Reliability-Optimal Offloading in Low-latency Edge Computing Networks: Analytical and Reinforcement Learning Based Designs

Yao Zhu, Yulin Hu, Tianyu Yang, Tao Yang, Jannik Vogt and Anke Schmeink

***Abstract*—In this paper, we consider a multi-access edge computing (MEC) network with multiple servers. Due to the low latency constraints, the wireless data transmission/offloading is carried by finite blocklength codes. We characterize the reliability of the transmission phase in the finite blocklength regime and investigate the extreme event of queue length violation in the computation phase by applying extreme value theory. Under the assumption of perfect channel state information (CSI), we follow the obtained characterizations and provide an optimal framework design including server selection and time allocation aiming to minimize the overall error probability. Moreover, when only the outdated CSI is available, a deep reinforcement learning based design is proposed applying the deep deterministic policy gradient method. Via simulations, we validate the convexity proven in our analytical model and show the performance advantage of proposed analytical solution and learning-based solution comparing to the benchmark for perfect CSI and outdated CSI, respectively.**

Keywords*—ultra-reliable and low-latency communication, edge computing, finite blocklength, extreme value theory, deep reinforcement learning

I. Introduction

As a potential enabler for latency-sensitive applications, the newly emerged multi-access edge computing (MEC) technologies have attracted a lot of attention from both academy and industry, especially in the context of next generation mobile networks [2]. It provides flexible and rapid deployment for these applications via making the storage, control and more importantly the computation to the edge of networks [3]. In comparison to cloud computing which suffers from long-latency due to the logical and spatial distance between centralized servers and users [4], MEC networks distribute many proximity servers to the users, e.g., WiFi access points (APs) as well as cellular base stations (BSs), which shows significant performance advantages with respect to communication latency reduction. Nevertheless, the shortcoming of a MEC server is that its storage capacity and computational ability are relatively limited, which are likely not sufficient to guarantee the latency constraint for large computation tasks. Fortunately, by applying the recent cooperative offloading technique, this weak point can be compensated, i.e., the MEC network could exploit the computational resources from multiple servers in computing the task(s) for one user.

On the other hand, in the future beyond fifth generation (B5G) or sixth generation (6G), the explosive demands on ultra-reliable communications [5], [6] are arising, such as in the applications of vehicle-to-everything (V2X) and the Internet-of-Thing (IoT) [7]. According to standard in 3GPP [8], the ultra-reliable and low-latency communication (URLLC) should provide communication with reliability being larger than 99.999%. This ultra-reliable requirement makes the wireless network designer taking the critical scenarios [9] into account, while the following types of errors are expected to be modelled: from the communication perspective the transmission error in the finite blocklength regime [10], [11] and in the computation phase the delay constraint violation error [12].

In the URLLC scenarios, the reliability in the perspective of either communication or computation is coupled with the available delay tolerance. Note that the total task offloading process contains the data transmission via wireless links and computation at the MEC servers. For a given maximal allowed service delay, there exists a trade-off in the time allocation for the communication phase and computation phase. Moreover, on the one hand the user likely has different distance to different servers, and on the other hand the channels behave randomly. Hence, the qualities of the channels from the user to servers are also different. In addition, the MEC servers also have different (instantaneous) computing capabilities and buffer statuses. Hence, offloading one user's tasks to more and more servers is not necessarily always a good choice to optimize the reliability, since the partial offloading to a server with a poor channel likely results in an error. Oppositely, if we always only offload (large) tasks to just one or two servers, this may introduce a significant computing time cost, which further increases the delay violation probability. Thus, it is essential and beneficial optimally selecting servers which could enhance the service reliability of the MEC network.

The task offloading in URLLC-supported MEC networks has been investigated in [12], [15], [16]. For instance, the authors in [12] study the extreme probabilistic cases that queue length in the servers violates the delay threshold. It is proposed in [15] an offloading scheme by jointly considering the latency and reliability as a cost function. The work in [16] considers a cross-layer design and investigates the performance of processor-sharing MEC servers. However, all the above results are conducted under the assumption of

Some materials in Section III and Section IV-B have been presented at the IEEE International Conference on Communications (ICC) B5G-URLLC Workshop, Shanghai, China, in May 2019 [1], while the remaining parts are the extension to the conference version.

Y. Zhu and Y. Hu are with School of Electronic Information, Wuhan University, 430072 Wuhan, China and ISEK Research Area/Lab, RWTH Aachen University, D-52074 Aachen, Germany, (e-mail: zhu@isek.rwth-aachen.de, yulin.hu@whu.edu.cn). Y. Hu is the corresponding author.

J. Vogt and A. Schmeink are with ISEK Research Area/Lab, RWTH Aachen University, D-52074 Aachen, Germany (e-mail: {Vogt, schmeink}@isek.rwth-aachen.de).

Tianyu. Yang is with TU Berlin, D-10587 Berlin, Germany (e-mail:tianyu.yang@tu-berlin.de).

Tao. Yang is with Fudan University, Shanghai, China (e-mail: taoyang@fudan.edu.cn).

transmissions being arbitrarily reliable at Shannon's Capacity, which is only true in the so-called infinite blocklength regime. For low-latency MEC network where the transmission blocklengths are short, it is more essential to consider the finite blocklength (FBL) [17] impacht in the network design. To the best of our knowledge, it is still missing in the literature the optimal offloading design for a multiple servers MEC network operating with FBL codes, especially the joint framework optimization with multi-server selection.

Additionally, the aforementioned optimal offloading problem in the studied multi-server MEC network is in practice particularly intractable due to the uncertainty of the stochastic wireless channels and the continuous influences on the future statuses of the offloading process, i.e., the status of a server is time-dependent. Targeting of solving such dynamic decision making problem, reinforcement learning (RL) has been efficiently applied to automatically learn an optimal policy of environment based decision making [18]. By modeling the dynamic decision making problem as a Markov decision process (MDP) [19], RL algorithm trains an agent to find the optimal action policy by repeatedly interacting with the environment. During the continuous interaction the agent explores the system and updates its action policy based on the instantaneous feedback of the environment and finally converges to an optimal policy which provides a solution of time-variant decision making problem.

Conventional RL algorithms suffer from the problem of slow convergence, which is unsuitable and inapplicable to large-scale systems [18]. To overcome this shortage, the recently emerged and fast developed deep reinforcement learning (DRL) technique combines RL with the deep neural network (DNN) as the function approximator so that the capability of generalization and the robustness for large-scale problems are significantly enhanced. Benefiting from the various improved DRL algorithms, e.g., deep Q-learning [20], deep deterministic policy gradient (DDPG) [21], DRL has been also adopted in many MEC systems to solve the dynamic offloading problems with respect to multiple metrics, e.g., execution latency and energy consumption [22], [23], computation rate [24] and long-term utility [25]. Nevertheless, the aforementioned works all focus on the non-URLLC scenario. The authors in [26] develops a multi-level architecture with data-driven deep learning approach for URLLC, while the reliability optimization with model-based learning under FBL regime is missing. Our previous work [27] was the first to investigate the feasibility of model-based DRL solution for the reliability optimised offloading problem in the MEC network with a single server. To the best of our knowledge, model-based DRL solution to reliability optimization for a multi-server MEC system has not been addressed, especially when the servers are time-dependent.

Motivated by the above observations, we consider multi-server MEC network and propose an optimal framework design aiming at minimizing the overall error probability via optimally selecting the servers, assigning workload and allocating time to the two phases. Our major contributions are:

- We leverage the FBL comunication performance model from Polyanskiy [17] to characterize the transmission reliability and the extreme value theory (EVT) [28] to model the errors possibly incurred at a server.
- Assuming the system with time-independent MEC server and perfect channel state information (CSI), we formulate an optimization problem to minimize the overall error probability by optimally selecting the cooperative servers, frame structure and workload assignment. We prove the convexity of the sub-problems after decomposing the original one, and reformulate the original problem to a mixed integer convex problem, via which the global optimal solution is obtained.
- In addition to the conference version [1] we introduce the integer task partition model. Based on that, we exploit the Karush-Kuhn-Tucker (KKT) condition and propose an efficient algorithm to assign the workload and make the server selection at the same time, which reduces the computational complexity significantly.
- Moreover, in order to cope with a more practical scenario where server states are time-dependent and only the outdated CSI is available, we propose a DRL, specifically DDPG, based algorithm to learn the optimal dynamic offloading policy by designing a proper reward mechanism. In particular, we exploit the FBL regime and apply extreme value theory to model the error probability of communication and computation to address the sparse reward problem in the training. The numerical results show the outperformed performance of the proposed DRL based solution.

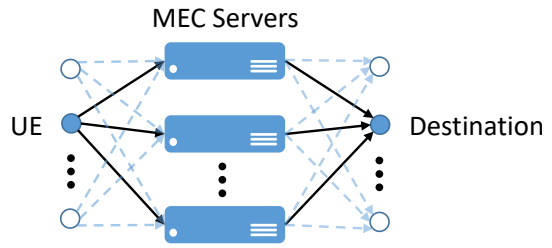


Fig. 1: An example of the considered system in which multiple UEs may connect to different servers, while the representative UE is mark as solid.

The rest of the paper is organized as follows. In Section II, the studied offloading problem in MEC system is described. In Section III, the end-to-end reliability of the considered network is characterized. The proposed analytical design is introduced with respect to framework and server selection in Section IV. After that, the DRL based algorithm for system with time-dependent servers is designed in Section V. The simulation results are provided in Section VI. Finally, Section VII concludes the whole work.

II. System Model

In this work, we consider a MEC network with K available servers $\mathcal{K} = \{1, .., K\}$ and one user equipment (UE), as shown in Fig. 1. The UE is assumed to have a compute-intensive application needed to be processed by the MEC network, e.g., a radar sensor has an anomaly detection that requires analysis. Therefore, the low-latency service of the application has to be offloaded to and computed by distributed MEC servers in the networks. The system is assumed to operate in a time-slotted fashion, where time is divided into frames. The application service (contains a group of computing tasks from various UEs with different task types) is required to be finished within a frame to satisfy the low-latency requirement. Moreover, as shown in Fig. 2, each frame

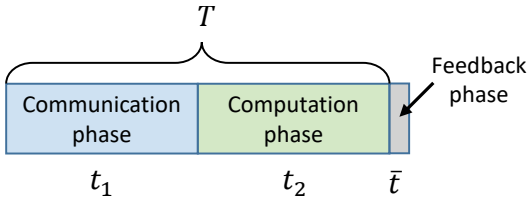


Fig. 2: The structure of a frame.

is divided into three phases: a communication phase with length of t_1, a computation phase with length of t_2 and a feedback phase with length of $\bar{t}$.

In the communication phase, the UE broadcasts the input data tasks of τ bits to different selected servers. Then, in the computation phase each selected server k processes the corresponding tasks with workloads c_k as long as it correctly decoding the data. Denote the total time length of the communication and computation phases by T, i.e., we have $T = t_1 + t_2$. Finally, the servers report the computation results to the destination in the feedback phase. Clearly, the total service time of the application satisfies $T + \bar{t}$. However, since the transmit power of the severs is relatively high than the transmit power of the UE P and the data size of the computation results is generally much smaller than the input data size, the length of feedback phase $\bar{t}$ is usually considered to be negligible in comparison to T [13]. In this work, we assume $\bar{t}$ to be constant and significantly small. In other words, in the framework optimization problem considered in the next section, we only focus on determining the optimal t_1 and t_2 while satisfying the constraint with a maximal allowed T. Let $T_{\rm s}$ denote the time duration of a single symbol. Therefore, we have the blocklength in the communication phase $n = \frac{t_1}{T_{\rm s}}$. Note that there are τ bits information transmitted in the communication phase. The corresponding coding rate can actually be written as $r_{\rm cod} = \frac{\tau}{n}$ (in bits/symbol). Wireless channels are assumed to experience quasi-static Rayleigh fading, and therefore the channel fading remains the same within each frame and varies from one frame to the next. Denote by the channel state of the link from the UE to server k by h_k, we have $h_k \sim \mathcal{CN}(0,1)$. Moreover, as a low-latency scenario with short blocklength is considered, the frame length is more likely shorter than the channel coherent time, i.e., the channels (via the same link) of adjacent frames are correlated. We adopt the widely-used Gauss-Markov model [30], [31] characterizing this channel correlation:

$$h_k = \rho h'_k + \sqrt{1-\rho^2}\Delta h_k, \tag{1}$$

where $0 \leq \rho^2 \leq 1$ is so-called the channel correlation coefficient, and h'_k is the channel state of the previous frame, $\Delta h \sim \mathcal{CN}(0,1)$ is a complex Gaussian random variable.

