QoS-based Task Replication for Alleviating Uncertainty in Edge Computing

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

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001580

***Abstract*—Edge Computing (EC) has been evolving towards harvesting latent yet underutilized computational resources of the Extreme Edge Devices (EEDs), such as autonomous vehicles, smartphones, and tablets. However, EEDs tend to be user-owned devices. This triggers a high level of uncertainty, the impact of which is mostly overlooked. Such uncertainty can stem from the potential loss of network connectivity, battery depletion, as well as the dynamic user access behavior that can affect the computational capability of EEDs and compromise the convenience of users. This uncertainty can profoundly impact the devices' reliability of executing the offloaded tasks. In this context, we propose the Replica Maximization at the Extreme Edge (RMEE) scheme. RMEE employs task replication to achieve maximum reliability and improve successful task execution while abiding by certain QoS requirements. Towards that end, RMEE aims to maximize the number of offloaded replicas for each task, while ensuring that the task execution delay is kept within a certain threshold. We formulate the task replication optimization problem as a Mixed-Integer Linear Program (MILP) and devise an analytical solution using the Karush–Kuhn–Tucker (KKT) conditions and Lagrangian analysis. Extensive simulations have shown that RMEE outperforms other baseline schemes that involve single and fixed number of replicas, in terms of drop rate, satisfaction ratio, and the number of replicas by up to 100%, 100% and 60%, and 95.1% and 85.4%, respectively.**

***Index Terms*—Edge Computing, Extreme Edge Device, Task Replication, Replica Maximization, Uncertainty**

I. Introduction

The Internet of Things (IoT) has been witnessing a significant surge; it is estimated that 125 billion IoT devices will be connected to the Internet by 2030 [1]. This excessive growth can trigger a broad spectrum of latency-sensitive and/or data-intensive IoT applications, such as Tactile Internet, smart cities, and healthcare applications [2]. Cloud Computing (CC) tends to fall short of meeting the stringent QoS requirements associated with such applications. This is since CC involves transmitting massive data to geographically remote data centers, which can cause high delay and impose heavy traffic influx at backhaul links [3].

Edge Computing (EC) has emanated as a primary enabling paradigm that can alleviate the aforementioned deficiencies of CC by fostering data processing closer to the end-users [4]. The performance gain rendered by EC has caused so much momentum that its global market size is expected to reach $10.96 billion by 2026 [5]. Recently, leveraging underutilized computational resources of IoT devices, also referred to as Extreme Edge Devices (EEDs), has been explored [6]–[8]. Harvesting the computational resources of multiple EEDs for parallel processing can significantly improve the QoS since it brings the computing service closer to the end-user, which drastically curtails the delay [9]. In addition, in contrast to existing EC platforms that are comprised of dedicated infrastructure-based edge nodes, which are entirely controlled by cloud service providers and network operators, EEDs can break this monopoly and democratize the edge, thus permitting more players to build their own edge cloud [10].

The advantageous impact of using EEDs for task offloading can be hindered by the fact that EEDs are user-owned, which imposes a high level of uncertainty. Such uncertainty ensues from multiple sources, including battery depletion, possible loss of network connectivity, hardware failure, as well as dynamic user access behavior. The latter refers to the notion that at any time, the user can access their own device to run an intensive application, such as streaming a video, playing a video game, etc. Accordingly, the computational capability of the device tends to change dynamically. To preserve its own compute power and avoid compromising the user's convenience, an EED might refrain from executing an offloaded task if it decides to run an intensive application. Note that the ensued uncertainty can affect the device's availability, and in turn impact its reliability of executing the offloaded task. Such reliability has been often overlooked in EED-enabled computing environments.

In this paper, we exploit the use of task replication to maximize reliability and improve successful task execution of offloaded tasks in EED-enabled computing environments, while abiding by certain QoS requirements. Towards that end, we propose the Replica Maximization at the Extreme Edge (RMEE) scheme. To the best of our knowledge, RMEE is the first task replication scheme that maximizes reliability in EED-enabled computing environments.

Existing replication schemes in EC have either focused on service replication [11], [12] or task replication that reinforces a fixed number of replicas [13]–[15], thus leading to under-provisioned resources. In contrast, RMEE maximizes the number of replicas allocated to EEDs for each task to minimize the drop rate resulting from uncertainty, thus maximizing reliability. In order to ensure that the demanded QoS requirements are met, RMEE performs task replication while ensuring that the response delay of each task does not exceed a certain deadline

and the energy consumed by each EED is kept within a certain limit. We formulate the task replication problem as a Mixed Integer Linear Program (MILP) and then derive the analytical solution using Karush–Kuhn–Tucker (KKT) conditions and Lagrangian analysis [16].

