Task Replication in Unreliable Edge Networks

Ibrahim M. Amer*, Sharief M. A. Oteafy*†, and Hossam S. Hassanein*
*School of Computing, Queen's University, Kingston, Ontario, Canada
ibrahim.amer@queensu.ca, hossam@cs.queensu.ca
†School of Computing, DePaul University, Chicago, Illinois, USA
soteafy@depaul.edu

2022 IEEE 47th Conference on Local Computer Networks (LCN) | 978-1-6654-8001-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/LCN53696.2022.9843613

***Abstract*— Edge networks provide ample resources for low-latency service recruitment, unlike remote resources in the Cloud. As such, smart devices and Internet of Things (IoT) nodes form a pool of Extreme Edge Devices (EED) that are within reach of *Mist* and *Fog* networks, providing significant advantages in latency, geographic cognizance, and reduced communication costs. EEDs are often recruited in Edge networks assuming they are reliable in their commitment to tasks. However, many EEDs may fail to fulfill their tasks because they operate under opportunistic approaches and are prone to intermittent connectivity. To ameliorate task failure, we aim to optimize task allocation under the assumption of failure. Additionally, we optimize CPU utilization to engage reliable EEDs, resorting to replication when needed to exceed a tunable reliability margin. We demonstrate the efficacy of our model in multiple scenarios and present future work in EED utilization.**

***Index Terms*—Edge Computing, Extreme Edge Device, Uncertainty, Task Replication, System Reliability**

I. Introduction

Mobile Edge Computing (MEC) is a paradigm that aims to facilitate and support the computational capability of mobile devices such as smartphones, wearable devices, laptops, tablets, and other low-resource systems at the periphery of the network. MEC is used to mitigate latency challenges with relatively distant Cloud servers, traditionally used to enable resource-demanding storage and computational resources in Mobile Cloud Computing (MCC) [1]. While many Edge services enlist the resources of MCC given the vast capacity of the Cloud [2], it is prudent to account for the inherent delays and costs associated with such services, especially for time-sensitive edge services.

The proximity of the MEC architecture to the Edge devices inevitably aids in latency challenges, which is a concern, especially with the projected growth in IoT devices and services at the network Edge. What if we can utilize devices that are much closer to mobile devices, within the realm of personal and body networks? This may reduce both latency and network congestion as it aggregates upstream. Resource-limited devices themselves are computing devices, and their ubiquitous deployment has sparked significant research on short-range resource probing and provisioning. Resource-limited devices can perform a range of tasks from small-sized to medium-sized tasks with ease. Most power-constrained devices such as smartphones, IoT devices, and tablets, can be referred to as Extreme Edge Devices (EED) or Mist nodes [3] and are thereby recruitable to Edge services. EEDs can push the computation further to the network Edge because they include any device close to the requesting service [4]. In addition, traffic heading to the Cloud infrastructures would be pruned to tasks that require intensive computing capabilities or non-immediate responses. As a result, this will enhance the overall performance of Cloud-based networks by applying this separation of concerns principle.

An EED can be any user-owned smart device, any orchestrator that probes its resources cannot assume guaranteed availability when the task offloading process starts. For example, the battery capacity of an EED may no longer be sufficient because the device is executing a power-draining task, or is experiencing significantly high CPU utilization. Furthermore, a worker might reject the task for other reasons, including connection loss or device mobility that distances the resource from the orchestrator's region of operation. Thus, the uncertainty inherent in Edge networks is a critical factor that mandates careful planning when probing resources and provisioning task assignments, whereby task replication is crucial to ensure task completion with minimal rejection rates.

This paper, introduces our Task Replication in Unreliable Edge networks (**`TRUE`**) scheme. **`TRUE`** aims to assign tasks requested by multiple services to available EEDs, in a highly uncertain environment. To manage uncertainty, we replicate the set of tasks on multiple EEDs until a certain level of reliability is met without over-provisioning the available EEDs. A central controller or orchestrator will probe the environment for available EEDs, whether classified as Mist or Fog nodes [3], which will hereafter be referred to as EEDs. The uncertainty in the computing environment herein applies to any uncertainty. However, we will focus on the battery capacity of the EED. The lower the battery capacity of the workers/EEDs, the more replicas will be allocated for the task.

To guarantee we are not over-provisioning the available EEDs, **`TRUE`** takes a best-fit approach to recruit with the CPU capacity closest to the tasks' requirements. Therefore, we first introduce a mathematical model for the problem based on the Integer Linear Program (ILP) formulation. We then derive the analytical solution to the problem using Karush–Kuhn–Tucker (KKT) conditions for optimality and Lagrangian analysis [5] to attain bounds on the optimal solution. Finally, We show the results using the Gurobi solver [6] and then compare them with two baseline approaches.

The main contributions of this work can be summarized in the following points:

1) We investigate the task recruitment and replication problem in highly unreliable Edge networks. The workers/EEDs we aim to offload and replicate the tasks to have battery capacities that follow a probability distribution of failure. **TRUE** replicates tasks on multiple workers to avoid task failure until a pre-determined reliability limit is met.
2) We formulate the problem as an ILP with constraints on computation and communication delays, CPU capacity of the worker, energy consumption limit of the worker, CPU requirement of the task, the maximum number of tasks that can be carried out by the worker, and most importantly, controlling the overall system reliability by aggregating the reliability scores of the most reliable workers according to a pre-determined threshold.
3) We derive the analytical solution of the formulated ILP using KKT and Lagrangian analysis accompanied with the proof to attain lower bounds on the optimal solution.