Channels are considered to be independent. We denote the noise power at server k by σ_k^2, UE transmit power by P, and the path-loss of the UE-server k link by ϕ_k. Then, the signal-to-noise ratio (SNR) of the received signal at server k, denoted by γ_k , is given by

$$\gamma_k = \frac{|h_k|^2 P}{\phi_k \sigma_k^2}. \tag{2}$$

For the computation process, we denote the total workloads for the application by $c_{\rm O}$ and the computation power at server k by f_k. We assume that each server starts to process tasks as long as the input data of the tasks is successfully received. Note that only the selected servers are active to the process tasks from the UE. We denote the decision vector of the server selection results by

$$\mathbf{A} = \{a_1, ..., a_K\}, \tag{3}$$

where a_k indicates the selection result of server k. For instance, when $a_k = 1$, this implies that the UE will offload server k. Case $a_k = 0$, the server k is not selected. Then, the set of the selected servers can be represented by $\hat{\mathcal{K}} = \{\hat{k} | \forall a_{\hat{k}} = 1\}$. Hence, the number of selected servers, i.e., the size of set $\hat{\mathcal{K}}$, is $\sum_k a_k$.

Then, the required task from UE is partitioned into subtasks, which are computed by the selected servers in set $\hat{\mathcal{K}}$. We denote by $\mathbf{C} = \{c_1, ..., c_K\}$ the assigned workload vector, where c_k is the assigned workload to the server k. Since the task is only finished when all assigned workload is executed, the assignment of workload must fulfill following conditions: $i)$ all workload should be assigned to the servers; $ii)$ the assigned workload should not exceed the total amount of workload; $iii)$ none of workload should be assigned to the non-selected server. Those conditions can be summarized as $c_{\rm O} = \sum_k c_k$ and $c_k \leq a_k c_{\rm O}$, where $k \in \mathcal{K}$.

As we consider a low-latency service, the total service time must be lower than a stringent threshold. Moreover, the reliability is also one of the major concerns in the our design. To this end, we investigate the communication behaviour via the wireless channel following the FBL theory and characterize the computation delay by exploiting the extreme value theory.

III. Characterization of the End-to-End Error Probability

In this section, we first model the FBL communication errors and extreme event-related computation errors, and subsequently characterize the end-to-end error probability.

A. Communication Error in the FBL Regime

Following the FBL transmission model [17], the (block) error probability of the transmission to server k is given by

$$\begin{aligned}\varepsilon_{1,k} &= \mathcal{P}(\gamma_k, r_{\rm cod}, n) \\ &\approx \text{qfunc}\left(\sqrt{\frac{n}{V_k(\gamma_k)}}(\mathcal{C}_k(\gamma_k) - r_{\rm cod})\log_e 2\right),\end{aligned} \tag{4}$$

where n and $r_{\rm cod}$ are blocklength and coding rate introduced in the the previous section, i.e., $n = \frac{t_1}{T_{\rm s}}$ (in symbols) and $r_{\rm cod} = \frac{\tau}{n}$ (in bits/symbol). qfunc(x) is the Q-function in statistics[1], i.e., qfunc$(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-t^2/2}dt$, and $\mathcal{C}_k = \log_2(1+\gamma_k)$ is the Shannon capacity. Moreover, $V_k(\gamma_k)$ is the channel dispersion [17] between the UE and the server k. Under a complex AWGN channel, $V_k = 1 - \frac{1}{(1+\gamma_k)^2}$.

B. Extreme Event-related Computation Error

In this subsection, we characterize the computation model at each server. We denote the computing time/delay of server k by D_k. Note that the server k may receive computation request from multiple UEs. In this work, we assume each server following the First-Come First-Serve (FCFS) policy,

[1]It represents the tail distribution function of the standard normal distribution, which is different from the Q-function in Q-learning.

i.e., the assigned task is only computed if previous tasks assigned to server k are all finished. In general, the execution time at server k contains the computing time and the queuing delay (i.e., waiting time delay in the queue buffer), which can be expressed as

$$D_k = \frac{c_k}{f_k} + D_k^{\text{rest}} + W_k, \tag{5}$$

where W_k is the queue delay for serving tasks which randomly arrive in the current frame before the current task and D_k^{rest} is the remaining queue delay from the previous frame. Specially, D_k^{rest} is influenced by not only the assigned workload c_k^{pre} at the previous frame, but also the intermission time T_k^{pre} from execution of the previous offloaded task to the arrival of the current offloaded task. Hence, it can be written as $D_k^{\text{rest}} = \max\{\frac{c_k^{\text{pre}}}{f_k} - T_k^{\text{pre}}, 0\}$.

A computation delay violation error occurs at server k, if the server fails in finishing the assigned tasks within t_2. The probability of this computation error at server k is expressed by

$$\varepsilon_{2,k} = \Pr(D_k \geq t_2), \tag{6}$$

where $\Pr(D_k \geq t_2)$ is the probability that the computing time exceeds t_2. Note that D_k consists of two components W_k and D_k^{rest}, where W_k is a random variable that depends on the distribution of incoming offloaded tasks from other UEs in the MEC network. As the server follows the FCFS principle, the queue delay W_k is generally proportional to current queue length according to the little's law [32]. In fact, we consider a finite queue buffer size on the server side, where the corresponding maximal queue delay is way larger than $T_{\max}$. Incoming tasks are directly dropped, if they exceed the queue buffer (regardless from which UE). Therefore, $\varepsilon_{2,k}$ also includes the possibility that the offloaded task from UE of interest being dropped by server k. Meanwhile, D_k^{rest} is a deterministic parameter that is influenced by the previous actions. Obviously, $\varepsilon_{2,k}$ is monotonously decreasing function w.r.t. c_k/f_k and t_2, i.e., a loose threshold or a small portion of workload leads to a low computation delay violation probability. In addition, for a server k, the corresponding frequency f_k is fixed[2].

The distribution of the execution time is coupled to the distribution of the waiting time W_k. In particular, we have

$$\varepsilon_{2,k} = \Pr(D_k \geq t_2) = \Pr(W_k \geq t_2 - \frac{c_k}{f_k} - D_k^{\text{rest}}). \tag{7}$$

For the sake of simplicity, we denote the modified delay tolerance in the computation phase by $\hat{t}_2 = \max\{t_2 - \frac{c_k}{f_k} - D_k^{\text{rest}}, 0\}$. It should be pointed out that W_k is influenced by the traffic condition of the system, e.g., the task arrivals of served UEs, and can be modelled by a certain task arriving process. Without loss of generality, we consider W_k follows a general random distribution with an unbounded upper tail.

Due to the stringent reliability requirement, the system demands an extreme low computation error (delay violation) probability, i.e., the complementary cumulative distribution function (CCDF) of the queuing time satisfies $\bar{F}_{W_k}(\hat{t}_2) = \varepsilon_{2,k} = \Pr(W_k \geq \hat{t}_2) \ll 1$. In other words, the intervall of monotonically increasing CCDF with a sufficiently high $\hat{t}_2$, i.e., the tail performance of the CCDF, is with the high interests in the design of such reliable MEC network. Following the extreme value theory (EVT) model in [12], [28],we charaterize the tail of the probability distribution of $\varepsilon_{2,k}$ as follows. Let $X_k = \max\{\hat{t}_2 - d, 0\}$ denotes the exceedance of delay tolerance and we consider the distribution of D_k conditionally exceeding a high threshold d. According to [28], if the threshold d closely approaches $F_{W_k}^{-1}(1)$, the conditional CDF of the exceedance X_k can be expressed as

$$\begin{aligned} F_{X_k|D_k>d}(x_k) &= \Pr(W_k - d \leq x_k | W_k > d) \\ &\approx G(x_k; \sigma, \xi) \\ &= \begin{cases} e^{-x_k/\sigma}, \text{ if } \xi = 0, \\ 1 - \left(1 + \frac{\xi x_k}{\sigma}\right)^{-\frac{1}{\xi}}, \text{ otherwise} \end{cases} \end{aligned} \tag{8}$$

where $G(x; \sigma, \xi)$ is the generalized Pareto distribution (GPD) characterized by the scale parameter $\sigma > 0$ and shape parameter ξ. In particular, the value of ξ influences the tail behaviors. In this work, we only consider the cases of $\xi > -1/2$, which implies that the CCDF has finite upper endpoint [12]. In fact, the frame duration is generally insufficient to completely prevent the occurrence of extreme events in an URLLC scenario. Therefore, there is always a possibility that queue delay in the server is too long to be handled within t_2 regardless of actual queue buffer size.

Hence, the error probability with given time slot of computation phase t_2 in the ultra-reliable scenario with the threshold d is given by

$$\begin{aligned} \varepsilon_{2,k} = &(1 - F_{W_k}(d)) \\ &\left(1 - G(\max\{t_2 - \frac{c_k}{f_k} - D_k^{\text{rest}} - d, 0\}; \sigma, \xi)\right), \end{aligned} \tag{9}$$

where σ and ξ are parameters that are depends on the character of the computing task arrival model, as well as the computation power of the server k. Therefore, we can obtain them with sufficient historical data. More importantly, the validity of the expression does not depend on such specific task distribution model [12]. Equation (9) actually indicates the influence of traffic condition, e.g., task arriving rate, number of UEs in the coverage area, etc., on both $G(\cdot)$ and d. In particular, $\varepsilon_{2,k}$ is monotonically increasing in d. Hence, for any task arriving processes with a dense traffic, the value of d becomes large, i.e., approaching $F_{W_k}^{-1}(1)$, which leads to a higher computation error probability.

C. End-to-End Error Probability

Note that the system may select multiple servers, while the total workload is assigned among them to be computed simultaneously. The whole service in the current frame is considered as successful only if all parts of the corresponding selected servers are successful, i.e., no error occurs during communication and computation at each server link. Let ε_k denotes the error probability of the link k, i.e.,the probability that the decoding at server k side fails or the computation delay exceeds t_2. Furthermore, if server k is not selected, such link has no impact to the error probability, i.e., $\varepsilon_k = 0, \forall k \notin \hat{\mathcal{K}}$. Hence, for all $k \in \mathcal{K}$, ε_k can be written as

$$\varepsilon_k = a_k(\varepsilon_{1,k} + (1 - \varepsilon_{1,k})\varepsilon_{2,k}) = a_k(\varepsilon_{1,k} + \varepsilon_{2,k} - \varepsilon_{1,k}\varepsilon_{2,k}). \tag{10}$$

[2]Although the server may adopt the dynamic frequency and voltage scaling (DVFS) technique frame-wise, we assume the CPU frequency during the single frame is fixed.