The remainder of this paper is organized as follows. Section II overviews some related work. Section III presents the proposed scheme (RMEE). Section IV analyzes the performance evaluation and simulation results. Section V concludes the paper and highlights future directions.

II. Related Work and Motivation

In EC, replication can be classified into Service-oriented Replication (SoR), Data-oriented Replication (DoR), and Task-oriented Replication (ToR). SoR is concerned with replicating the service that consumes storage, CPU, network, and memory, while DoR is concerned with replicating the underlying data [17]. In contrast, ToR is concerned with allocating each task to multiple edge nodes [15].

SoR has been widely explored in EC paradigms [11], [12]. For instance, in [12], the authors introduce task offloading and service replication methods on local and remote Mobile Edge Computing (MEC) servers. The scheme strives to minimize the total task response delay. However, it allocates only one replica per task, which can compromise the reliability of task execution, particularly if applied within the context of EED-enabled computing environments.

ToR has been recently investigated [13]–[15]. Sun et al. [13] introduces task replication in vehicular edge computing. The task is replicated to multiple vehicles and is marked as complete upon the first response received from either replica. The optimal number of replicas is a predefined number determined based on network conditions.

Qian et al. [14] proposes an online vehicle selection strategy for task replication in vehicular networks with unknown parameters. The scheme's objective is to minimize the task execution delay. The selection process is modeled as a budgeted multi-arm bandit problem. However, the number of task replicas is fixed using a predefined parameter.

In [15], the authors propose a learning-based task replication algorithm (LTRA) based on a combinatorial multi-armed bandit (CMAB) aiming to minimize the average offloading delay. A predefined fixed number of replicas is also adopted.

The existing body of works either adopt the use of single task replica or perform task replication using fixed number of replicas. This can cause resources under-provisioning. While such schemes can be suitable for edge computing environments with dedicated infrastructure-based edge nodes, they fail to account for the high level of uncertainty associated with EED-enabled computing environments. In contrast, our proposed scheme maximizes reliability by maximizing the number of replicas allocated to EEDs for each task while abiding to certain demanded QoS requirements.

III. Replica Maximization at the Extreme Edge (RMEE)

The **RMEE** allocates and replicates a set of requested tasks to a number of available EEDs within a given area. The objective is to maximize the number of replicas allocated to EEDs per each task. The task is considered complete upon receiving the first response among the assigned replicas.

In this section, we provide a detailed discussion of the system model, problem formulation, and analytical solution of **RMEE**.

A. System Model

Consider a set of M tasks denoted $\Gamma = \{\gamma_1, \cdots, \gamma_M\}$ and a set of N workers (i.e., EEDs) denoted $\mathcal{W} = \{w_1, \cdots, w_N\}$. Note that the terms EEDs and workers are used interchangeably throughout this paper. Each task $\gamma_j \in \Gamma$ is defined by, data size γ_j^{data} in bits, processing density $\gamma_j^{\text{density}}$ in CPU cycles/bit, i.e., the number of CPU cycles required to process a single bit of task's data, and a certain computation delay deadline $\gamma_j^{\text{deadline}}$ which is the maximum acceptable computation delay specified by the task requester. Each task γ_j needs to be executed and replicated on at least a single worker w_i. Each worker $w_i \in \mathcal{W}$ has a certain CPU clock speed w_i^{CPU} in CPU cycles/sec. The availability of each worker is known a priori. Each task $\gamma_j \in \Gamma$ has a computation delay τ_{ij}^{comp}, which is the time it takes to execute it on worker w_i. The computation delay τ_{ij}^{comp} is given by equation (1). Each worker can take on multiple tasks. The maximum number of tasks that a worker w_i can accept is defined by w_i^{tasks}. Each worker w_i has an on-board CPU capacitance denoted κ. The energy consumed by each worker w_i due to computing task γ_j, denoted E_{ij}, is given by equation (2). Note that each worker is associated with a certain battery consumption threshold, denoted $E_i^{\max}$, that should not be exceeded.

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

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

All task requests are sent to a centralized controller c that is continuously probes the area for workers. The task allocation and replication decision is made by the controller c according to the problem formulation in (3).