We compare the formulated IL program with two baseline approaches. The first depends on maximizing the number of replicas in general, while the second aims to maximize the CPU utilization of workers, assuming they are committed to completing their assigned tasks. The results demonstrate the impact of accounting for the reliability of workers as an important factor in utilizing the workers better, avoiding over-provisioning or under-provisioning the resources and minimizing the probability of task failure.

The remainder of the paper is organized as follows. Section II highlights some of the related works. Section III presents the proposed **TRUE** scheme. Section IV discusses the performance evaluation and the simulation results. Section V concludes the paper and presents future insights into the EED assignment problem.

II. Related Work and Motivation

Task replication under uncertainty requires significant consideration in MEC, especially given the recent approaches in opportunistically recruiting Mist and Fog devices for time-sensitive services. Furthermore, uncertainty affects decision-making by introducing hindering factors to the computing environment , such as performance degradation, battery drainage, and service failure. In this section, we survey novel research on uncertainty in Edge computing; then, we contrast the operational mandate of our presented model, **TRUE**.

Xu et al. [7] proposed a software-defined network-based (SDN) Edge computing framework and a dynamic resource provisioning (UARP) method to address the uncertainty. The uncertainty was addressed through the advantages of a Software Defined Network (SDN) with good results. In addition, a non-dominated sorting genetic algorithm was employed to optimize the energy consumption and the completion time. However, the work did not address the replication problem.

Chang et al. [8] investigated the continuous application offloading decision in Edge computing. In their system model, it is uncertain how users operate continuous applications and how long these applications would last before completion. The uploading and downloading data size for offloading computation of each user operation, the number of user operations, and the number of CPU cycles required to execute the computation of each operation are unknown. The problem was formulated as an energy consumption constrained average response time minimization problem among multiple users under uncertainty. A Response Time-improved Offloading algorithm with Energy Constraint (RTIOEC) was proposed to make the offloading decision.

Ghoorchian and Maghsudi [9] investigate the computation offloading problem in a dynamic network under uncertainty. A smart device chooses a server under uncertainty to enhance the decision-making based on historical time and energy consumption. The problem was modeled and solved using a budgeted non-stationary Multi-armed Bandit (MAB) formulation.

Amer and Sorour [10] proposed a scheme for EEDs recruitment based on recruitment costs. The workers are recruited subject to constraints on communication and computation delays, task's CPU requirements, and the amount of task's data. A task is divided into several subtasks, where each subtask is to be executed by a worker. The problem was formulated as Mixed Integer Quadratically Constrained Quadratic Program (MIQCQP).

However, these approaches assume that each worker would fulfill its assigned task. Realistically, workers may fail to do so due to intermittent or permanent failures. Such failures include depleted energy reservoirs, run-time errors on the nodes, errors stemming from multi-tenancy (when serving multiple applications), operating system-induced pre-emptions, and permanent failures such as outright component malfunction.

In addition, in environments where EEDs are opportunistically recruited, these workers may decline tasks based on imposed local optimization criteria beyond those communicated to the central controller, or changes to their operational conditions after their availability is communicated. In these cases, the promise that they could deliver on a task would fail.

To remedy such failure, we aim to optimize our task offloading to workers, and factor in their unreliability by replicating the tasks to multiple workers, knowing that some may fail due to the previously mentioned intermittent or permanent failures. The unreliability of each worker is quantified using a reliability probability metric. The more unreliable workers exist, the more replicas our scheme would recruit. We control the system level of reliability using a pre-determined threshold. By increasing the threshold value, our scheme assigns more workers to the task until the desired reliability is reached.

III. Modelling Task Replication in Unreliable Edge Networks

In this section, we describe our proposed **TRUE** scheme. **TRUE** strives to assign a set of tasks initiated by some services to some EEDs in the area. When the worker's reliability level is insufficient, **TRUE** will replicate the task to multiple workers to ensure the task's completion under specific constraints. Note that the terms workers and EEDs are used interchangeably

throughout this paper. Our objective is to minimize the wasted resources by recruiting the best fit workers, i.e., the workers with CPU capacities closest to the task's requirements. In addition, the number of replicas will be maximized for tasks assigned to workers with less battery capacity to ensure that these tasks will be completed.

We now introduce our system model in detail. Then, in section III-B, the problem is formulation as an ILP is presented. Finally, in subsection III-C, we derive the analytical solution for the problem.

A. System Model

Consider a set of M tasks, which require processing by EEDs, are denoted by $\Gamma = \{\gamma_1, \cdots, \gamma_M\}$. Each task $\gamma_j \; \forall j \in \Gamma$ is defined by three attributes, namely: data size γ_j^{data} in bits, CPU requirement γ_j^{CPU}, and processing density $\gamma_j^{\text{density}}$ in cycle/bit, i.e., the number of CPU cycles (burst) required to process a single bit of task's data. Each task γ_j has a computation delay τ_{ij}^{comp} when it gets executed on worker w_i. The computation delay of task γ_j is given by equation (1). In addition, each task has a delay threshold denoted by $\gamma_j^{\text{deadline}}$, i.e., the maximum delay that can be tolerated by the task's request.

To execute the required tasks, a central controller c located in the area will receive tasks requests that need processing. The controller will probe local EEDs to determine their availabilities and facilitate their inclusion in the assignment round. It would then assign tasks to the available workers within the area optimally based on the information it has about the workers. Consider a set of N workers denoted by $\mathcal{W} = \{w_1, \cdots, w_N\}$ that are available within the area. Each worker w_i has a computation capability denoted w_i^{CPU}, i.e., the CPU frequency or clock speed expressed in cycles/second. Each worker can take on multiple tasks. The maximum number of tasks that a worker w_i can accept is denoted by w_i^{tasks}. Furthermore, the energy consumed by the computation resources of each worker is defined by equation (2).

$$\tau_{ij}^{\text{comp}} = \frac{\gamma_j^{\text{density}} \gamma_j^{\text{data}}}{w_i^{\text{CPU}}} \tag{1}$$

$$E_{ij}^{\text{comp}} = \kappa \; \gamma_j^{\text{data}} \; \gamma_j^{\text{density}} \; \left(w_i^{\text{CPU}}\right)^2 \tag{2}$$

Where κ is the on-chip capacitance factor of the worker's CPU. Each worker has a cap on the maximum energy consumed due to processing task's data denoted by $E_i^{\max}$.