Let $\varepsilon_{\rm O}$ denotes the end-to-end error probability over all the selected servers. Then, $\varepsilon_{\rm O}$ is given by

$$\varepsilon_{\rm O} = 1 - \prod_k (1-\varepsilon_k). \tag{11}$$

IV. Optimal Framework Design and Workload Assignment For Time-independent System With Prefect CSI

We start with the simplified time-independent system under the assumption that we have perfect knowledge of channels and the server statuses are not influenced by previous actions, i.e., h_k is known while $\rho = 0$ and $D_k^{\rm rest} = 0$. Hence, the system can be optimized frame-wise. Furthermore, we consider the offloading task follows data-partition model, where it can be divided into arbitrarily small sub-tasks and executed at different MEC servers [29]. In particular, we propose a framework optimization design to minimize the end-to-end error probability by optimally allocating the sum (of communication and computation) time T to the two phases t_1 and t_2 as well as the assigned workload c_k, while optimally determining the offloading decisions $\mathbf{A}$ with consideration of both the task-partition and the server selection in each frame.

A. Problem Statement

We aim to minimize $\varepsilon_{\rm O}$ by optimally allocating the maximal allowed T, denoted by $T_{\max}$, to t_1 and t_2, and optimally selecting multiple severs. In addition, we also determine the optimal workload assignment c_k to each selected server k. Hence, the optimization problem is formulated by

$$\begin{aligned}
&\underset{t_1,t_2,\mathbf{C},\mathbf{A}}{\text{minimize}} && \varepsilon_{\rm O} && \text{(12a)}\\
&\text{subject to} && t_1 + t_2 \le T_{\max}, && \text{(12b)}\\
& && \mathbf{A} \in \{0,1\}^K, && \text{(12c)}\\
& && \varepsilon_k \le \varepsilon_{\max}, \quad \forall k \in \mathcal{K}, && \text{(12d)}\\
& && a_k \le c_k \le a_k c_O, \forall k \in \mathcal{K}, && \text{(12e)}\\
& && \sum\nolimits_{k=1}^{K} c_k = c_O, && \text{(12f)}\\
& && \sum\nolimits_{k=1}^{K} a_k \ge 1, && \text{(12g)}
\end{aligned}$$

where constraint (12b) limits the operation within the maximal duration of one frame $T_{\max}$. The constraint (12d) restricts the error probability of the selected links lower then given threshold, to prevent waste of network resource. In addition, the constraints (12e) - (12g) are the conditions of the workload assignment.

B. Optimal Solution to (12)

In this subsection, we handle Problem (12), where our methodology is briefly described as follows: Firstly, the original problem (12) will be decomposed into subproblems. Subsequently, the subproblems will be characterized, and the relationship between the optimal solutions of t_1 and t_2 will be investigated. Finally, following the characterization and investigation, we reformulate the subproblems, based which reformulate the original problem in (12) to be solvable.

1) Decomposition and subproblems of (12)*:* Note that in total K servers are available, thus there exists $2^K - 1$ possible combinations of selected servers $\hat{\mathcal{K}}$. Hence, we could decompose the original problem in (12) into $2^K - 1$ subproblems corresponding to different $\hat{\mathcal{K}}$. For each set/combination $\hat{\mathcal{K}}$, the subproblem is given by

$$\begin{aligned}
&\underset{t_1,t_2,\mathbf{C}}{\text{minimize}} && \varepsilon_{\rm O} && \text{(13a)}\\
&\text{subject to} && a_k = 1, \quad \forall k \in \hat{\mathcal{K}}, && \text{(13b)}\\
& && a_k = 0, \quad \forall k \in \mathcal{K} \setminus \hat{\mathcal{K}}, && \text{(13c)}\\
& && \text{(12d), (12e) and (12f)} && \text{(13d)}
\end{aligned}$$

2) Characterization of Subproblem (13)*:* First, we fix the task assignment by letting $c_k = c_k^{\circ}$ and investigate the subproblem

$$\begin{aligned}
&\underset{t_1,t_2}{\text{minimize}} && \varepsilon_{\rm O} && \text{(14a)}\\
&\text{subject to} && c_k = c_k^{\circ}, \quad \forall k \in \mathcal{K}, && \text{(14b)}\\
& && \text{(13b), (13c) and (13d)} &&
\end{aligned}$$

In order to characterize Subproblem (14), we provide three lemmas as follows.

Lemma 1. *$\varepsilon_{\rm O}$ is convex in both t_1 and t_2.*

Proof: First, we discuss the convexity of ε_k. For a non-selected server $k \notin \hat{\mathcal{K}}$, $\varepsilon_k = 1$ holds, which is clearly convex (constant) in either t_1 or t_2. Then, for a selected server $k \in \hat{\mathcal{K}}$, the second derivative to t_1 is

$$\begin{aligned}
\frac{\partial^2 \varepsilon_k}{\partial t_1^2} &= \frac{\partial^2 \varepsilon_{1,k}}{\partial t_1^2} + 0 - \frac{\partial^2 \varepsilon_{1,k}}{\partial t_1^2}\varepsilon_{2,k}\\
&= \frac{\partial^2 \varepsilon_{1,k}}{\partial n_1^2}\left(\frac{\partial n_1}{\partial t_1}\right)^2 (1-\varepsilon_{2,k})\\
&= \frac{\partial^2 \varepsilon_{1,k}}{\partial n_1^2}\frac{(1-\varepsilon_{2,k})}{T_{\rm S}^2}.
\end{aligned} \tag{15}$$

It is proven in [33] that $\frac{\partial^2 \varepsilon_{1,k}}{\partial n_1^2} \ge 0$ holds. In addition, we have the computation error probability lesser than 1, i.e., $\varepsilon_{2,k} \le 1$. Therefore, we have $\frac{\partial^2 \varepsilon_k}{\partial t_1^2} \ge 0$. In other words, ε_k is convex in t_1.

Similarly, the second derivative of ε_k to t_2 is

$$\frac{\partial^2 \varepsilon_k}{\partial t_2^2} = \frac{\partial^2 \varepsilon_{2,k}}{\partial t_2^2} + 0 - \frac{\partial^2 \varepsilon_{2,k}}{\partial t_2^2}\varepsilon_{1,k} = \frac{\partial^2 \varepsilon_{2,k}}{\partial t_2^2}(1-\varepsilon_{1,k}). \tag{16}$$

Considering an ultra-reliable scenario, where the system only selects the server when the error probability of the link ε_k fulfills a given threshold, e.g., $\varepsilon_{\max} < 0.01$, the error probability in both communication and computation phases must also fulfills such threshold as shown in (11). Hence, it is reasonable to assume that we have a sufficient t_2 so that the EVT can be applied for the selected server, i.e., $t_2 > d$. According to (8), the second order derivative of $\varepsilon_{2,k}$ w.r.t. t_2

is given by

$$\frac{\partial^2 \varepsilon_{2,k}}{\partial t_2^2} = F_{D_k}(d)\frac{\partial^2 G(t_2 - \frac{c_k}{f_k}d_k;\sigma,\xi)}{\partial t_2^2} = F_{D_k}(d)\frac{(1+\xi)}{\sigma^2}\cdot\left(1-G(t_2-\frac{c_k}{f_k}-d;\sigma,\xi)\right)^{-\frac{2+\xi}{\xi}} \geq 0. \tag{17}$$

As $1-\varepsilon_{1,k} \geq 0$ holds, we have $\frac{\partial^2 \varepsilon_{2,k}}{\partial t_2^2} \geq 0$.

Subsequently, we further investigate the convexity of ε_{O}. Let $v_k = 1-\varepsilon_k$ denote the reliability of server k for the convenience of notations. Clearly, it holds $v_k = 1$, if $a_k = 0$. It implies that, for a non-selected server, the offloading between such server and the UE is always considered as reliable with "empty" input data. Then, the second order derivative of ε_{O} to t_1 can be written as

$$\frac{\partial^2 \varepsilon_{\mathrm{O}}}{\partial t_1^2} = -\sum_k \frac{\partial^2 v_k}{\partial t_1^2}\prod_{l\neq k} v_l + \sum_k\sum_{l\neq k}\frac{\partial v_k}{\partial t_1}\frac{\partial v_l}{\partial t_1}\prod_{p\neq k, p\neq l} v_p. \tag{18}$$

According to (15), $\varepsilon_k \leq 1$ is a monotonic and convex function w.r.t. t_1. $v_k = 1-\varepsilon_k \geq 0$ is therefore monotonic and concave. It holds that $\frac{\partial^2 v_k}{\partial t_1^2} \geq 0, \forall a_k = 1$ and $\frac{\partial^2 v_k}{\partial t_1^2} = 0, \forall a_k = 0$. Furthermore, it also hods that $\mathrm{sgn}\left(\frac{\partial v_k}{\partial t_1}\right) = \mathrm{sgn}\left(\frac{\partial v_l}{\partial t_1}\right), \forall a_k = a_l = 1$ and $\frac{\partial v_k}{\partial t_1} = 0, \forall a_k = 0$, where $\mathrm{sgn}(\cdot)$ is the sign function. Hence, we have $\frac{\partial^2 \varepsilon_{\mathrm{O}}}{\partial t_1^2} \geq 0$. Analog to t_1, we have that $\frac{\partial^2 \varepsilon_{\mathrm{O}}}{\partial t_2^2} \geq 0$ since ε_k is convex and monotonic in t_2 according to (16).

As a result, ε_k is convex in both t_1 and t_2. ■

Since both communication and computation phases are restricted within the frame duration T, there exists clearly a trade-off between the two phases. Therefore, we have the following lemma to characterize the relationship between the optimal solutions of t_1 and t_2.

Lemma 2. *Denote by t_1^* and t_2^* the optimal solutions to Problem* (13)*. Then, $t_1^* + t_2^* = T_{\max}$ holds.*

Proof: The lemma can be proved with contradiction in the following way. First, we assume the optimal solution t_1' and t_2' satisfying the strict constraint given in (12b), i.e., $T_{\max} - (t_1' + t_2') = \alpha > 0$. As the solution is optimal, $\varepsilon_{\mathrm{O}}'(t_1', t_2')$ is definitely the global minimum. Then, we have $\varepsilon_{\mathrm{O}}'(t_1', t_2') \geq \varepsilon_{\mathrm{O}}(t_1, t_2)$.

On the other hand, there exists a feasible solution $(t_1'' = t_1' + \alpha, t_2'' = t_2') \in \{t_1, t_2 | t_1 + t_2 \leq T_{\max}\}$. Recall that it has been shown in the proof of lemma 1 that ε_{O} is decreasing in blocklength $n = \frac{t_1}{T_{\mathrm{s}}}$ and in t_1. Hence, it can be concluded that the (t_1'', t_2'') lead to a lower error probability in comparison to (t_1', t_2'), i.e., $\varepsilon_{\mathrm{O}}''(t_1'', t_2'') < \varepsilon_{\mathrm{O}}'(t_1', t_2')$. Therefore, the assumption of the optimal solution (t_1', t_2') is violated. ■

Thus, we can replace the inequality constraint (12b) with equality constraint, i.e., $t_1 + t_2 = T_{\max}$. Consequently, the optimization variables of t_1 and t_2 are reduced to a single variable, e.g., t_1. Then, we characterize the problem by jointly optimizing t_1 and c_k, resulting in the following problem

$$\begin{aligned} \underset{t_1,\mathbf{C}}{\text{minimize}} \quad & \varepsilon_{\mathrm{O}} && \text{(19a)}\\ \text{subject to} \quad & t_1 + t_2 = T_{\max}, && \text{(19b)}\\ & \text{(12d), (12g) and (12e).} && \text{(19c)} \end{aligned}$$

To handle this problem, we have following lemma.