B. Problem Formulation

The objective is to maximize reliability (i.e., minimize drop rate) by maximizing the number of replicas assigned to EEDs for each task. We formulate the problem as a Mixed Integer Linear Program (MILP), where the number of replicas/workers reserved for a task γ_j is determined by the decision variable x_j, and the binary resource allocation decision is reflected via the decision variable λ_{ij}, where λ_{ij} is set to 1 if task γ_j is assigned to worker w_i, and 0 otherwise. The objective function in (3a) aims to maximize the total number of replicas for each task subject to the constraints below.

$$\max_{\lambda_{ij},\, x_j} \quad \sum_{j=1}^{M} x_j \tag{3a}$$

$$\text{s.t.} \quad \tau_{ij}^{\text{comp}} \lambda_{ij} \leq \gamma_j^{\text{deadline}} \qquad \forall w_i \in \mathcal{W}, \ \forall \gamma_j \in \Gamma \tag{3b}$$

$$\sum_{j=1}^{M} \lambda_{ij} E_{ij}^{\text{comp}} \leq E_i^{\max} \qquad \forall w_i \in \mathcal{W} \tag{3c}$$

$$\sum_{j=1}^{M} \lambda_{ij} \leq w_i^{\text{tasks}} \qquad \forall w_i \in \mathcal{W} \tag{3d}$$

$$\sum_{i=1}^{N} \lambda_{ij} = x_j \qquad \forall \gamma_j \in \Gamma \tag{3e}$$

$$\sum_{j=1}^{M} x_j \leq \sum_{i=0}^{N} w_i^{\text{tasks}} \tag{3f}$$

$$\lambda_{ij} \in \{0, 1\} \qquad \forall w_i \in \mathcal{W}, \ \forall \gamma_j \in \Gamma \tag{3g}$$

$$1 \leq x_j \leq N \qquad \forall \gamma_j \in \Gamma \tag{3h}$$

Constraint (3b) ensures that the computation delay τ_{ij}^{comp} of executing task γ_j on worker w_i does not exceed the specified task's deadline $\gamma_j^{\text{deadline}}$. Constraint (3c) ensures that the total energy consumed by each worker w_i due to executing all the tasks assigned to it should not exceed a certain threshold $E_i^{\max}$. Constraint (3d) ensures that no worker is assigned more than the maximum value w_i^{tasks} of allowed tasks. Constraint (3e) ensures that the total number of workers reserved for each task equals x_j. Constraint (3f) ensures that the total number of replicas assigned to all tasks does not exceed the total number of allowed tasks that the workers can execute collectively. Constraint (3g) is the integrality constraint associated with the binary decision variable λ_{ij}. Constraint (3h) sets the bounds for the decision variable x_j.

C. Analytical Solution

The aforementioned problem formulation in (3) has a linear objective function and involves integer and binary decision variables, which are x_j and λ_{ij}, respectively. The inequality constraints in (3b), (3c), (3d), and (3f) are all linear constraints. Accordingly, the problem we have at hand is a Mixed-Integer Linear Program (MILP). MILPs are generally NP-hard [18] and thus difficult to solve. Thus, we simplify the problem by relaxing the integrality constraints in order to solve it analytically. To do that, we first check the convexity of the problem in (3) by examining the inequality and equality constraints. All inequality constraints are convex except for the discrete constraint (3g). By omitting the constraint (3g) and replacing it with the continuous constraint $0 \leq \lambda_{ij} \leq 1$, all inequality constraints become convex. The equality constraints in (3e) are affine constraints, therefore the feasible region (i.e., the set of points that satisfy these constraints) is convex. Consequently, the relaxed continuous problem is convex and the analytical solution can thus be attainable. Note that closed form solutions for ILPs are generally not possible. As a result, we strive to attain a lower bound on the optimal objective value using Lagrangian analysis and KKT conditions.

To obtain the solution analytically, we can use KKT conditions to attain lower bounds on the optimal values of λ_{ij} and x_j using Lagrangian multipliers. The Lagrangian function of the relaxed program in (3) is given by equation (6) in Appendix A. The Lagrangian multipliers associated with the objective function in (3a), and the constraints in (3b) to (3h) are given by the vectors: $\boldsymbol{\lambda}, \mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\omega^{(1)}}, \boldsymbol{\omega^{(2)}}, \boldsymbol{\vartheta}, \boldsymbol{\theta^{(1)}}, \boldsymbol{\theta^{(2)}}, \boldsymbol{\xi^{(1)}}$, and $\boldsymbol{\xi^{(2)}}$, respectively.

To formalize the optimal values of the decision variables, we introduce Theorem 1 where the asterisk $(\cdot)^*$ indicates the optimal value of the super-scripted term and u denotes the replacement of the square bracketed term in equation (20) for simplification.