1) Communication Model

We assume that our system is based on the Orthogonal Frequency-Division Multiple Access (OFDMA) modulation scheme. The time needed to send the task's data γ_j^{data} from the controller c to a worker w_i is denoted by τ_{ij}^{comm}, i.e., the transmission delay based on the data rate of the link. The total bandwidth of the controller is denoted by W. To send the data between the controller and the worker, each worker is assigned a sub-channel w_i^{channel} with a bandwidth equal to $\frac{W}{N}$. To quantify the level of the transmitted signal (from c to w_i) to the level of the background noise, we use the Signal-to-Noise Ratio (SNR) measure given by:

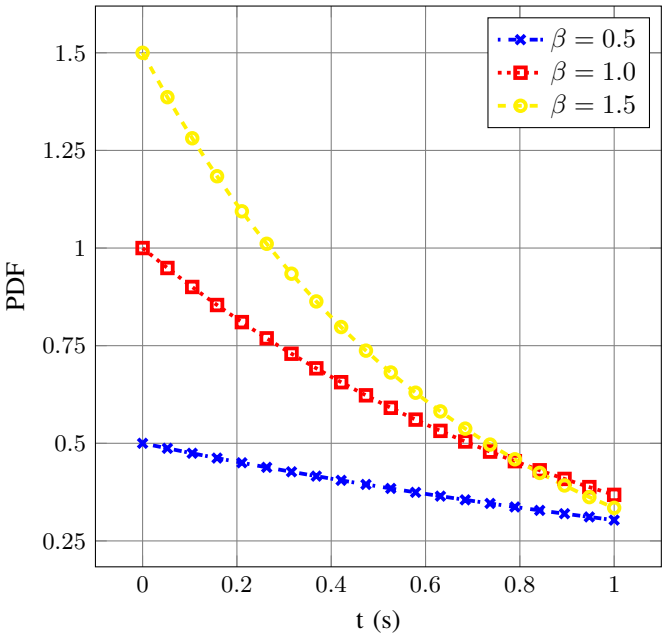

Figure 1: Probability density function of battery drainage with time using different values of failure/hazard rates.

$$\varphi_i = \frac{P \; h_i}{\sigma^2} \tag{3}$$

where P is defined as the transmission power of the controller in watts (W), σ^2 is the channel noise variance in watts (W), and h_i is the channel gain for each worker expressed as $h_i = d_i^{-\nu} g_i^2$ where d_i is the distance between the controller and the worker w_i in meters (m), ν is the path loss exponent, and g_i is the Rayleigh fading channel coefficient for each worker.

The data rate (bit/s), which is defined as the amount of data transmitted during a specified time period over a network for each worker, is given by:

$$R_i = W_i \log_2(1 + \varphi_i) \tag{4}$$

The communication delay τ_{ij}^{comm} is defined as the ratio between the task's bits γ_j^{data} and the data rate R_i and is given by:

$$\tau_{ij}^{\text{comm}} = \frac{\gamma_j^{\text{data}}}{R_i} \tag{5}$$

The total delay that is imposed by the network parameters, computation capability of the worker, and the task's requirement is given by:

$$T_{ij} = \tau_{ij}^{\text{comp}} + \tau_{ij}^{\text{comm}} \tag{6}$$

2) System Reliability

We denote the failure probability distribution of each worker by $F(t) = P(x \leq t)$, where $F(\cdot)$ is the Cumulative Distribution Function (CDF) and $P(\cdot)$ is the Probability Distribution

Function (PDF). Such distribution represents the probability that a worker w_i will fail prior to time t [11]. The time $t = 0$ represents the moment when the worker is put into operation. Therefore, we can define the system reliability function as follows:

$$R(t) = 1 - F(t) = P(x > t) \tag{7}$$

The probability of failure is exponentially distributed with time and is given by:

$$P(t) = \beta e^{-\beta t} \tag{8}$$

where β is the hazard or the failure rate of the worker. The mean time for a hazard/failure is denoted by $\frac{1}{\beta}$. Fig. 1 depicts the effect of varying the hazard rate on the PDF of failure across the time.

In addition, the failure CDF is defined as the area under the curve of the PDF between time 0 and t and is given by:

$$F(t) = \int_0^t P(t)\, dx = 1 - e^{-\beta t} \tag{9}$$

As a result of (9), the reliability probability $R(t)$ can be given by:

$$R(t) = e^{-\beta t} \tag{10}$$

Each worker w_i has a hazard rate of β_i, and hence, a reliability probability at time t and is denoted by $R_i(t)$. The reliability represents the probability that a task will be executed successfully by the worker prior to a failure time t. In the next section, we will illustrate how we can utilize the reliability of the workers within the edge network infrastructure to meet a certain reliability level.