Lemma 3. *The end-to-end error probability ε_{O} is jointly convex in t_1 and $\mathbf{C}$, if it holds $\varepsilon_k \leq \varepsilon_{\max} \ll 1$, $\forall k \in \mathcal{K}$.*

Proof: If the error probability of each link k is sufficiently small, i.e., $\varepsilon_k \ll 1$, the end-to-end error probability can be approximated as

$$\varepsilon_{\mathrm{O}} = 1 - \prod(1-\varepsilon_k) \approx \sum\nolimits_k \varepsilon_k. \tag{20}$$

Furthermore, it implies that both error probability in computation and communication phase of the link k must also smaller than $\varepsilon_{\max}$ according to (10), i.e., the error probability in link k can also be approximated as $\varepsilon_k \approx \varepsilon_{1,k} + \varepsilon_{2,k}$.

To determine the joint convexity of ε_{O}, we first investigate the convexity of arbitrary ε_k, the Hessian matrix of which with respect to t_1 and $\mathbf{C}$ can be written as

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 \varepsilon_{1,k}}{\partial t_1^2} + \frac{\partial^2 \varepsilon_{2,k}}{\partial t_1^2} & \frac{\partial^2 \varepsilon_{2,k}}{\partial t_1 \partial c_k} \\ \frac{\partial^2 \varepsilon_{2,k}}{\partial t_1 \partial c_k} & \frac{\partial^2 \varepsilon_{2,k}}{\partial c_k^2} \end{bmatrix}. \tag{21}$$

After some manipulations according to Lemma 1 and (9), the determinate of $\mathbf{H}$ is given by

$$\det \mathbf{H} = \frac{\partial^2 \varepsilon_{1,k}}{\partial t_1^2}\frac{\partial^2 \varepsilon_{2,k}}{\partial c_k^2} \tag{22}$$

where $\frac{\partial^2 \varepsilon_{1,k}}{\partial n_1^2}\frac{1}{T_{\mathrm{s}}^2} \geq 0$ and $\frac{\partial^2 \varepsilon_{2,k}}{\partial c_k^2} = \frac{\partial^2 \varepsilon_{2,k}}{\partial t_2^2}\frac{1}{f_k^2} \geq 0$. Hence, it holds $\det \mathbf{H} \geq 0$. As a result, ε_k is jointly convex over t_1 and $\mathbf{C}$.

Since the error probability of each link k is jointly convex and the end-to-end error probability is the sum of error probabilities in every link, the end-to-end error probability is also jointly convex over t_1 and $\mathbf{C}$. ■

3) Reformulation of Problem (12)*:* According to the above lemmas characterizing the subproblem (13), the original problem given in (12) could be reformulated to

$$\begin{aligned} \underset{t_1,\mathbf{C},\mathbf{A}}{\text{minimize}} \quad & \varepsilon_{\mathrm{O}} = 1 - \prod_k (1-\varepsilon_k) && \text{(23a)}\\ \text{subject to} \quad & t_1 + t_2 = T_{\max}, && \text{(23b)}\\ & \varepsilon_k \leq a_k, \quad \forall k \in \mathcal{K}, && \text{(23c)}\\ & a_k(\varepsilon_{1,k} + \varepsilon_{2,k} - \varepsilon_{1,k}\varepsilon_{2,k}) \leq \varepsilon_k, \forall k \in \mathcal{K}, && \text{(23d)}\\ & \text{(12c) to (12g).} && \text{(23e)} \end{aligned}$$

where constraints (23c) and (23d) help us to eliminate the the multiplication of variable a_k and t_1 by linearizing the objective function. We can easily show that both the objective function and the constraints in Problem (23) are either convex or affine. Therefore, the optimization problem is a mixed integer convex problem (MICP), which can be efficiently solved by the recently developed algorithm in [34]. In particular, which results in a computational complexity

level of $\mathcal{O}(2^K-1)$, since it is essential to solve 2^K-1 standard convex problems and select the minimal value. By letting $\mathcal{A}=\{\mathbf{A}_1,...,\mathbf{A}_{2^K}\}$ denote the set of all possible combinations of $\mathbf{A}$, where $\mathbf{A}_i\in\{0,1\}^K$, the pseudocode of the algorithm to solve problem (12) is summarized in Algorithm 1.

Algorithm 1 for solving Problem (23) as MICP

for $i=1,...,2^K$ **do**
  Initilziation: $\varepsilon^*\leftarrow 1$
  Choose the i-th possible combination: $\mathbf{A}\leftarrow\mathbf{A}_i$
  Solve Problem (19) according to Lemma 3: $\varepsilon^*_{\mathrm{O},(i)}=\min\limits_{t_1,\mathbf{C}}\varepsilon_{\mathrm{O}}$ and $\left(t^*_{1,(i)},\mathbf{C}^*_{(i)}\right)=\arg\min\limits_{t_1,\mathbf{C}}\varepsilon_{\mathrm{O}}$
  Update the global optimum: $\varepsilon^*\leftarrow\min\{\varepsilon^*,\varepsilon^*_{\mathrm{O},(i)}\}$
end for
Obtain optimal solutions: $(t_1^*,\mathbf{C}^*)=\arg\min\limits_{t_1,\mathbf{C}}\varepsilon^*_{\mathrm{O}}$
Obtain solution of t_2: $t_2^*\leftarrow T-t_1^*$

C. Low-complexity Server Selection and Task Partition Schemes

According to the results in previous subsections, we are able to obtain the optimal solution of the subproblem with given server selection in $\hat{\mathcal{K}}$, the optimal server selection can be determined. However, the complexity of the process is significant, i.e., we have to calculate and compare all optimal results of 2^K-1 serve selection combinations. In particular, the computational complexity level of the process is $\mathcal{O}(2^K-1)$. In addition, in the model of previous subsections, tasks are treated/assumed to be arbitrarily dividable, which is too optimistic in a practical system.

To reduce the complexity and consider a more realistic partition model (the task is no longer arbitrarily dividable), in the following we exploit the KKT conditions [38] to provide an efficient method for both server selection and workload assignment. In particular, under this partition model, we propose an efficient algorithm to obtain the optimal workload assignment instead of exhaustive search.

1) Server Selection based on KKT conditions: Similar to the design in Section IV-B, the objective is to minimize the end-to-end error probability, i.e., the objective function remains the same as (19). In particular, we achieve this objective by optimizing the workload assignment and the server selection with a fixed frame structure and CPU-frequency, i.e., $t_1=t_1^\circ\le T$ and $f_k=f$. It is worthwhile to mention that the optimization problem is considered for all possible frame structures instead of targeting at any specific setup. Hence, we have

$$\underset{\mathbf{C},\mathbf{A}}{\text{minimize}}\quad \varepsilon_{\mathrm{O}} \tag{24a}$$
$$\text{subject to}\quad t_1=t^\circ, \tag{24b}$$
$$\text{(23b) to (23d).} \tag{24c}$$

Next, we establish the following lemma by exploiting the KKT conditions to solve the above problems.

Lemma 4. *Given the duration of the first slot t_1 and the duration of the second slot t_2, the optimal solution of workloads c_k^* for the offloaded task with the selected servers in set of $\hat{\mathcal{K}}$ satisfy*

$$\varepsilon_{2,k}(c_k^*)=\max\{\nu-\varepsilon_{1,k},0\} \tag{25}$$

where $0\le\nu\le 1$, such that it holds

$$\sum c_k=c_{\mathrm{O}}. \tag{26}$$

Proof: First, we have $c_k=0,\forall k\notin\mathcal{K}$ and the corresponding $\varepsilon_k=0$ can be eliminated from ε_{O}. We denote $\nu\ge 0$ and $\lambda_{\hat{k}}\ge 0,\forall k\in\hat{\mathcal{K}}$, the corresponding dual variables for the constraint (12f) and constraint (12d), respectively. The partial Lagrangian of problem (24) can therefore be given by

$$L=1-\prod(1-\varepsilon_k)+\nu(\sum_k c_k-c_o)+\sum\lambda_k(\varepsilon_k-\varepsilon_{\max}). \tag{27}$$

Based on the KKT conditions, the optimal solution of ν^*, λ_k^* and c_k^* must satisfy the following necessary and sufficient conditions:

$$\lambda_k^*\ge 0,\forall k\in\hat{\mathcal{K}}, \tag{28a}$$
$$\lambda_k^*(\varepsilon_k^*-\varepsilon_{\max})=0,\forall k\in\hat{\mathcal{K}}, \tag{28b}$$
$$\frac{\partial L}{\partial c_k}=\frac{\partial\varepsilon_k}{\partial c_k}\prod_{l\neq k}(1-\varepsilon_l)+\nu^*+\lambda_k\frac{\partial\varepsilon_k}{\partial c_k}=0,k\in\hat{\mathcal{K}}. \tag{28c}$$

By eliminating λ_k^*, we reformulate conditions in (28b) as

$$\frac{\partial\varepsilon_k}{\partial c_k}\frac{1}{1-\varepsilon_k}=\frac{\nu^*}{\prod(1-\varepsilon_k)},\forall k\in\hat{\mathcal{K}}. \tag{29}$$

The equality of ν^* holds $\forall k\in\hat{\mathcal{K}}$. Therefore, we have

$$\frac{\partial\varepsilon_1}{\partial c_1}\frac{1}{1-\varepsilon_1}=...=\frac{\partial\varepsilon_k}{\partial c_k}\frac{1}{1-\varepsilon_k}. \tag{30}$$

Since ε_k is a monotonically increasing function with respect to c_k according to (10) and (6), above equality chain holds and only holds if

$$\varepsilon_1(c_1^*)=...=\varepsilon_k(c_k^*)=\nu. \tag{31}$$

where ν is the positive value, so that it holds both $\varepsilon_{2,k}(c_k^*)=\max\{\nu-\varepsilon_{1,k},0\}$, and $\sum c_k=c_O$. ■

Remark. Lemma 4 provides the optimal workload assignment. This can be intuitively interpreted in the following way distinguishing if the system is homogeneous or not: For a homogeneous system, i.e., $h_k=h_l$ and $f_k=f_l$ for all $k,l\in\mathcal{K}$, the optimal solutions are achieved with homogeneous distributed workload $c_k^*=c_l^*=\frac{c_O}{\sum_{k=1}^K a_k}$ regardless of frame structure. In this case, all servers in $\hat{\mathcal{K}}$ should be selected. On the other hand, for a heterogeneous system, the optimal solution of this problem can be considered as a non-linear (inverse) waterfilling application, where $\varepsilon_{1,k}$ and ν can be seen as the ground level above patch k and the depth of the waterflood for the region (all servers), respectively. The total amount of water filled into each servers is then $\varepsilon_{2,k}(c_k^*)$. We increase the flood level by feeding the server k with increment Δc_k until we have used all amount of water c_O. Then, the workload c_k causing the increment depth of water above patch k is then the optimal value. In addition, any server that above the waterflood ν should be deselected.

According to Lemma 4, instead of going through all possible server selection combinations as the design in Section IV-B, here we reduce the total number of server selection combinations in the following steps: We firstly sort the servers

in descending order of $\varepsilon_{1,k}$. Subsequently, we build K server selection combinations $\mathcal{K}_k = \{1, ..., k\}$ from $\mathcal{K}$. For each subset, we let $a_l = 1$, where $l \in \mathcal{K}_k$. For example, for $\mathcal{K}_1 = \{1\}$, we set $a_1 = 1$. Meanwhile, for $\mathcal{K}_2 = \{1, 2\}$, we set $a_1 = 1$ and $a_2 = 1$. Then, we solve subproblems expressed in (19) for all K subsets and obtain the optimal solution denoted by $\mathbf{C}^{(k)}_{\text{KKT}}$ and $t^{(k)}_{1,\text{KKT}}$ and the optimal value denoted by $\varepsilon^{(k)}_{\text{O,KKT}}$. On the last step, we choose the minimal $\varepsilon^{(k)}_{\text{O,KKT}}$ as the final/overall optimal value, i.e., $\varepsilon^* = \min\{\varepsilon^{(k)}_{\text{O,KKT}}\}$ and the corresponding $\mathbf{C}^{(k)}_{\text{KKT}}$, $t^{(k)}_{1,\text{KKT}}$ are the optimal workload assignment and optimal frame structure, respectively. In this way, we reduce the complexity from $\mathcal{O}(2^K - 1)$ to $\mathcal{O}(K)$.

