ReLIEF: A Reinforcement-Learning-Based Real-Time Task Assignment Strategy in Emerging Fault-Tolerant Fog Computing

Roozbeh Siyadatzadeh, Fatemeh Mehrafrooz, Mohsen Ansari, Bardia Safaei, Muhammad Shafique, *Senior Member, IEEE*, Jörg Henkel, *Fellow, IEEE*, and Alireza Ejlali

***Abstract*—Due to the real-time requirements in several IoT applications, fog computing has emerged to overcome the long latency and other constraints of cloud computing. Due to the high probability of packet loss, energy limitation of IoT devices, and the external disturbances that may frequently occur on the fog infrastructure, the timing constraints of real-time tasks may be compromised. Therefore, the reliability of executing real-time tasks has always been a significant challenge in fog computing. In addition to the correct execution of the tasks, it is also important to execute them before their deadlines according to their real-time classification. State-of-the-art methods generally focus on the delay or functionality of tasks in fog computing systems. However, those methods do not widely focus on the reliability of tasks with real-time constraints in dynamic environments. In this article, a novel primary backup task assignment strategy based on machine learning (ReLIEF) is proposed to improve the reliability of fog-based IoT systems. To identify suitable nodes for the execution of the primary and backup tasks, ReLIEF employs a reinforcement learning (RL) approach, which has an outstanding performance in dynamic environments by establishing a balance between communication delay and workload on each fog device. Based on the simulations, our newly proposed technique has been able to reduce the amount of task dropping rate by up to 84% against the state of the art. Moreover, it is capable of balancing the workload distribution while increasing the reliability of the system by nearly 72% compared with its counterparts.**

***Index Terms*—Fog computing, Internet of Things (IoT), reinforcement learning (RL), reliability, resource allocation.**

I. Introduction

THE Internet of Things (IoT) has turned our world into a place where things can communicate with one another without human intervention. This technology refers to an infrastructure that establishes Internet-based communications between an enormous number of resource-constrained embedded devices. These devices are significantly restricted in terms of their processing capabilities, power resources, communication coverage, storage, etc. [1]. Similar to cyber–physical systems (CPSs) [2], the employed embedded devices in these systems can collect and process the gathered information from their physical environment [3]. Nowadays, IoT is being widely used in a wide variety of applications, including but not limited to, smart agriculture, smart cities, smart homes, smart healthcare, automotive, and logistics [4]. The increasing trend in the number of connected devices is followed by a significant rise in the volume of generated information, which should be later submitted to processing components.

Due to the resource restrictions in IoT devices, they have constrained processing capabilities and are unable to process this massive information [5], [6]. Cloud computing has provided an appropriate platform for processing and storing the generated information by IoT devices [7]. However, the physical distance between the IoT nodes and the cloud servers may result in a considerable amount of delay [8]. As many IoT applications are composed of tasks that are real time and should be completed before their deadlines, the delay of cloud computing is incompatible with these types of applications [9], [10].

To address the limitations of cloud computing, fog computing was introduced to mitigate the amount of delay by executing a number of tasks near the network's edge without transmitting them to cloud servers [11]. As a result, the delay of fog computing is less than cloud computing, and it is more appropriate for real-time applications with a short deadline. Therefore, a three-tier architecture is introduced such that the first layer is composed of IoT nodes, the second layer is made up of fog nodes, and the last layer contains cloud servers [12].

Although fog computing reduces the delay, the distributed and open structure nature of fog-based IoT systems makes them vulnerable and prone to various faults [13], which might occur during the operation of fog nodes, preventing them from functioning properly. Moreover, unreliable communication links can result in tasks not being received at the destination fog nodes. Consequently, reliability management techniques must be utilized in fog computing. To improve reliability and ensure that tasks are completed prior to their deadline in the fog layer, this article proposes ReLIEF, *a learning-based primary–backup strategy* for increasing the reliability of executing the tasks in the fog nodes. In this novel approach, the selection process of the primary and backup

Manuscript received 16 June 2022; revised 4 November 2022 and 4 January 2023; accepted 23 January 2023. Date of publication 26 January 2023; date of current version 7 June 2023. *(Corresponding author: Mohsen Ansari.)*

Roozbeh Siyadatzadeh, Fatemeh Mehrafrooz, Mohsen Ansari, Bardia Safaei, and Alireza Ejlali are with the Department of Computer Engineering, Sharif University of Technology, Tehran 1458889694, Iran (e-mail: roozbeh.siyadatzadeh@sharif.edu; fatemeh.mehrafrooz@sharif.edu; ansari@sharif.edu; bardiasafaei@sharif.edu; ejlali@sharif.edu).

Muhammad Shafique is with the Division of Engineering, New York University Abu Dhabi, Abu Dhabi, UAE (e-mail: muhammad.shafique@nyu.edu).

Jörg Henkel is with the Department of Computer Science, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany (e-mail: henkel@kit.edu).