Theorem 1. *The optimal value of $\boldsymbol{\lambda}^*$, which is responsible for assigning task γ_j to worker w_i, is given by:*

$$\lambda_{ij}^* = \begin{cases} 0 & \theta_{ij}^{*(1)} = 0 \text{ and } u > 0 \\ 1 & \theta_{ij}^{*(1)} > 0, \ u > 0, \\ & \text{and } \theta_{ij}^{*(1)} \approx u \\ \frac{\gamma_j^{deadline}}{\tau_{ij}^{comp}} & \alpha_{ij}^* > 0 \\ \frac{E_i^{\max} - \sum_{1 \leq l \leq M \atop kl \neq ij} \lambda_{kl} E_{kl}^{comp}}{E_{ij}^{comp}} & \beta_i^* > 0 \\ w_i^{tasks} - \sum_{1 \leq l \leq M \atop kl \neq ij} \lambda_{kl} & \mu_i^* > 0 \\ x_j - \sum_{1 \leq l \leq M \atop kl \neq ij} \lambda_{kl} & \omega_j^{*(1)} > 0 \end{cases} \tag{4}$$

The optimal value of $\mathbf{x}$ which is responsible for determining the exact number of replicas allocated for task γ_j is given by:

$$x_j^* = -\omega_j^{*(1)} \sum_{i=1}^{n} \lambda_{ij} + \omega_j^{*(2)} \sum_{i=1}^{n} \lambda_{ij} - \vartheta^* \left(\sum_{i=0}^{n} w_i^{tasks} - \sum_{k \neq j}^{m} x_k \right) - \xi_j^{*(1)} + N \xi_j^{*(2)} \quad \forall i, \ \forall j \tag{5}$$

Proof. According to the bounds obtained in equations (4), and (5), the closed form solution is not attainable because each decision variable's bound is expressed in terms of other unknown variables. Hence, we use Gurobi optimizer [19] to find the near-optimal solution. The proof of Theorem 1 and the full analytical solution can be found in Appendix A. ■

IV. Performance Evaluation

We evaluate the proposed **RMEE** scheme using the MILP formulation and compare it to two baseline approaches that represent resource allocation schemes with single task replicas (i.e., no task replication) [12] or fixed number of replicas [13], [14]. For simplicity, we refer to the former as Replication-Single Worker (**Rep-SW**) [12], and the latter as Replication-K Workers (**Rep-KW**) where K is a predetermined number of replicas. We assume that each worker has a probability of failure. The cause of failure can be of any type; a connection loss, battery drainage of the worker, dynamic user access behavior, etc. The failure probabilities of workers are exponentially distributed with time t. We use the following performance metrics: 1) the total number of task replicas that are allocated to workers, 2) the drop rate, which is the ratio of

Table I: Simulation Parameters

Symbol	Parameter	Value
N	Number of workers	110
M	Number of tasks	65
$\gamma_j^{\text{density}}$	Processing density per task	[1 – 5]×10^2 cycle/bit
γ_j^{data}	Data size per task	[1 – 20] MB
$\gamma_j^{\text{deadline}}$	The deadline per task	[3 – 5] ms
$E_i^{\max}$	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
κ	On-chip capacitance factor of the worker's CPU	1×10^{-29}

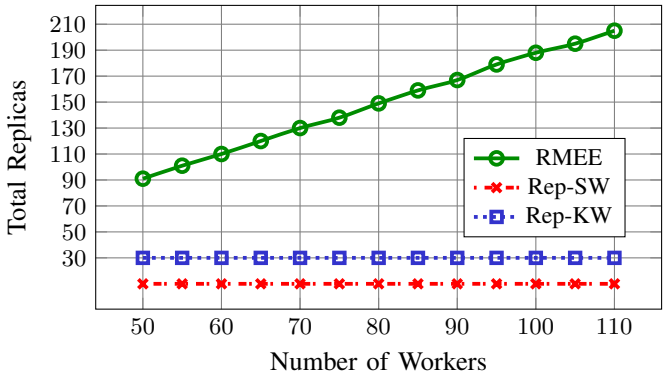

Figure 1: Performance results of **RMEE**, **Rep-SW**, and **Rep-KW** in terms of the total number of task replicas while varying the number of workers.