2) Integer Task Partition Model: Instead of simple data-partition model, which assumes that the task can be bit-wise independently and arbitrarily partitioned into different sub-tasks, we consider a more realistic partition model without losing the generality, where the partition is pre-defined by the character of the task with finite slices. In particular, we denote by Δc the smallest amount of workload of each sub-task. Hence, let $\Omega = \frac{c_O}{\Delta c}$ present the total slices of the tasks and ω_k the number of slices that belong the assigned sub-task for server k, we have $c_k = \omega_k \Delta c$. The elements of set $\mathbf{C}$ can be mapped with element of set $\mathbf{\Omega}$, where $\mathbf{\Omega} = \{\omega_1, ..., \omega_k\}$ is the set of the number of assigned workload slices. Specially, if $\Delta c = 1$ bit, our model coincides to the data-partition model. If $\Delta c = c_O$, the partition model degrades into binary offloading model.

To adapt the new partition model, we replace the constraints and reformulate the problem as

$$\underset{\mathbf{C},\mathbf{A}}{\text{minimize}} \quad \varepsilon_{\text{O}} \tag{32a}$$

$$\text{subject to} \quad c_k = \omega_k \Delta c, \qquad \forall k \in \mathcal{K}, \tag{32b}$$

$$\omega_k \in \mathbb{Z}_+, \tag{32c}$$

$$\text{(24b) and (24c).} \tag{32d}$$

However, the challenge to exploit the lemma 4 is how to find an optimal ν^*. Unlike the classical waterfilling problem, the optimal solution in (25) consists of the computational error probability $\varepsilon_{2,k}$ with respect to assigned non-linear workload c_k. Furthermore, instead of allocating the limited resource to nodes, our problem is to assign all the workload c_{O} to the servers. Therefore, the previous geometrical approach [35] to solve this problem cannot be applied for our case. To this end, we propose following algorithm to find the optimal ν^*:

1) Let all servers be assigned with no workload initially, i.e., $c_k = 0$, $\forall k \in \mathcal{K}$. Sort the servers by ascending order of error probability ε_k. They are essentially sorted based on the decoding error probability $\varepsilon_{1,k}$. Let all workloads be assigned to the first server, i.e., $c_{1,\text{temp}} = c_{\text{O}}$ and $c^{k,\text{temp}}_k = 0$. Moreover, ν is defined by the maximum from the sum of $\varepsilon_{1,k}$ and $\varepsilon_{2,k}$, i.e., $\nu = \max_k\{\varepsilon_{1,k} + \varepsilon_{2,k}\}$, where $k \in \hat{\mathcal{K}}$. Any server k has a lower error probability than ν should be selected, i.e., $a_k = 1$ if $\varepsilon_{1,k} + \varepsilon_{2,k} \leq \nu$.
2) Move a slice of workload with the size of Δc from the first server to the second, i.e., $c_{1,\text{temp}} = c_{1,\text{temp}} - \Delta c$ and $c_{2,\text{temp}} = c_{2,\text{temp}} + \Delta c$. Let $\nu_{\text{temp}} = \max_k\{\varepsilon_{1,k} + \varepsilon_{2,k}\}$. If $\nu_{\text{temp}} > \nu$, move the slice to the next server, i.e., $c_{2,\text{temp}} = c_{2,\text{temp}} - \Delta c$ and $c_{3,\text{temp}} = c_{3,\text{temp}} + \Delta c$. Repeat the process until the last element of $\hat{\mathcal{K}}$ or $\nu_{\text{temp}} \leq \nu$.
3) Let $\nu^*_{\text{temp}} = \nu_{\text{temp}}$ and deselect all servers that have higher error probability than ν, i.e., let $a_k = 0$, if $\nu < \varepsilon_{1,k} + \varepsilon_{2,k}$. Resort the servers by ascending order of error probability ε_k.
4) Repeat 2) and 3) till ν cannot be reduced anymore. Then, let $\nu^* = \nu^*_{\text{temp}}$, $c^*_k = c_k$ and $a^*_k = a_k$.

With this algorithm, we are able to not only assign all the workload, but also select the corresponding servers. The complexity of the algorithm is relatively low. In particular, the worst case of the algorithm requires $MK!\log(MK!)$ times of operations and results in the complexity of $\mathcal{O}(MK!\log(MK!))$ which is significantly lower than $\mathcal{O}(2^{MK})$, i.e. the complexity of the design in Section IV-B with exhaustive search. The pesudocode of the algorithm is summarized in Algorithm 2.

Algorithm 2 for solving Problem (32) with integer task partition model

```
Initialize the workload assignment and server selection: c_{1,temp} ← 0, a_k ← 0, c_k^{k,temp} ← 0, v*_temp = 0 and v_temp ← 1
Sort ε_k in ascending order according to (10)
Assign all workload to the first server: c_{1,temp} ← c_O
Calculate the initial waterflood: ν = max_k{ε_{1,k} + ε_{2,k}}
while v*_temp < v do
    assign the temporal optimal waterflood: v ← v*_temp
    for k=1,...,K-1 do
        Move a slice of workload: c_{k,temp} ← c_{k,temp} − Δc and c_{k+1,temp} ← c_{k+2,temp} + Δc
        Calculate the temporal waterflood: ν_temp = max_k{ε_{1,k} + ε_{2,k}}
        if ν_temp ≤ ν || c_{k,temp} == 0 then
            Break
        end if
    end for
    Assign the temporal optimal waterflood: v*_temp ← v_temp
    Resort ε_k in ascending order according to (10)
end while
for k=1,...,K do
    if ν < ε_{1,k} + ε_{2,k} then
        Select the server: a_k ← 1
    end if
end for
Assign the optimal solution: v* ← v*_temp, c*_k ← c_k, a*_k ← a_k, ∀k ∈ K
```

V. Framework Design And Workload Assignment For Time-dependent System: A Reinforcement Learning Approach

Although we are able to provide an optimal solution analytically in the previous section with the assumption that the CSI is perfect known for every server and the queue length of server is independent to the previous actions with bit-wise partitionable tasks. In this section, we study the more realistic scenario, where only outdated CSI is available and the current queue length in the server is influenced by the assigned workload from previous frame. Denote m the frame

index, (1) and (5) can be respectively reformulated as

$$h_k^{(m)} = \rho h_k^{(m-1)} + \sqrt{1-\rho^2}\Delta h_k, \tag{33}$$

$$D_k^{(m)} = \frac{c_k^{(m)}}{f_k} + \max\left\{\frac{c_k^{(m-1)}}{f_k} - T_k^{\text{pre}}, 0\right\} + Q_k^{(m)}, \tag{34}$$

where $h_k^{(m-1)}$ is the known previous channel gain of server k at frame $m-1$, ρ is the coherent factor and Δh_k is an i.i.d. random variable. In addition, $c_k^{(m)}$ is the assigned workload to server k in frame m and T_k^{pre} is the time gap between previous computation phase and current computation phase. Although T_k^{pre} may vary based on the frame structure and task arriving rate, the discussion of randomness of T_k^{pre} is beyond the scope of this paper. In the rest of the section, we consider that it is constant over all frames. Furthermore, the task follows the integer task partition model we introduced in Sec. IV-C. Note that the error probability in the current frame is influenced by the actions in previous frames and the channel in current frame also correlated to the previous. Therefore, in such time-dependent case, it is not meaningful to provide a system design per frame, i.e., aiming at minimizing the instantaneous error probability. We thus focus on the average error probability by jointly selecting servers and allocating the blocklength and workload in the long-term. We denote by $\mathbf{b}^{(m)} = \{n^{(m)}, \mathbf{A}^{(m)}, \mathbf{C}^{(m)}\} \in \mathcal{B}$ the action of each frame and by $\mathbf{B} = \{\mathbf{b}^{(1)}, ..., \mathbf{b}^{(m)}, ...\}$ the actions of all frames, where $\mathcal{B}$ is the action space containing all possible combinations of the blocklength n, partitioned workload $\mathbf{C} = [c_1, ..., c_K]$ and server selection $\mathbf{A} = [a_1, ..., a_K]$. Then, we have the following optimization problem:

$$\underset{\mathbf{B}}{\text{minimize}} \quad \int_0^\infty \sum_{m=1}^{\infty} \varepsilon_{\text{O}}^{(m)} \prod_{k=1}^{K} f_{\Gamma_k^{(m)}}(\gamma_k^{(m)}) d\gamma^{(m)} \tag{35a}$$

$$\text{subject to} \quad (12b) - (12g),$$

where $f_{\Gamma_k^{(m)}}$ is the PDF of $\gamma_k^{(m)}$. Clearly, the proposed algorithm in the previous section can no longer be applied for the time-dependent scenario and to obtain the global optimal results requires prior channel knowledge. Therefore, an analytical solution is highly intractable. To overcome the challenge, we refer to a DRL approach aiming to minimize the overall error probability by jointly optimizing the cooperative server selection, frame structure design and workload assignment.

A. State and Action Space

In DRL algorithm, an agent is trained by interacting with the environment. At m-th time frame, the agent obtains the current observation of the environment denoted by $x^{(m)} = \{\boldsymbol{\gamma}^{(m)}, \mathbf{c}^{(m)}\}$, which is composed from all SNR and workloads of the previous frame, i.e.,

- $\boldsymbol{\gamma}^{(m)} = [\gamma_1^{(m-1)}, \gamma_2^{(m-1)}, \ldots, \gamma_K^{(m-1)}]$: vector of length K containing previous SNR of each link.
- $\mathbf{c}^{(m)} = [c_1^{(m-1)}, c_2^{(m-1)}, \ldots, c_K^{(m-1)}]$: vector of length K containing size of previous workload of each server.

Moreover, in order to exploit the impact of previous offloading decision on the computation error at the current time frame as well as the correlation of the channel states in adjacent frames, the state of the environment at frame m is defined as the set of W_{ht} historical observations, which is denoted as: $s^{(m)} = \{x^{(m)}, x^{(m-1)}, \ldots, x^{(m-W_{\text{ht}})}\} \in \mathcal{S}$, where $\mathcal{S}$ is the state space.

The action made by the agent is $b^{(m)} = \{n^{(m)}, \mathbf{A}^{(m)}, \mathbf{C}^{(m)}\} \in \mathcal{B}$. We consider a reasonable range of n as $[100, \frac{T}{T_s}]$ and the integer partitioned workload model $c_k = \omega_k \Delta c$ as explained in Section IV-C2. Thus, at m-th time frame the agent decides on action $b^{(m)}$ based on $s^{(m)}$ according to policy $\pi(s^{(m)})$, which leads to the next state $s^{(m+1)}$. The sequence of dynamic transitions $\mathcal{T}(s^{(m+1)}|s^{(m)}, b^{(m)})$ represents a Markov decision process.

B. Reward Function

In order to learn and update the decision policy, the agent requires a reward regarding its action. The reward should reflect the performance of the action in terms of the resulting error probabilities. Since the absolute value of error probabilities are normally quite small, it is more convenient to consider their order of magnitude. Therefore, we define the reward function depending on the error probability as: $r^{(m)}(\varepsilon_O) = -\log_{10}(\varepsilon_O^{(m)})$. In this way, the range of reward is restricted and the learning process is more stable, particularly when the channel varies dramatically. Furthermore, in the case that the action leads to an infeasible result, i.e., $\varepsilon_k \leq \varepsilon_{\max}, \forall k \in \mathcal{K}$ is not satisfied, the agent would obtain a minimum reward with $r^{(m)} = 0$.

