, Setareh Maghsudi, and Ekram Hossain, “Computation offloading and activation of mobile edge computing servers: A minority game,” *IEEE Wireless Communications Letters*, 2018.

[28] H. Takagi and L. Kleinrock, “Optimal transmission ranges for randomly distributed packet radio terminals,” *IEEE Transactions on Communications*, vol. 32, no. 3, pp. 246–257, March 1984.

[29] Pedro Acevedo Contla and Milos Stojmenovic, “Estimating hop counts in position based routing schemes for ad hoc networks,” *Telecommunication Systems*, vol. 22, no. 1–4, pp. 109–118, 2003.

[30] Shadi M Harb and Janise Menair, “Analytical study of the expected number of hops in wireless ad hoc network,” in *International Conference on Wireless Algorithms, Systems, and Applications*. Springer, 2008, pp. 63–71.

[31] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms, Third Edition*, The MIT Press, 3rd edition, 2009.

[32] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer, “Finite-time analysis of the multiarmed bandit problem,” *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.