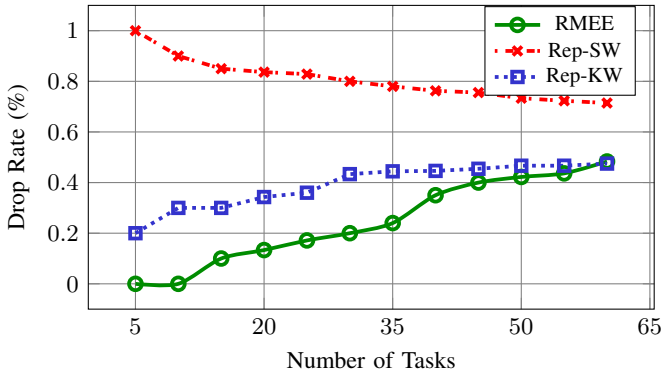

Figure 2: Task requests drop rate for **RMEE**, **Rep-SW**, and **Rep-KW** schemes while varying the number of tasks (lower values are better).

the number of tasks that could not be successfully executed, due to worker failure, to the total number of tasks, and 3) the satisfaction ratio, which is the ratio of the number of tasks that could be executed before the corresponding deadline to the total number of tasks.

A. Simulation Setup

RMEE, **Rep-SW**, and **Rep-KW** are implemented in MATLAB using the Gurobi solver [19] to find the near-optimal solution. Workers are assumed to be any resource-limited device, which can be positioned indoors or outdoors, as long as it is within the probing range of the controller c. Unless otherwise specified, the simulation parameters used are listed in Table I. The value of K is set to 3 in **Rep-KW** (i.e., 3 replicas per task).

B. Simulation Results and Analysis

We conducted four experiments to assess the performance of **RMEE** compared to **Rep-SW** and **Rep-KW** using different metrics previously mentioned in Section IV.

Experiment 1. Fig. 1 depicts the total number of task replicas rendered by **RMEE** over varying the number of workers. Note that the total replicas remain constant over the varying number of workers in both **Rep-SW** and **Rep-KW** since they adopt a single and fixed number of replicas, respectively. In contrast, as the number of workers increases in **RMEE**, the total number of task replicas increases. The percentage of change is 95.1% and 85.4% compared to **Rep-SW**, and **Rep-KW**, respectively.

Experiment 2. Fig. 2 illustrates the performance of **RMEE**, **Rep-SW**, and **Rep-KW** in terms of the drop rate over the number of tasks. As shown in Fig. 2, **RMEE** yields the lowest drop rate compared to the other schemes, with an improvement of up to 100% compared to **Rep-SW**, and **Rep-KW**. This is because **RMEE** maximizes the number of replicas assigned to each task, which maximizes the reliability by maximizing the chance of executing the task in the case of worker failure. In contrast, **Rep-SW** uses only one replica per task, while **Rep-KW** enables the use of K replicas per task. This indicates that if the worker to which a task is assigned fails, the task has no chance of being executed in **Rep-SW**, and a slightly higher but limited chance in **Rep-KW**. As depicted in Fig. 2, the drop rate rendered by **RMEE** increases as the number of tasks increases. This can be attributed to the fact that as the number of tasks increases, a lower number of replicas is allowed per each task, reducing the chance of successfully executing the task in case of multiple workers failure.

Experiment 3. Fig. 3 shows the satisfaction ratio of **RMEE**, **Rep-SW**, and **Rep-KW** over the number of tasks. As shown in Fig. 3, **RMEE** has the highest satisfaction ratio compared to the other schemes, with an improvement of up to 100% and

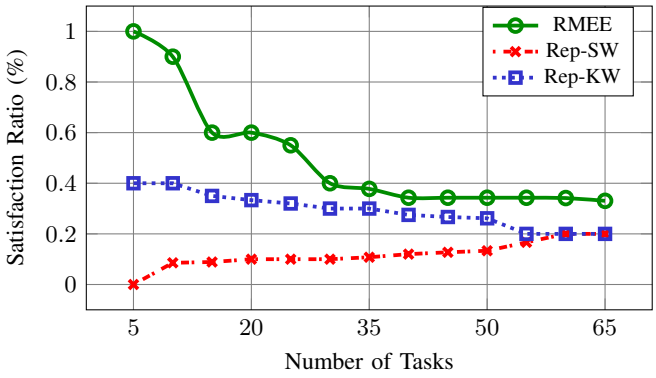

Figure 3: Performance results of **RMEE**, **Rep−SW**, and **Rep−KW** in terms of the satisfaction ratio over varying number of tasks (higher values are better).