The objective for the agent in each interaction with the environment is to optimize policy $\pi(s)$. Specifically, for a given state $s^{(m)}$, the policy aims at finding the action that returns the highest reward for future state-action processes. Thus, the optimal policy can be written as [40]

$$\pi^\star = \arg\max \mathbb{E}[R^{(m)}|\pi], \tag{36}$$

where $R^{(m)} = r^{(m)} + \gamma_{\text{df}} R^{(m+1)}$ is the accumulated discounted reward. $\gamma_{\text{df}} \in [0, 1]$ is the discount factor for weighting future rewards into the current policy.

C. Deep Deterministic Policy Gradient

We adopt an actor-critic model based on DDPG algorithm with an actor that makes decisions according to its actor function μ[3] and a critic that evaluates the action made by actor based on its Q-function Q. The actor function μ and the Q-function Q of the critic are approximated by DNNs as actor network with parameters θ_μ and critic network with parameters θ_Q, respectively.

The Q-value generated by the critic network represents the quality of an action $b^{(m)}$ for a certain state $s^{(m)}$ as $Q(s^{(m)}, b^{(m)}) = \mathbb{E}[R^{(m)}|s^{(m)}, b^{(m)}]$. With the target policy being deterministic and $\mu : \mathcal{S} \leftarrow \mathcal{B}$, the Q-value can be formulated using the Bellman equation as [37]

$$Q(s^{(m)}, b^{(m)}) = \mathbb{E}[r(s^{(m)}, b^{(m)}) + \gamma_{\text{df}} Q(s^{(m+1)}, \mu(s^{(m+1)}))]. \tag{37}$$

Since the Q function guarantees convergence to the optimal point with $Q^{(m)} \rightarrow Q^\star$ as $m \rightarrow \infty$ and with $\mathbb{E}[\nabla_b Q^{(m)}(s, b)|_{b=\mu(s)} \nabla_{\theta_\mu} \mu(s|\theta_\mu)]$ being the off-policy deterministic policy gradient, the critic can be used to adapt the

[3]Note that the actor function μ actually represents the policy function due to its deterministic property.

action function. Thus, the actor is learned by applying the chain rule to the expected return from the start distribution J with respect to the actor parameters θ_μ, which results in the policy gradient [21]

$$\nabla_{\theta_\mu} J \approx \mathbb{E}_{s^{(i)}}[\nabla_b Q(s,b|\theta_Q)|_{s=s^{(i)},b=\mu(s^{(i)})} \nabla_{\theta_\mu}\mu(s|\theta_\mu)|_{s=s^{(i)}}]. \tag{38}$$

On the other hand, as mentioned before the critic network maps the states and actions to a Q-value, which is similar to the Q-network in deep Q-learning [20]. The critic network updates its parameters θ_Q by minimizing the loss L as

$$L(\theta_Q) = \mathbb{E}_i[(y^{(i)} - Q(s^{(i)}, b^{(i)}|\theta_Q))^2], \tag{39}$$

where $y^{(i)} = \mathbb{E}[r^{(i)} + \gamma_{\mathrm{df}} Q'(s^{(i+1)}, \mu'(s^{(i+1)}|\theta_{\mu'})|\theta_{Q'})]$ is the target Q-value following the Bellman equation, and it is computed by the extra introduced target actor network $\mu'(s|\theta_{\mu'})$ and target critic network $Q'(s,b|\theta_{Q'})$ to avoid unstable training and local optimum [20]. The parameters of target networks $\theta_{\mu'}$ and $\theta_{Q'}$ are similar to θ_μ and θ_Q with a time delay and slowly updated with the update ratio $\tau_{\mathrm{c}} \ll 1$ as $\theta_{\mu'} \leftarrow \tau_{\mathrm{c}}\theta_\mu + (1-\tau_{\mathrm{c}})\theta_{\mu'}$ and $\theta_{Q'} \leftarrow \tau_{\mathrm{c}}\theta_Q + (1-\tau_{\mathrm{c}})\theta_{Q'}$.

Using the policy gradient $\nabla_{\theta_\mu} J$ and loss $L(\theta_Q)$ the parameters θ_μ and θ_Q are updated iteratively using the Stochastic Gradient Descent (SGD) as

$$\theta_\mu = \theta_\mu - \alpha_\mu \nabla_{\theta_\mu} J, \tag{40}$$

$$\theta_Q = \theta_Q - \alpha_Q \nabla_{\theta_Q} L, \tag{41}$$

where α_μ and α_Q are the learning rate of actor and critic networks, respectively.

In order to break the similarity of subsequent training samples and obtain a better convergence for function approximator, experience replay is used, where the transitions of $(s^{(m)}, b^{(m)}, r^{(m)}, s^{(m+1)})$ are stored in a replay memory of size U_{mem}. At each time slot a random minibatch of size U_{b} is sampled from the memory for performing a SGD update. By sampling from the large replay buffer the training at each frame is proceeded over a large set of uncorrelated state transitions [21]. Furthermore, to explore the environment with a continuous action space, noise is added to the decided action in the training phase. We adopt the Ornstein-Uhlenbeck process to produce correlated exploration [39]. Specifically, during training the action at m-th frame is then composed to $b^{(m)} = \mu(s^{(m)}|\theta_\mu^{(m)}) + \mathcal{N}^{(m)}$, where $\mathcal{N}^{(m)} = X^{(m)} + X'^{(m)}$ and $X^{(m+1)} = \mathcal{N}^{(m)}$. $X'^{(m)}$ follows Ornstein-Uhlenbeck process as

$$X'^{(m)} = \theta_{\mathrm{OU}}\left(\mu_{\mathrm{OU}} - X^{(m)}\right)\mathrm{dt} + \sigma_{\mathrm{OU}}\omega'^{(\mathrm{m})}, \tag{42}$$

where θ_{OU} denotes the mean reversion rate, μ_{OU} denotes the mean reversion level and σ_{OU} indicates the influence of the random variable $\omega'^{(m)}$.

The complete proposed DDPG algorithm is presented in Algorithm 3 and an illustration is shown in Figure 3 on the next page.

Algorithm 3 for Deep Deterministic Policy Gradient based Offloading

Randomly initialize actor network $\mu(s|\theta_\mu)$ and critic network $Q(s,b|\theta_Q)$ with weights θ_μ and θ_Q
Initialize target networks μ' and Q' with weights $\theta_{\mu'} \leftarrow \theta_\mu$, $\theta_{Q'} \leftarrow \theta_Q$
Initialize replay memory D_{mem} with size U_{mem}
Initialize a random noise $\mathcal{N}^{(1)}$
Receive initial observation state $s^{(1)}$
for $m = 1, 2, 3, \ldots$ **do**
  Select action $b^{(m)} = \mu(s^{(m)}|\theta^\mu) + \mathcal{N}^{(m)}$
  Execute action $b^{(m)}$ and obtain reward $r^{(m)}$ and observe new state $s^{(m+1)}$
  Store transition $(s^{(m)}, b^{(m)}, r^{(m)}, s^{(m+1)})$
  Sample a random minibatch of U_{b} transitions $(s^{(i)}, a^{(i)}, r^{(i)}, s^{(i+1)})$
  Set $y^{(i)} = r^{(i)} + \gamma_{\mathrm{df}} Q'(s^{(i+1)}, \mu'(s^{(i+1)}|\theta_{\mu'})|\theta_{Q'})$
  Compute the loss and update via SGD:

$$L(\theta_Q) = \frac{1}{U_b}\sum_i (y^{(i)} - Q(s^{(i)}, b^{(i)}|\theta_Q))^2$$
$$\theta_Q = \theta_Q - \alpha_Q \nabla_{\theta_Q} L(\theta_Q)$$

  Compute the sampled policy gradient and update via SGD:

$$\nabla_{\theta\mu} J \approx \frac{1}{U_{\mathrm{b}}}\sum_i \nabla_b Q(s,b|\theta_Q)|_{s=s^{(i)},b=\mu(s^{(i)})} \cdot \nabla_{\theta\mu}\mu(s|\theta_\mu)|_{s=s^{(i)}} \tag{43}$$
$$\theta_\mu = \theta_\mu - \alpha_\mu \nabla_{\theta_\mu} J$$

  Update the target networks:

$$\theta_{Q'} \leftarrow \tau_{\mathrm{c}}\theta_Q + (1-\tau_{\mathrm{c}})\theta_{Q'}$$
$$\theta_{\mu'} \leftarrow \tau_{\mathrm{c}}\theta_\mu + (1-\tau_{\mathrm{c}})\theta_{\mu'}$$

end for

VI. Simulation Results

A. Parameterization

In this section, We evaluate our proposed design via the numerical simulations. The following parameter setups are considered in the simulation. First, consider the UE is located in the center of an area with radius of 50 m. To describe the wireless channel, we set carrier frequency $F = 2.4$ GHz, thermal noise power $N_0 = -174$ dBm/Hz and transmit power $P = 20$ dBm. Furthermore, we consider the path-loss follows the model in [36], which can be written as $PL = 17.0 + 40.0\log_{10}(d_{\mathrm{server}})$ for $F = 2.4$ GHz, where d_{server} is the distance between UE and server. The duration of symbol is set as $T_{\mathrm{s}} = 0.025$ ms. The input data is set as $\tau = 1600$ bits and the total workloads $c_{\mathrm{O}} = 24$ Mcycles. we set the CPU-frequency of each servers as $f = 3$ GHz, while the task follows Poisson arriving model with rate $\lambda = 3$ Mcycles/s. Therefore, the shape parameter $\xi = -0.0214$ and scale parameter $\sigma = 3.4955 \times 10^6$ with a threshold $d = 5.7$ms above 99.9% reliability are obtained. Finally, we set the total delay tolerance as $T = 25$ms and the maximal error probability threshold as $\varepsilon_{\max} = 0.01$. In the following, we start with evaluations of the proposed designs by means of simulations under the above parameterization.

B. Performance of System with Time-Independent Servers and Perfect CSI

We first evaluate the system performance of the scenario with perfect knowledge of CSI and MEC server status. We start with Fig. 4 to present the error probability of a selected link ε_1 versus the duration of time slot t_1 under uniformed distance d_{server} of 20m and 50m, as well as assigned workload c_1 of $c_{\mathrm{O}}/2$ and $c_{\mathrm{O}}/4$. First , we can observe that the error

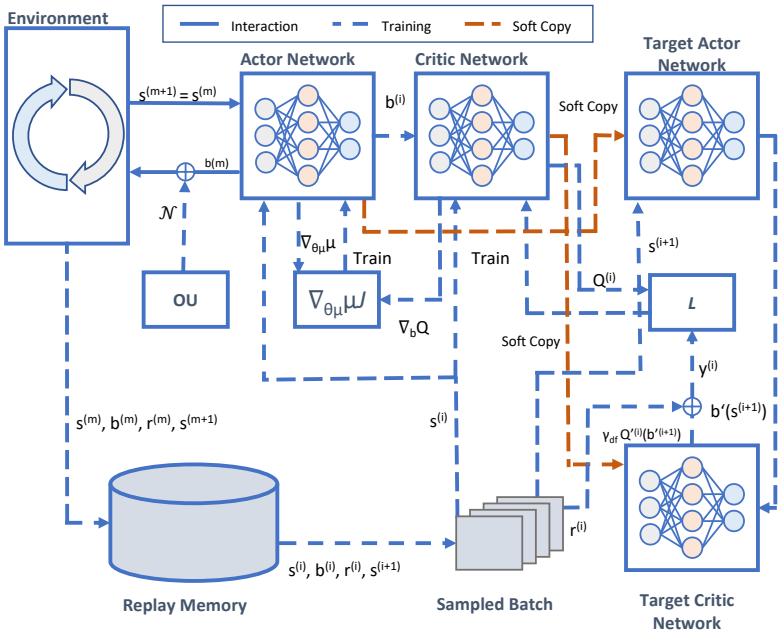


Fig. 3: Deep Deterministic Policy Gradient.