B. Problem Formulation

Our objective is to assign the set of tasks Γ to the available workers set $\mathcal{W}$ within an area taking into consideration maximizing the number of replicas for tasks assigned to workers with less reliability subject to the constraints in the constrained program in (11). The objective function in (11a) aims to minimize the wasted allocated resources of the workers, in other words, it tries to find the best fit among the available workers by minimizing the difference between the task γ_j CPU requirement and the CPU capability of the allocated worker w_i.

$$\min_{x_{ij}} \quad \sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij}\left(w_i^{\text{CPU}} - \gamma_j^{\text{CPU}}\right) \tag{11a}$$

$$\text{s.t.} \quad x_{ij}\left(w_i^{\text{CPU}} - \gamma_j^{\text{CPU}}\right) \geq 0 \quad \forall i \in \mathcal{W}, \forall j \in \Gamma \tag{11b}$$

$$\sum_{j=1}^{M} x_{ij} E_{ij}^{\text{comp}} \leq E_i^{\max} \quad \forall i \in \mathcal{W} \tag{11c}$$

$$\sum_{i=1}^{N} x_{ij} R_i(\gamma_j^{\text{deadline}}) \geq \epsilon \quad \forall j \in \Gamma \tag{11d}$$

$$T_{ij} x_{ij} \leq \gamma_j^{\text{deadline}} \quad \forall i \in \mathcal{W}, \forall j \in \Gamma \tag{11e}$$

$$\sum_{i=1}^{N} x_{ij} \geq 1 \quad \forall j \in \Gamma \tag{11f}$$

$$\sum_{j=1}^{M} x_{ij} \leq w_i^{\text{tasks}} \quad \forall i \in \mathcal{W} \tag{11g}$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in \mathcal{W},\ \forall j \in \Gamma \tag{11h}$$

Constraint (11b) ensures that each worker w_i assigned to task γ_j has a CPU frequency that is sufficient to handle the CPU requirement γ_j^{CPU} of task γ_j, which will prevent the objective function to allocate a worker with CPU that is less than the task's requirement. Constraint (11c) ensures that the total energy consumed by the tasks' load assigned to the worker w_i is less than a certain threshold $E_i^{\max}$. Constraint (11d) ensures that the total reliability probability of all the replicas assigned to task γ_j is more than or equal to a certain threshold ϵ. The reliability probability $R_i(\gamma_j^{\text{deadline}})$ for worker w_i represents the probability that the worker will finish the execution of task γ_j successfully before the deadline $\gamma_j^{\text{deadline}}$.

The objective function (11a) along with constraint (11d) will enforce allocating workers whenever needed only. For example, if task γ_j is allocated to a worker with a reliability probability equal to the system reliability threshold ϵ, then it is less likely to get allocated/replicated to another worker because recruiting more workers, in this case, would violate the minimization objective in (11a). Therefore, we can guarantee no under-provisioning or over-provisioning of the available resources would occur. The worker reliability probability $R_i(\cdot)$ here can represent any reliability metric but, in this paper, we will refer to the reliability as the probability of the worker executing the task successfully within the deadline before the battery is completely drained.

Constraint (11e) assures that the total delay of task γ_j when it gets executed on worker w_i is less than a certain deadline $\gamma_j^{\text{deadline}}$ enforced by task γ_j. Constraint (11f) guarantees that at least one worker will be assigned to each task so that all tasks get executed. Constraint (11g) guarantees that no worker is assigned more than the maximum allowed number of tasks w_i^{tasks}. Constraint (11h) is the integrality constraint associated with the binary decision variable x_{ij}. Each x_{ij} is equal to 1 if task γ_j is assigned to worker w_i and is equal to 0 otherwise.

C. Analytical Solution

The model illustrated in (11) has a linear objective function with binary decision variables, i.e., x_{ij}. The inequality constraints in (11b), (11c), (11d), (11e), (11f), and (11g) are all linear constraints. Given these inputs alongside the integrality constraint in (11h), we conclude that the program we have at hand is an Integer-Linear Program (ILP). ILPs are generally NP-hard and difficult to solve [12] so we need to relax the program to be able to solve it analytically. First, we have to check whether the problem is convex or not. By examining the objective function and the inequality constraints, we can see that they are all linear except for the integrality constraint in (11h). Hence, this leads us to conclude that the problem is non-convex. However, we can convexify the problem by

relaxing the constraint (11h) to be in the continuous form of $0 \leq x_{ij} \leq 1$, and then in the solution we can employ rounding techniques (*e.g.*, stochastic, or deterministic rounding). Therefore, the program after relaxation is transformed into a linear program.

Closed-form solutions for linear programs is generally difficult to attain but, we can obtain a lower bound on the optimal solution of the problem using Lagrangian analysis and KKT conditions. The Lagrangian multipliers associated with the objective function in (11a) are given by the vectors: $\boldsymbol{\lambda}, \boldsymbol{\delta}, \boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\omega}, \boldsymbol{\theta^{(1)}}$,and $\boldsymbol{\theta^{(2)}}$, respectively.

To find lower bound on the optimal solution, we introduce Theorem 1. $(\cdot)^*$ indicates the optimal value of the superscripted term and z denotes the replacement of the square bracketed term in equation (25) for simplification.

Theorem 1. *The lower bound on the optimal value of the vector of decision variables $\boldsymbol{x^*}$ that is responsible of deciding which task goes to which worker, is given by:*

$$x_{ij}^* = \begin{cases} 0 & \theta_{ij}^{*(1)} = 0 \text{ and } z > 0 \\ 1 & \theta_{ij}^{*(1)} > 0,\ z > 0, \\ & \text{and } \theta_{ij}^{*(1)} \approx z \\ \frac{E_i^{\max} - \sum_{\substack{1 \leq l \leq M \\ kl \neq ij}} x_{kl}^*}{E_{ij}^{comp}} & \delta_i^* > 0 \\ \frac{\epsilon - \sum_{\substack{1 \leq k \leq N \\ kl \neq ij}} x_{kl}^* R_k(\gamma_l^{deadline})}{R_i(\gamma_j^{deadline})} & \rho_j^* > 0 \\ \frac{\gamma_j^{deadline}}{T_{ij}} & \phi_{ij}^* > 0 \\ 1 - \sum_{\substack{1 \leq k \leq N \\ kl \neq ij}} x_{kl}^* & \psi_j^* > 0 \\ w_i^{tasks} - \sum_{\substack{1 \leq l \leq M \\ kl \neq ij}} x_{kl}^* & \omega_i^* > 0 \end{cases} \tag{12}$$

Proof. From the results conducted in Theorem 1, we can conclude that the closed-form solution can not be attained because the bounds on the decision variables $x_{ij}^*\ \forall i, j$ are expressed in terms of other unknown variables, i.e., the Lagrangian multipliers and the decision variables.