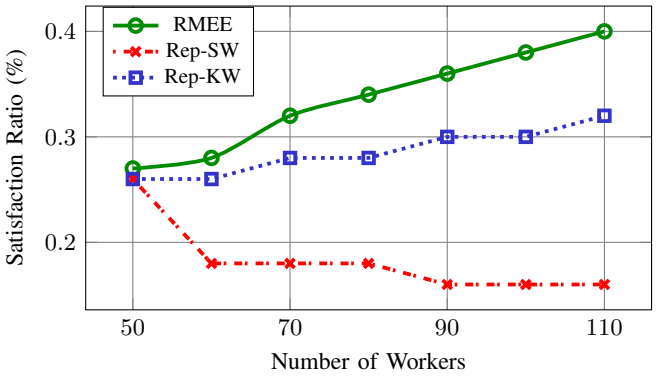

Figure 4: Performance results of **RMEE**, **Rep−SW**, and **Rep−KW** in terms of the satisfaction ratio over varying number of workers (higher values are better).

60% compared to **Rep−SW**, and **Rep−KW**, respectively. **RMEE** scheme has the highest ratio because it maximizes the number of replicas assigned to each task, decreasing the chance of worker failure. We can also conclude from Fig. 3 that the satisfaction ratio of **RMEE** decreases as the number of tasks increases. This is because increasing the number of tasks while fixing the number of workers, lowers the number of replicas that are assigned to each task, which reduces the chance of task completion in case of multiple workers' failure. Note that in Fig. 3, the satisfaction ratio of **Rep−SW** increases as the number of tasks increases because as the number of tasks increase, the ratio of completed tasks of **Rep−SW** increase.

Experiment 4. Fig. 4 evaluates the satisfaction ratio of **RMEE**, **Rep−SW**, and **Rep−KW** over varying the number of workers. **RMEE** has the highest satisfaction ratio compared to the other schemes with an improvement of up to 60%, and 20% compared to **Rep−SW**, and **Rep−KW**, respectively. As depicted in Fig. 4, the satisfaction ratio of **RMEE** increases as the number of workers increases. This is attributed to the fact that as the number of workers increases, a higher number of replicas is assigned to each task, which increases the reliability of the workers recruited and minimizes the chance of multiple workers' failure. We can also notice from Fig. 4 that the satisfaction ratio for **Rep−SW** decreases as the number of workers increases. The reason for this is, that as the number of workers increases, workers with chances of failure increase and that causes the satisfaction ratio of **Rep−SW** to drop because it utilizes a single worker for each task.

V. Conclusion and Future Work

In this paper, we have proposed a task replication scheme called RMEE to foster the notion of democratizing the edge and leveraging the use of underutilized resources at the extreme edge. RMEE accounts for the high level of uncertainty in EED-based computing environments. In particular, RMEE maximizes reliability, and thus increases the chance of successful task execution in such environments by maximizing the number of replicas assigned to workers per task. We formulated the task replication problem as a Mixed Integer-Linear Program (MILP) and then relaxed the problem to a convex Linear Program (LP). Upper bounds on the optimal solution have been attained, accompanied by the analytical solution using KKT optimality conditions and Lagrangian analysis. Extensive simulations have been conducted to assess the performance of RMEE compared to two other baseline approaches; one is based on using a single replica per task, while the other adopts the use of a predetermined fixed number of replicas. Simulations show that RMEE outperforms the former and latter schemes in terms of drop rate, satisfaction ratio, and the number of replicas by up to 100%, 100% and 60%, and 95.1% and 85.4%, respectively.

In the future, we will quantify the level of uncertainty associated with each worker using prediction and stochastic techniques.

Acknowledgment

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

References

[1] K. Gyarmathy, "Comprehensive guide to iot statistics you need to know in 2020," *VXchnge [online]. Tampa, Florida: vXchnge, 2020 (1), 3 [cit. 2020-07-10]. Dostupné z: https://www. vxchnge. com/blog/iot-statistics*, 2020.

[2] J. B. Awotunde, A. K. Bhoi, and P. Barsocchi, "Hybrid cloud/fog environment for healthcare: An exploratory study, opportunities, challenges, and future prospects," in *Hybrid Artificial Intelligence and IoT in Healthcare*. Springer, 2021, pp. 1–20.

[3] Y. Zhao, W. Wang, Y. Li, C. Colman Meixner, M. Tornatore, and J. Zhang, "Edge Computing and Networking: A Survey on Infrastructures and Applications," *IEEE Access*, vol. 7, pp. 101 213–101 230, 2019.

[4] B. Varghese, E. De Lara, A. Y. Ding, C. H. Hong, F. Bonomi, S. Dustdar, P. Harvey, P. Hewkin, W. Shi, M. Thiele, and P. Willis, "Revisiting the Arguments for Edge Computing Research," *IEEE Internet Computing*, vol. 25, no. 5, pp. 36–42, 2021.