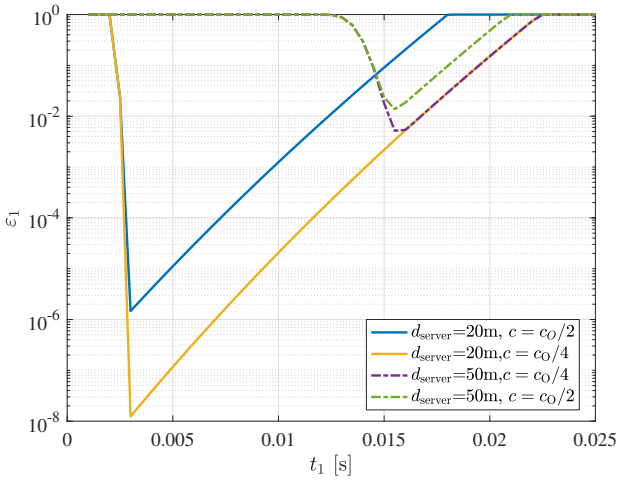


Fig. 4: The error probability in the selected link versus the duration of communication phase t_1 with different setups of the average transmission distances d_{server} (from the UE to servers) and the average assigned workloads c.

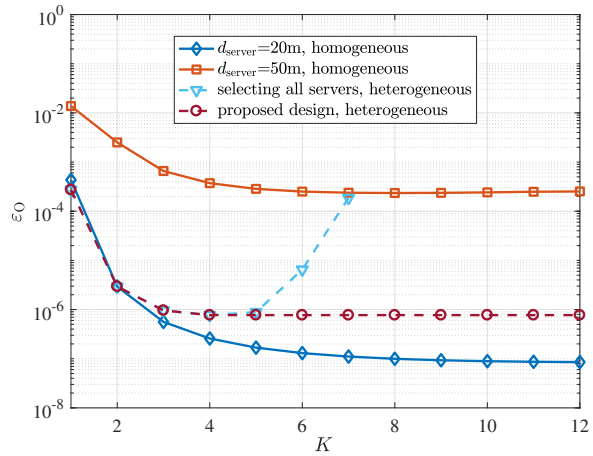


Fig. 5: The overall error probability ε_{O} versus the number of available servers K with homogeneous servers or heterogeneous servers.

probability is convex in t_1 with variant setups, which confirms our analytical result in Lemma 1. Secondly, the optimal solutions of t_1 vary with different values of distance d_{server} (between the servers and UE). When we have a short d_{server}, the system prefers also a small t_1. On the other hand, if we fix d_{server}, the value of c barely influences the optimal solution of t_1. Furthermore, it shows that one possible improvement of the reliability is to select more servers, which corresponds to reduce c of each server. However, such approach becomes significantly inefficient, if the distance between the server and UE is relatively far e.g., $d_{\text{server}} = 50$m.

Next we present Fig. 5 to further investigate the impact of server selection to the system performance. We plot the end-to-end error probability ε_{O} versus the total number of servers K under both homogeneous scenario (servers with same distances from the UE) and heterogeneous scenario (servers with different distances from the UE). Firstly, The reliability of the whole network is improved by increasing the number of available servers regardless of scneraio types. This observation coincides with the results from Fig. 4. However, adding more servers brings lesser performance improvement when K becomes relatively large, i.e., the slope of the curves decreases when K increases. In such scenario, the bottleneck of the system lies on the computation error. It is worth to mention that by applying the proposed design, the reliability of the considered network outperforms other approaches. More importantly, owning to our design, introducing more additional servers brings more considerable performance improvement, comparing to the rest cases. Furthermore, it shows the importance of having the proposed server selection method. In particular, it is not always optimal by adding more server, if we simply select all available servers deployed in the network (only keeps the workload assignment process of the proposed algorithm). In fact, it can worsen the performance when K is already large and the diversity of SNR is high. The system can even fail to find a feasible solution if it selects a server

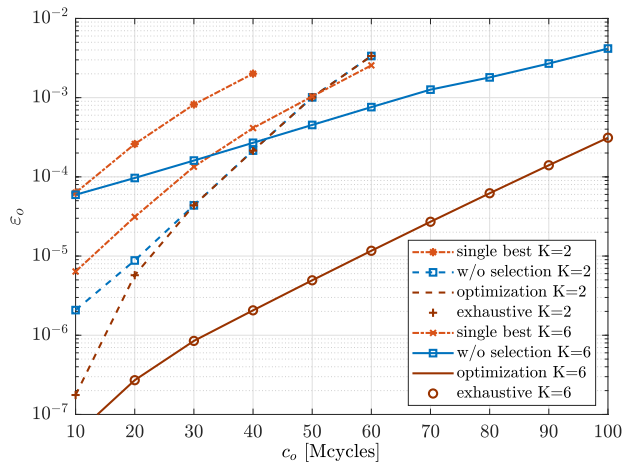


Fig. 6: Overall error probability ε_{O} versus total workloads c_{O} with available server size K=2 and K=6. The proposed design is provided along with the case selecting all servers and selecting single best server. In addition, the solution with exhaustive searching is also shown for comparison.

with an extreme low SNR. It also implies that the proposed joint design is more important in practical scenarios with heterogeneous servers.

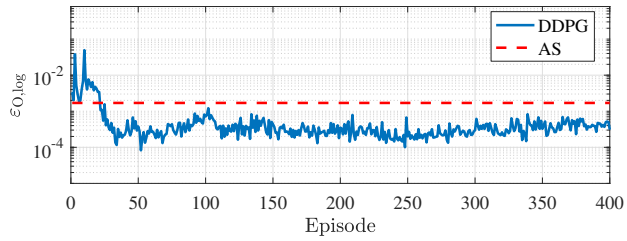
At the end of this subsection, we evaluate the impact of the total workloads on the reliability of the considered MEC networks, while comparing the performance of the proposed design with two low-complexity approaches, i.e, selecting all available servers and selecting single best server with highest SNR. In addition, we also show the results of exhaustive search to illustrate the gap between our design and the global optimum. The results are shown in Fig 6. We observe that all the overall error probability curves are increasing in c_{O}. In particular, these curves increase in different manners. Firstly, the performance with proposed design advances significantly comparing to the case without a proper server selection scheme (simply selecting all servers) regardless of setups. Subsequently, such performance advantage of the proposed design becomes more significant for the scenario, where more servers are available in the network. This phenomenon coincides with the results observed from Fig. 5. Interestingly, we can observe that if we select all servers or select only the single best server, the performance curves of cases with $K = 6$ and $K = 2$ are across with each other. Moreover, when the workload is low, selecting the single best server is a better strategy, in comparison to selecting all servers. This implies that if the system does not select the server properly , letting more servers compute the task holds back the performance when the task is computation-intensive with heavy workload. Furthermore, with the proposed server selection process, the system benefits even more significantly from having more servers (for selection) under the cases with relatively heavier workloads. Furthermore, the proposed design matches with the exhaustive search very well. In fact, when the error probability is low, the impact of high order terms $\varepsilon_1\varepsilon_2$ is so tiny so that the influence to the optimal solution is barely visible. Since the proposed design is essentially to solve a mixed integer convex problem, the (nearly) global optimal solution can be obtained.

C. Performance of Systems with Time-Dependent Servers and Outdated CSI

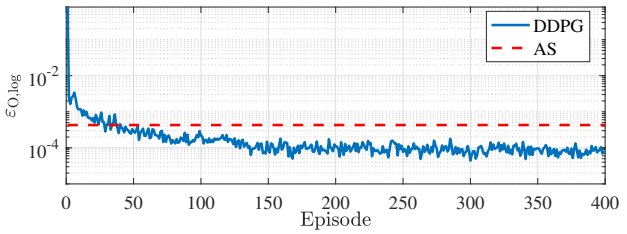
In this subsection, we evaluate the system performance under the scenario where only outdated CSI knowledge are available and the statuses of servers are dependent on the previous actions. We extend the simulation parameters with $\rho = 0.9$ and $T_k^{\mathrm{pre}} = 6$ms. The performance of DDPG algorithm, denoted as DDPG in the figures, is compared to the benchmark, denoted as AS in the figures. The benchmark results are obtained from the heuristic way, where decisions of frame structure, workload assignment and server selection are based on the analytical solution in section III by considering average channel gain based on the outdated CSI and ignoring the influence of current actions at time slot m to the future, i.e., $R^{(m+1)} = 0$.

The parameters setting for DDPG network as follows: The actor network consists of 4 hidden layers with 32 and 64 neurons. The critic network consists of 2 hidden layers for the action input with 32 and for the state input with 16 neurons. We apply batch normalization for the input of both the actor and critic networks, which improves the training stability in DDPG [21]. The learning rate of the actor and critic are set as $\alpha_\mu = 0.00005$ and $\alpha_Q = 0.0001$, respectively. We use a discount factor $\gamma_{\mathrm{df}} = 0.3$. For random exploration we consider $\theta_{\mathrm{OU}} = 0.5$ and $\sigma_{\mathrm{OU,max}} = 0.8$ with σ_{OU} decreasing to $\sigma_{\mathrm{OU,min}} = 0.1$. In training process the size of each sampled batch is set as $U_{\mathrm{b}} = 512$. The agent learns for a total of 400 episodes for each scenario, where the network is trained for $M_{\mathrm{train}} = 1000$ times with correlated channel realizations in each episode. To capture the overview of the trained network over the fading channel realizations, we take the log-average error probability $\varepsilon_{\mathrm{O,log}} = \exp\left(\frac{1}{M_{\mathrm{train}}}\sum_1^{M_{\mathrm{train}}} \log\left(\varepsilon_{\mathrm{O}}^{(m)}\right)\right)$ as the performance metric. First, in Fig. 7, the log-average error probability versus training episodes is depicted. It is observed that in both cases, where the number of available servers is $K = 2$ (the top subplot) and $K = 4$ (the bottom subplot), the error probability converges over episodes and the proposed DDPG algorithm eventually outperforms the analytical solution. Although increasing the number of available servers K improves the performance as expected, it is also worth to mention that it also requires more training episodes to converge. This is due to the fact that increasing K not only introduces more possible combinations of server selection, but also increases the possible frame structures and workload assignments accordingly, resulting a significantly larger action space.

Next, we investigate the influence of coherent coefficient to the system shown in Fig. 8, which depicts the log-average error probability versus episode over training duration with a high coherent channel with correlation coefficient $\rho = 0.9$ and the low coherent channel with $\rho = 0.1$. Similarly, the error probability decreases over the training and converges at the end for both kind of channels. However, the convergence behaviour with the low coherent channel is more unstable than with the high coherent one. Moreover, $\rho = 0.1$ implies that the channel is nearly random and independent at each frame, where one small change of the action that the agent chooses, e.g., increase of blocklength or increase workload, leads to the volatility of performance. In spite of that, our DDPG algorithm still performs better than the benchmark. Finally, we show the impact of total frame duration T and total workload

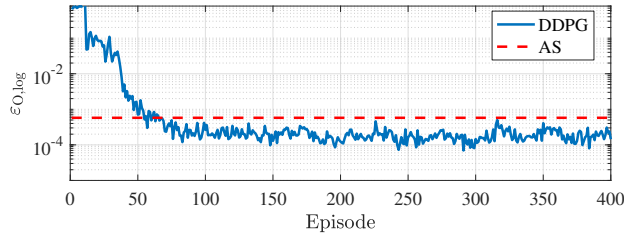


(a) $K = 2$ with fixed $\rho = 0.9$.