To proof Theorem 1, we start by deriving the Lagrangian function associated with the objective function and the constraints in (11):

$$\begin{aligned} \mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\delta}, \boldsymbol{\rho}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\omega}, \boldsymbol{\theta^{(1)}}, \boldsymbol{\theta^{(2)}}) &= \sum_{i=1}^{N} \sum_{j=1}^{M} x_{ij} \left(w_i^{\text{CPU}} - \gamma_j^{\text{CPU}} \right) \\ &+ \sum_{i=1}^{N} \sum_{j=1}^{M} \lambda_{ij} \left(x_{ij} \left(w_i^{\text{CPU}} - \gamma_j^{\text{CPU}} \right) \right) + \sum_{i=1}^{N} \delta_i \left(\sum_{j=1}^{M} x_{ij} E_{ij}^{\text{comp}} - E_i^{\max} \right) \\ &+ \sum_{j=1}^{M} \rho_j \left(\epsilon - \sum_{i=1}^{N} x_{ij} R_i(\gamma_j^{\text{deadline}}) \right) + \sum_{i=1}^{N} \sum_{j=1}^{M} \phi_{ij} \left(T_{ij} x_{ij} - \gamma_j^{\text{deadline}} \right) \\ &+ \sum_{j=1}^{M} \psi_j \left(1 - \sum_{i=1}^{N} x_{ij} \right) + \sum_{i=1}^{N} \omega_i \left(\sum_{j=1}^{M} x_{ij} - w_i^{\text{tasks}} \right) \\ &+ \sum_{i=1}^{N} \sum_{j=1}^{K} \theta_{ij}^{(1)} \left(x_{ij} - 1 \right) - \theta_{ij}^{(2)} x_{ij} \end{aligned} \tag{13}$$

Applying KKT optimality conditions to the constraints in (11) we get:

$$\lambda_{ij} \left(x_{ij} \left(w_i^{\text{CPU}} - \gamma_j^{\text{CPU}} \right) \right) = 0 \quad \forall i \text{ and } \forall j \tag{14}$$

$$\delta_i \left(\sum_{j=1}^{M} x_{ij} E_{ij}^{\text{comp}} - E_i^{\max} \right) = 0 \quad \forall i \tag{15}$$

$$\rho_j \left(\epsilon - \sum_{i=1}^{N} x_{ij} R_i(\gamma_j^{\text{deadline}}) \right) = 0 \quad \forall j \tag{16}$$

$$\phi_{ij} \left(T_{ij} x_{ij} - \gamma_j^{\text{deadline}} \right) = 0 \quad \forall i \text{ and } \forall j \tag{17}$$

$$\psi_j \left(1 - \sum_{i=1}^{N} x_{ij} \right) = 0 \quad \forall j \tag{18}$$

$$\omega_i \left(\sum_{j=1}^{M} x_{ij} - w_i^{\text{tasks}} \right) = 0 \quad \forall i \tag{19}$$

$$\theta_{ij}^{(1)} \left(x_{ij} - 1 \right) = 0 \quad \forall i \text{ and } \forall j \tag{20}$$

$$\theta_{ij}^{(2)} x_{ij} = 0 \quad \forall i \text{ and } \forall j \tag{21}$$

The gradient of $\mathcal{L}$ at the optimal point is 0:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial x_{ij}} &= w_i^{\text{CPU}} - \gamma_j^{\text{CPU}} + \lambda_{ij}^* \left(w_i^{\text{CPU}} - \gamma_j^{\text{CPU}} \right) + \delta_i^* E_{ij}^{\text{comp}} \\ &- \rho_j^* R_i(\gamma_j^{\text{deadline}}) + \phi_{ij}^* T_{ij} - \psi_j^* + \omega_i^* \\ &+ \theta_{ij}^{*(1)} - \theta_{ij}^{*(2)} = 0 \qquad \forall i \text{ and } \forall j \end{aligned} \tag{22}$$

Solving for x_{ij}. From equation (22) we have:

$$\begin{aligned} &w_i^{\text{CPU}} - \gamma_j^{\text{CPU}} + \lambda_{ij}^* \left(w_i^{\text{CPU}} - \gamma_j^{\text{CPU}} \right) + \delta_i^* E_{ij}^{\text{comp}} \\ &- \rho_j^* R_i(\gamma_j^{\text{deadline}}) + \phi_{ij}^* T_{ij} - \psi_j^* + \omega_i^* \\ &= \theta_{ij}^{*(2)} - \theta_{ij}^{*(1)} \end{aligned} \tag{23}$$

multiplying both sides of equation (23) by x_{ij}^* we get:

$$\begin{aligned} &\Big[w_i^{\text{CPU}} - \gamma_j^{\text{CPU}} + \lambda_{ij}^* \left(w_i^{\text{CPU}} - \gamma_j^{\text{CPU}} \right) + \delta_i^* E_{ij}^{\text{comp}} \\ &- \rho_j^* R_i(\gamma_j^{\text{deadline}}) + \phi_{ij}^* T_{ij} - \psi_j^* + \omega_i^* \Big] x_{ij}^* \\ &= x_{ij}^* \theta_{ij}^{*(2)} - x_{ij}^* \theta_{ij}^{*(1)} \end{aligned} \tag{24}$$