[5] H. Stipp, "Edge computing: Global market size 2026 — Statista." [Online]. Available: https://www.statista.com/statistics/948762/worldwide-edge-computing-market-size-forecast/

[6] 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.

[7] E. M. Dogo, A. F. Salami, C. O. Aigbavboa, and T. Nkonyana, "Taking cloud computing to the extreme edge: A review of mist computing for smart cities and industry 4.0 in africa," *EAI/Springer Innovations in Communication and Computing*, pp. 107–132, 2019. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-99061-3_7

[8] N. Bruschi, A. Garofalo, F. Conti, G. Tagliavini, and D. Rossi, "Enabling mixed-precision quantized neural networks in extreme-edge devices," *17th ACM International Conference on Computing Frontiers 2020, CF 2020 - Proceedings*, pp. 217–220, 5 2020.

[9] R. K. Barik, A. C. Dubey, A. Tripathi, T. Pratik, S. Sasane, R. K. Lenka, H. Dubey, K. Mankodiya, and V. Kumar, "Mist Data: Leveraging Mist Computing for Secure and Scalable Architecture for Smart and Connected Health," *Procedia Computer Science*, vol. 125, pp. 647–653, 1 2018.

[10] R. Tourani, S. Srikanteswara, S. Misra, R. Chow, L. Yang, X. Liu, and Y. Zhang, "Democratizing the Edge: A Pervasive Edge Computing Framework," 7 2020. [Online]. Available: https://arxiv.org/abs/2007.00641v1

[11] I. Farris, T. Taleb, A. Iera, and H. Flinck, "Lightweight service replication for ultra-short latency applications in mobile edge networks," *IEEE International Conference on Communications*, 7 2017.

[12] S. A. Mohamed, S. Sorour, and H. S. Hassanein, "Group-Delay Aware Task Offloading with Service Replication for Scalable Mobile Edge Computing," *2020 IEEE Global Communications Conference, GLOBECOM 2020 - Proceedings*, vol. 2020-January, 12 2020.

[13] Y. Sun, S. Zhou, and Z. Niu, "Distributed Task Replication for Vehicular Edge Computing: Performance Analysis and Learning-Based Algorithm," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1138–1151, 2 2021.

[14] Y. Qian, Z. Zuo, and Y. Hao, "Online vehicle selection for task replication via bandit learning," *Proceedings - 2021 IEEE 45th Annual Computers, Software, and Applications Conference, COMPSAC 2021*, pp. 1627–1632, 7 2021.

[15] Y. Sun, J. Song, S. Zhou, X. Guo, and Z. Niu, "Task Replication for Vehicular Edge Computing: A Combinatorial Multi-Armed Bandit Based Approach," *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings*, 2018.

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

[17] S. Slimani, T. Hamrouni, and F. Ben Charrada, "Service-oriented replication strategies for improving quality-of-service in cloud computing: a survey," *Cluster Computing*, vol. 24, no. 1, pp. 361–392, 3 2021. [Online]. Available: https://link.springer.com/article/10.1007/s10586-020-03108-z

[18] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*. Berlin: Springer, 1 2003, vol. 24. [Online]. Available: https://link.springer.com/book/9783540443896

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

APPENDIX A
PROOF OF THEOREM 1

The Lagrangian function associated with the relaxed program in (3) is given by:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\mu}, \boldsymbol{\omega^{(1)}}, \boldsymbol{\omega^{(2)}}, \boldsymbol{\vartheta}, \boldsymbol{\theta^{(1)}}, \boldsymbol{\theta^{(2)}}, \boldsymbol{\xi^{(1)}}, \boldsymbol{\xi^{(2)}}) = \\ \sum_{j=1}^{M} x_j - \Big(\sum_{i=1}^{N} \sum_{j=1}^{M} \alpha_{ij} \left(\tau_{ij}^{\text{comp}} \lambda_{ij} - \gamma_j^{\text{deadline}} \right) \\ + \sum_{i=1}^{N} \beta_i \left(\sum_{j=1}^{M} \lambda_{ij} E_{ij}^{\text{comp}} - E_i^{\max} \right) + \sum_{i=1}^{N} \mu_i \left(\sum_{j=1}^{M} \lambda_{ij} - w_i^{\text{tasks}} \right) \\ + \sum_{j=1}^{M} \omega_j^{(1)} \left(\sum_{i=1}^{N} \lambda_{ij} - x_j \right) + \sum_{j=1}^{M} \omega_j^{(2)} \left(x_j - \sum_{i=1}^{N} \lambda_{ij} \right) \\ + \vartheta \left(\sum_{i=0}^{N} w_i^{\text{tasks}} - \sum_{j=1}^{M} x_j \right) \\ + \sum_{i=1}^{N} \sum_{j=1}^{K} \theta_{ij}^{(1)} \left(\lambda_{ij} - 1 \right) - \theta_{ij}^{(2)} \lambda_{ij} \\ + \sum_{j=1}^{M} \xi_j^{(1)} \left(1 - x_j \right) + \sum_{j=1}^{M} \xi_j^{(2)} \left(x_j - N \right) \Big)\end{aligned} \tag{6}$$