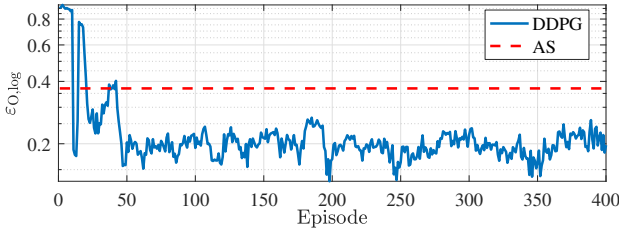


(b) $K = 4$ with fixed $\rho = 0.9$.

Fig. 7: Log-average error probability versus episodes of training for outdated CSI scenario with $\rho = 0.9$. Two cases with $K = 2$ and $K = 4$ are considered.



(a) $\rho = 0.9$ with fixed $K = 3$



(b) $\rho = 0.1$ with fixed $K = 3$

Fig. 8: Log-average error probability versus episode in outdated CSI scenarios with different CSI accuracy, i.e.,$\rho = 0.9$ and $\rho = 0.1$. In addition, K is set to 3. In addition, the case selecting all servers and selecting single best server is also sown for comparison.

c_{O} to the system performance. In particular, we compare the proposed DDPG algorithm with the analytical solution as well as two low-complexity approaches. It is shown that both increasing available T and decreasing required c_{O} improve the reliability performance. More interestingly, we can find that the DDPG algorithm shows advantages when the system has more resources, e.g., more transmission and computation time in the subfigure (a), or less last, e.g., less required workloads in the subfigure (b). We can also observe that the gap becomes slighter in the other way around. However, the performance in those cases are already non-reliable, which are not the target scenario of our works. Furthermore, it also confirms the necessity of server selection in comparison to simply offloading the task to all available servers. In particular, when the workload is low and the frame duration is long, with a high channel correlation selecting the best server is actually

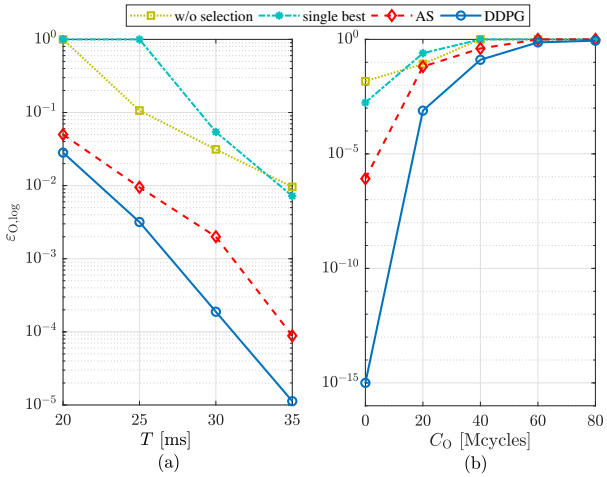


Fig. 9: Log-average error probability versus (a) total frame duration T and (b) total workload c_{O} with $K = 3$ and $\rho = 0.9$ based on outdated CSI.

a decent strategy, since a strong link is unlikely to become extremely worse in the next frame. Otherwise, we should select more servers to mitigate the risk of extreme cases. Therefore, it shows the advantage of the proposed DDPG from both analytical and practical perspectives.

VII. Conclusion

In this work, we proposed a reliability-optimal design in a multi-server edge computing network by jointly optimally selecting multiple servers and optimally allocating the allowed time for the communication and computation phases. In particular, both the communication errors due to FBL impact and computation errors caused by delay violation are taken into account in the characterization of the overall error probability. Under the assumption of the system with time-independent servers and perfect CSI, we formulated an optimization problem which minimizes the overall error probability of the whole service within the maximal allowed service delay. Based on the analysis of the decomposed problems, we reformulated the original problem as a MICP problem, which can be solved efficiently. For scenario with the time-dependent servers and outdated CSI, we propose a DRL approach with the deep deterministic policy gradient method to minimize the error probability while taking the future reward into account.

The simulation results confirmed our analytical model and verified the applicability of our DRL approach. Moreover, the performance advantages of both designs have been shown in the corresponding scenarios. In particular, the proposed analytical design under the assumption of perfect CSI and time-independent servers is able to obtain the global optimum in such scenario. On the other hand, as this design aims at the instantaneous reliability performance, it is not preferred for the scenarios where the CSI is outdated (but correlated with the current one) and the status of a server is also correlated in time (i.e., the decisions of servers selection and workload allocation in one frame influence the statuses of the servers in the next frame). At the same time, the DRL based approach shows a higher performance owning to considering a long term performance by taking the future reward into account.

Although in the considered system the feedback phase is ignored, it should be pointed out that the proposed design can be extended to scenarios with practical assumptions on the feedback. In particular, with consideration of the feedback time cost and feedback error probability, Lemma 3 also holds after certain extensions in the time-independent case, while for the time-dependent case the DDPG algorithm could also handle the new scenarios with an increased action space, which are interesting extensions as our future works.

REFERENCES

[1] Y. Zhu, et al., "Reliability-Optimal Offloading in Multi-Server Edge Computing Networks with Transmissions Carried by Finite Block-length Codes," in *IEEE ICC Workshops*, Shanghai, China, May 2019, pp. 1-6.

[2] S. Keki, et al., "MEC in 5G networks" [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf

[3] Y. Mao, C. You, J. Zhang, K. Huang and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322-2358, Fourthquarter 2017.

[4] M.R.Rahimi, et al.,"Mobile Cloud Computing: A Survey, State of Art and Future Directions", in *Mobile Netw. Appl.*, vol. 19, no. 2, pp.133-143, April 2014.

[5] C. She, C. Yang, and T. Q. S. Quek, "Radio Resource Management for Ultra-reliable and Low-latency Communications,"*IEEE Commun. Mag.*, vol. 55, no. 6, pp 72-78, Jun. 2017.

[6] C. She, C. Yang, and T. Q. S. Quek,"Cross-layer Optimization for Ultra-reliable and Low-latency Radio Access Networks,"*IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 127-141, Jan. 2018.

[7] A. Ghasempour, "Internet of Things in Smart Grid: Architecture, Applications, Services, Key Technologies, and Challenges," *Inventions journal*, vol. 4, no.1, pp. 1-12, 2019.

[8] 3GPP Release 16. Avaliable: http://www.3gpp.org/release-16.

[9] M. S. Elbamby et al., "Wireless Edge Computing With Latency and Reliability Guarantees," *Proc. IEEE*, vol. 107, no. 8, pp. 1717-1737, Aug. 2019.

[10] G. Durisi, T. Koch and P. Popovski, "Toward Massive, Ultrareliable, and Low-Latency Wireless Communication With Short Packets," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1711-1726, Sept. 2016.

[11] C. Sun, C. She, C. Yang, T. Q. S. Quek, Y. Li and B. Vucetic, "Optimizing Resource Allocation in the Short Blocklength Regime for Ultra-Reliable and Low-Latency Communications," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 402-415, Jan. 2019.

[12] C. Liu, M. Bennis, M. Debbah and H. V. Poor, "Dynamic Task Offloading and Resource Allocation for Ultra-Reliable Low-Latency Edge Computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132-4150, June 2019.

[13] F. Wang, J. Xu, X. Wang and S. Cui, "Joint Offloading and Computing Optimization in Wireless Powered Mobile-Edge Computing Systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784-1797, March 2018.

[14] Y. Dai, D. Xu, S. Maharjan and Y. Zhang, "Joint Load Balancing and Offloading in Vehicular Edge Computing and Networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377-4387, June 2019

[15] 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.

[16] C. She, Y. Duan, G. Zhao, T. Q. S. Quek, Y. Li and B. Vucetic, "Cross-Layer Design for Mission-Critical IoT in Mobile Edge Computing Systems," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9360-9374, Dec. 2019.

[17] Y. Polyanskiy, H. Poor, and S. Verdu, "Channel coding rate in the finite blocklength regime," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2307–2359, May 2010.

[18] Sutton, Richard S., and Andrew G. Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[19] M. L. Puterman, *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*, 2014.

[20] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529.

[21] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).

[22] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *IEEE WCNC*, 2018, pp. 1–6.

[23] J. Chen, S. Chen, Q. Wang, B. Cao, G. Feng and J. Hu, "iRAF: A Deep Reinforcement Learning Approach for Collaborative Mobile Edge Computing IoT Networks," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 7011-7024, Aug. 2019.

[24] L. Huang, S. Bi and Y. J. Zhang, "Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks," *IEEE Trans. on Mobile Comput.*, early access.

[25] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji and M. Bennis, "Optimized Computation Offloading Performance in Virtual Edge Computing Systems Via Deep Reinforcement Learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005-4018, June 2019.

[26] C. She, R. Dong, Z. Gu, Z. Hou, Y. Li, W. Hardjawana, C. Yang, L. Song, and B. Vucetic, "Deep learning for ultra-reliable and low-latency communications in 6G networks," *arXiv preprint* arXiv:2002.11045, 2020.

[27] T. Yang, Y. Hu, M. C. Gursoy, A. Schmeink and R. Mathar, "Deep Reinforcement Learning based Resource Allocation in Low Latency Edge Computing Networks," *IEEE ISWCS*, Lisbon, 2018, pp. 1-5.

[28] S. Coles, *An Introduction to Statistical Modeling of Extreme Values.* Springer, London. 2001.

[29] X. Cao, F. Wang, J. Xu, R. Zhang and S. Cui, "Joint Computation and Communication Cooperation for Energy-Efficient Mobile Edge Computing," *IEEE Internet Things J.* early access, Oct 2018

[30] J. Vicario and C. Anton-Haro, "Analytical assessment of multi-user vs. spatial diversity trade-offs with delayed channel state information," *IEEE Commun. Lett.*, vol. 10, no. 8, pp. 588–590, Aug. 2006.

[31] R. Mallik, "On multivariate Rayleigh and exponential distributions," *IEEE Trans. Inf. Theory*, vol. 49, no. 6, pp. 1499–1515, Jun. 2003

[32] Little, John D. C., "A Proof for the Queuing Formula: L = λ W", *Oper. Res.*, vol. 9, no. 3, pp.383-387, Jun.1961.

[33] Y. Hu, Y. Zhu, M. C. Gursoy, A. Schmeink, "SWIPT-Enabled Relaying in IoT Networks Operating with Finite Blocklength Codes", *IEEE J. Sel. Areas Commun.*, vol. 37, no. 2, pp.1-14, Feb. 2019.

[34] Lubin, M., Yamangil, E., Bent, R. et al. , "Polyhedral approximation in mixed-integer convex optimization" *Math. Program.*, vol. 172, no. 1, pp.139-168, Nov. 2018.

[35] P. He, L. Zhao, S. Zhou and Z. Niu, "Water-Filling: A Geometric Approach and its Application to Solve Generalized Radio Resource Allocation Problems," *IEEE Trans. Wireless Commun.*, vol. 12, no. 7, pp. 3637-3647, July 2013.

[36] Y. Corre, J. Stephan and Y. Lostanlen, "Indoor-to-outdoor path-loss models for femtocell predictions," in Proc. *IEEE PIMRC*, Toronto, ON, 2011, pp. 824-828.

[37] Watkins, C. J., & Dayan, P. (1992). Q-learning. Machine learning, 8(3-4), 279-292.

[38] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[39] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion" *Physical Rview*, vol. 36, no. 5, p. 823, 1930.

[40] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.