we then substitute equations (20), and (21) in (24) we get:

$$\begin{aligned} &\Big[w_i^{\text{CPU}} - \gamma_j^{\text{CPU}} + \lambda_{ij}^* \left(w_i^{\text{CPU}} - \gamma_j^{\text{CPU}} \right) + \delta_i^* E_{ij}^{\text{comp}} \\ &- \rho_j^* R_i(\gamma_j^{\text{deadline}}) + \phi_{ij}^* T_{ij} - \psi_j^* + \omega_i^* \Big] x_{ij}^* + \theta_{ij}^{*(1)} \\ &= 0 \end{aligned} \tag{25}$$

to simplify the rest of the analysis, we replace the square-bracketed term in (25) with z.

The final bounds on the optimal value of the decision variable x_{ij}^* has multiple cases and is given by (12). ■

IV. Performance Evaluation

In this section, we evaluate the proposed **TRUE** scheme using ILP formulation and compare it to two baseline schemes. We will refer to the first baseline as **TRUE-RD**, which aims to optimize the best-fit approach based on their promised capacities. The second baseline, which we will be referring to as **TRMEED**, aims to maximize the number of replicas allocated for each task without knowing their reliability scores, aiming

Table I: Simulation Parameters

Symbol	Parameter	Value
N	Number of workers	[50 – 100]
M	Number of tasks	[1 – 30]
$\gamma_j^{\text{density}}$	Processing density per task	[1 – 5]$\times 10^2$ cycle/bit
γ_j^{data}	Data size per task	[1 – 20]MB
$\gamma_j^{\text{deadline}}$	The deadline per task	[8 – 10] ms
γ_j^{CPU}	CPU requirement for each task	[1 – 2] GHz
$E_i^{\max}$	Maximum energy consumption limit for each worker	3 mW
w_i^{CPU}	CPU Frequency per worker	[1 – 4] GHz
w_i^{tasks}	Maximum number of tasks per worker	[1 – 3] tasks
$\boldsymbol{\beta}$	Hazard rates for workers	[0.01 – 0.5]
P	Controller's transmission power	50 mW
W	Controller's total bandwidth	10×10^6 bit s^{-1}
ν	Path loss exponent	2
σ^2	Channel noise variance	1×10^{-11} W
d_i	Distance between controller c and worker w_i	[5 – 50]m
κ	On-chip capacitance factor of the worker's CPU	1×10^{-29}

to improve the chances that the assigned tasks will be completed. Multiple simulations and experiments are conducted to evaluate the system performance regarding workers' CPU utilization, overall system reliability, and the total number of replicas over varying system parameters.

A. Simulation Setup

TRUE, **TRUE−RD**, and **TRMEED** are implemented in MATLAB using the Gurobi solver [6] to find the near-optimal solution. Workers/EEDs are assumed to be any resource-limited device and can be positioned indoors or outdoors, as long as they are within the probing range of the controller c. The simulation parameters used are described in Table I.

B. Simulation Results and Analysis

We assess the performance of **TRUE**, **TRUE−RD**, and **TRMEED** in terms of different metrics by conducting six experiments.

Experiment 1. We show in Fig. 2 that changing the reliability threshold ϵ increases the total number of recruited workers and increases the overall system reliability to minimize task failure due to the worker's unreliability probability. Meaning that constraint (11d) successfully controls the system's reliability. The results were obtained using 100 workers and 15 tasks.

Experiment 2. Fig. 3 depicts the CPU gap or under-utilization, i.e., the objective function's value over the number of workers available. We can see that increasing the number of workers allows the system model to find the best-fit workers more suited for the task's requirements. The results were obtained using 30 tasks.

Experiment 3. Fig. 4 shows that the system model has successfully minimized the CPU gap between recruited workers and tasks' requests up to 14 tasks. However, the CPU gap begins to increase from this point on. Since the number of EEDs is fixed, increasing the number of tasks will make it more complex for the system model to find the optimal CPU gap value. The results were obtained using 100 workers.

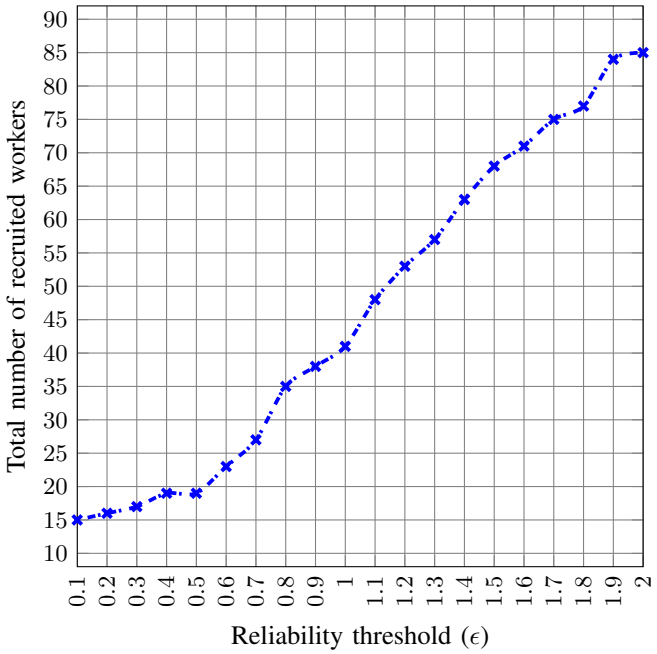

Figure 2: Impact of varying the reliability threshold ϵ on the total number of workers assigned to all tasks.