Applying KKT optimality conditions to the constraints in (3):

$$\alpha_{ij}^* \left(\tau_{ij}^{\text{comp}} \lambda_{ij} - \gamma_j^{\text{deadline}} \right) = 0 \quad \forall i \text{ and } \forall j \tag{7}$$

$$\beta_i^* \left(\sum_{j=1}^{M} \lambda_{ij} E_{ij}^{\text{comp}} - E_i^{\max} \right) = 0 \quad \forall i \tag{8}$$

$$\mu_i^* \left(\sum_{j=1}^{M} \lambda_{ij} - w_i^{\text{tasks}} \right) = 0 \quad \forall i \tag{9}$$

$$\omega_j^{*(1)} \left(\sum_{i=1}^{N} \lambda_{ij} - x_j \right) = 0 \quad \forall j \tag{10}$$

$$\omega_j^{*(2)} \left(x_j - \sum_{i=1}^{N} \lambda_{ij} \right) = 0 \quad \forall j \tag{11}$$

$$\vartheta^* \left(\sum_{i=0}^{N} w_i^{\text{tasks}} - \sum_{j=1}^{M} x_j \right) = 0 \tag{12}$$

$$\theta_{ij}^{*(1)} \left(\lambda_{ij} - 1 \right) = 0 \quad \forall i \text{ and } \forall j \tag{13}$$

$$\theta_{ij}^{(2)} \lambda_{ij} = 0 \quad \forall i \text{ and } \forall j \tag{14}$$

$$\xi_j^{(1)} \left(1 - x_j \right) = 0 \quad \forall j \tag{15}$$

$$\xi_j^{(2)} \left(x_j - N \right) = 0 \quad \forall j \tag{16}$$

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

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \lambda_{ij}} &= \omega_j^{*(2)} - \omega_j^{*(1)} - \mu_i^* - \beta_i^* E_{ij}^{\text{comp}} - \alpha_{ij}^* \tau_{ij}^{\text{comp}} \\ &\quad - \theta_{ij}^{*(1)} + \theta_{ij}^{*(2)} = 0 \ \ \forall i \text{ and } \forall j\end{aligned} \tag{17}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial x_j} &= 1 + \omega_j^{*(1)} - \omega_j^{*(2)} + \vartheta^* + \xi_j^{*(1)} \\ &\quad - \xi_j^{*(2)} = 0 \ \ \forall j\end{aligned} \tag{18}$$

Solving for λ_{ij}^*. From equation (17) we have:

$$\begin{aligned}\omega_j^{*(2)} - \omega_j^{*(1)} - \mu_i^* - \beta_i^* E_{ij}^{\text{comp}} - \alpha_{ij}^* \tau_{ij}^{\text{comp}} = \\ \theta_{ij}^{*(1)} - \theta_{ij}^{*(2)}\end{aligned} \tag{19}$$

Multiplying both sides of (19) by λ_{ij}^* then using equations (13) and (14), we get:

$$\left[\omega_j^{*(2)} - \omega_j^{*(1)} - \mu_i^* - \beta_i^* E_{ij}^{\text{comp}} - \alpha_{ij}^* \tau_{ij}^{\text{comp}} \right] \lambda_{ij}^* = \theta_{ij}^{*(1)} \tag{20}$$

To simplify the rest of the analysis for λ_{ij}^*, we replace the square bracketed term in (20) with the variable u. The optimal value of λ_{ij}^* has multiple possible cases can be found in equation (4).

The lower bound on x_j^* is obtained by multiplying (18) by x_j^*:

$$\begin{aligned}x_j^* + \omega_j^{*(1)} x_j^* - \omega_j^{*(2)} x_j^* + \vartheta^* x_j^* + \xi_j^{*(1)} x_j^* \\ - \xi_j^{*(2)} x_j^* = 0\end{aligned} \tag{21}$$

Plugging equations (10) to (12), (15), and (16) into (21) we get an upper bound on the optimal value of the variable x_j^* that is given by equation (5).