Experiment 4. We show in Fig. 5 the effect of varying the number of tasks while fixing the number of workers on the reliability score between the proposed scheme and **TRUE−RD**. In **TRUE−RD**, we do not consider the workers' reliability factor. As a result, **TRUE−RD** will recruit workers without their reliability. The figure shows that the total reliability score of the workers for **TRUE** outperforms **TRUE−RD** indicating

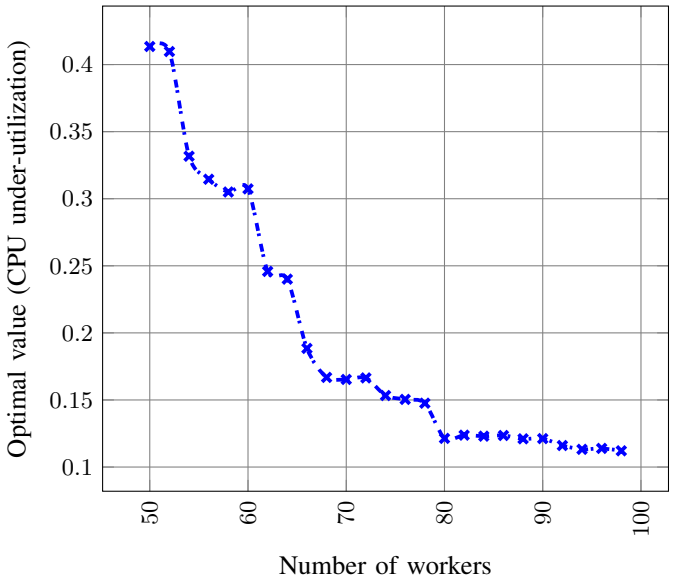

Figure 3: The impact of varying the number of workers on CPU gap (under-utilization) between the workers and the assigned tasks.

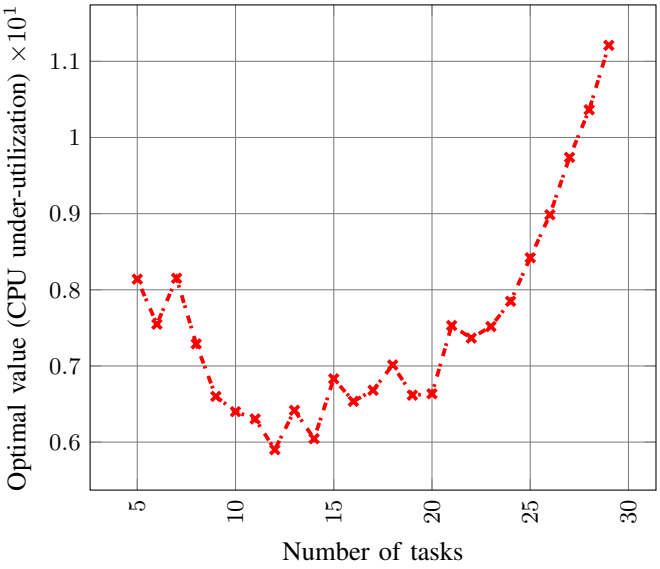

Figure 4: The effect of varying the number of tasks on CPU gap (under-utilization) of the workers.

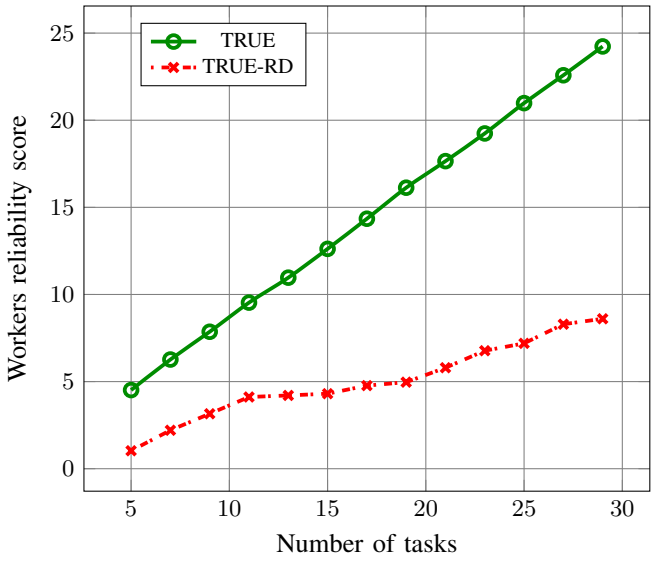

Figure 5: The impact of varying the number of tasks and fixing the number of workers on the total reliability score of the workers (higher is better).

the importance of considering the reliability of the workers in constraint (11d). The results were obtained using 100 workers and [5 – 30] tasks.

Experiment 5. **TRMEED** tends to maximize the number of replicas/workers for each requested task, **TRUE** performs better by smartly recruiting workers/replicas without over-provisioning the available workers, as depicted in Fig. 6. The number of workers/replicas can be controlled in **TRUE** efficiently by varying the reliability threshold ϵ.

Experiment 6. We evaluate the efficacy of our **TRUE** scheme against **TRUE-RD** and **TRMEED** using the percentage of failed tasks metric. In this experiment, we fix the number of workers to 100 while varying the number of tasks, and then we calculate the percentage of failed tasks. In Fig. 7, we can see that the percentage of failed tasks for **TRUE** is noticeably lower than its counterparts. We can deduce that **TRUE** maintained a constant 0% of failure for a maximum of 18 tasks. Additionally, we conclude that **TRUE** provides consistent behavior and a low percentage of failures, unlike **TRUE-RD** and **TRMEED** schemes that tend to offer inconsistent and higher percentages of failures.

We also deduce from Fig. 7 that **TRMEED** is performing better in minimizing the percentage of failed tasks than **TRUE-RD** this is because **TRMEED** aims to maximize the number of workers for each task therefore the chances of task failure is minimized.

V. Conclusions & Future Work

This paper presents Task Replication in Unreliable Edge (TRUE) networks scheme recruits, assigns, and replicates task requests to workers in an unreliable Edge network environment. The unreliability can be represented by any factor affecting the worker's availability, such as battery drainage, and connectivity loss. We firstly aim to minimize the CPU gap (under-utilization) between tasks and the target workers. Each worker has a reliability score that represents the ability of the worker to execute an assigned task successfully before a failure.

We control the overall system reliability by increasing the number of replicas for each task using a certain threshold parameter. The problem has been formulated as an ILP problem and relaxed into the equivalent convex linear program. Lower bounds on the optimal solution have been obtained using Lagrangian analysis and KKT conditions. Extensive simulations show the effectiveness of our scheme in multiple

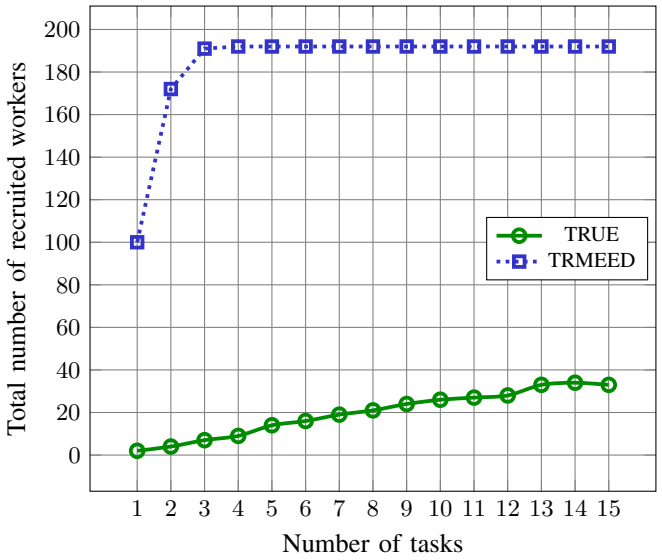

Figure 6: Impact of varying the number of tasks, while controlling the number of workers, on the total number of recruited workers.

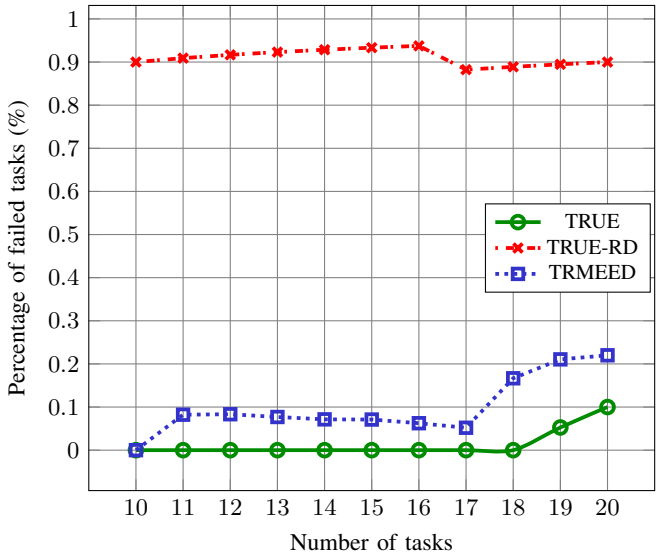

Figure 7: The percentage of failed tasks for **`TRUE`**, **`TRUE-RD`**, and **`TRMEED`** schemes while varying the number of tasks, demonstrating their reliability under varying loads (lower is better).

scenarios.

One potential direction of future work is to use stochastic optimization techniques to optimize the system uncertainty. We can also add other uncertainty factors such as the workers' communication, computation, and location unreliability.

ACKNOWLEDGMENT

This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant number ALLRP 549919-20.

REFERENCES

[1] 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, 2 2018.

[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[3] S. M. Oteafy and H. S. Hassanein, "IoT in the Fog: A Roadmap for Data-Centric IoT Development," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 157–163, 3 2018.

[4] J. Portilla, G. Mujica, J. S. Lee, and T. Riesgo, "The Extreme Edge at the Bottom of the Internet of Things: A Review," *IEEE Sensors Journal*, vol. 19, no. 9, pp. 3179–3190, 5 2019.

[5] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[6] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com

[7] X. Xu, H. Cao, Q. Geng, X. Liu, F. Dai, and C. Wang, "Dynamic resource provisioning for workflow scheduling under uncertainty in edge computing environment," *Concurrency and Computation: Practice and Experience*, p. e5674, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5674

[8] W. Chang, Y. Xiao, W. Lou, and G. Shou, "Offloading Decision in Edge Computing for Continuous Applications under Uncertainty," *IEEE Transactions on Wireless Communications*, vol. 19, no. 9, pp. 6196–6209, 9 2020.

[9] S. Ghoorchian and S. Maghsudi, "Multi-Armed Bandit for Energy-Efficient and Delay-Sensitive Edge Computing in Dynamic Networks with Uncertainty," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 279–293, 3 2021.

[10] I. M. Amer and S. Sorour, "Cost-based Compute Cluster Formation in Edge Computing," in *2022 IEEE International Conference on Communications (ICC): IoT and Sensor Networks Symposium (IEEE ICC'22 - IoTSN Symposium)*, Seoul, Korea (South), 2022.

[11] A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

[12] R. Schultz, L. Stougie, and M. H. Van Der Vlerk, "Two-stage stochastic integer programming: a survey," *Statistica Neerlandica*, vol. 50, no. 3, pp. 404–416, 1996.