Digital Object Identifier 10.1109/JIOT.2023.3240007

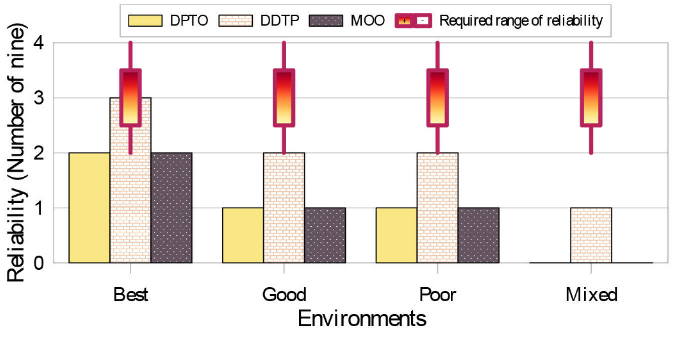


Fig. 1. Motivational example of a factory that needs a specific range of reliability levels in different environmental situations. In this example, DPTO, DDTP, and MOO are compared with the same number of tasks for 1500 times, and the average result is reported. The terms “Best,” “Good,” “Poor,” and “Mixed” refer to fog node conditions with varying degrees of failure rate.

nodes for every task is determined by a machine-learning model based on the reliability of the system, the workload of the fog nodes, and the deadline of the tasks.

A. Motivation

As illustrated in Fig. 1, the reliability of different components and environments in an IoT-enabled smart factory is demonstrated. The red lines indicate the required range of reliability levels of this factory. The terms *Best, Good, Poor,* and *Mixed* denote different environments with varying degrees of fault probability. *Best* depicts an environment with a low probability of fault occurrence, whereas *Mixed* depicts an environment with a high probability of fault occurrence.

As seen in Fig. 1, as the fault rate increases, the reliability of all approaches degrades. Therefore, in situations where various components, such as communication links or processing units, are prone to malfunction, a solution that can tolerate a high rate of failures is required. Nowadays, IoT is widely used in critical aspects of human life. From personal life to industry, the reliability of such systems is important and must be considered. Therefore, fault-tolerant techniques need to be utilized in IoT. Additionally, when tasks are real time, additional constraints, such as the task’s deadline, must be considered. Thus, typical strategies for improving reliability could be used to address the deadlines associated with these tasks. As a result, we proposed a strategy for achieving a higher level of reliability utilizing reinforcement learning (RL) and a primary backup strategy that takes all of these constraints into account.

RL has several advantages over classic approaches. Many of these classic algorithms for tackling optimization problems rely on handcrafted heuristics that successively develop a solution. These heuristics are created by domain specialists and frequently perform poorly due to the complexity of the underlying problems. RL provides a viable alternative to automate the search and outperform the heuristic methods, specifically in large search spaces, by supervising or self-supervising the training of an agent. Also, the other benefit of utilizing RL in such a problem is the ability to adapt to a wide range of problems effortlessly and automatically. ReLIEF uses a reinforcement-based technique to offload tasks in a manner that improves reliability. It is highly adaptive to a wide range

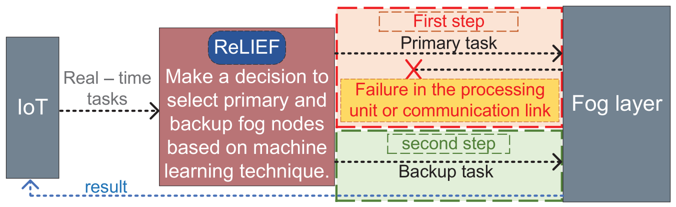

IoT
Real – time tasks
ReLIEF
Make a decision to select primary and backup fog nodes based on machine learning technique.
First step
Primary task
Failure in the processing unit or communication link
second step
Backup task
Fog layer
result

Fig. 2. Abstract view of novel contribution.

of applications and environments. As mentioned earlier, this is the advantage of RL compared to other techniques that can actively explore their environments. Therefore, ReLIEF is applicable to fog-based IoT systems with more than two fog devices needing reliability improvement.

B. Contributions and Novelty

ReLIEF considers two types of reliability: 1) functional reliability and 2) timing reliability. The former indicates that each task must be executed correctly, while the latter means that every task must be executed before its deadline. This method can tolerate both types of permanent and transient faults. It can also satisfy the real-time constraints of every task at runtime. Based on an extensive set of simulations, ReLIEF outperforms the state of the art in terms of reliability, workload distribution, and dynamic adaption for a different number of tasks. In this regard, ReLIEF has improved reliability by up to 72% against the state of the art, while it has reduced the tasks’ drop rate by up to 68%. As has been illustrated in Fig. 2, the main contributions of ReLIEF are listed as follows.

1) The dynamic selection of appropriate fog nodes for the primary and backup tasks with an RL technique in runtime, to the best of our knowledge, ReLIEF is the only study that uses a learning-based primary–backup technique to tolerate the failure of communication links and processing units, in addition to considering the real-time constraints determined for the tasks in a fog-based IoT system.
2) Employing a delayed backup-dropping strategy to reduce processing and network overhead, as well as power consumption. This indicates that backup tasks are only sent to fog nodes when there is only a time interval left before their deadline. This period of time is longer than the round trip time (RTT), plus the processing time of the task. Therefore, the backup task will have enough time to be completed.
3) Due to the fact that the ReLIEF is based on machine learning, it is highly adaptable, and it can operate in a variety of settings with a variety of system architectures independent of the number of fog nodes or workloads.
4) The effectiveness of ReLIEF is evaluated through a wide range of experiments. These experiments show that ReLIEF performs better in workload balancing, reliability, and delay compared to other state-of-the-art methods.

Article Organization: The remainder of this article is organized as follows. Section II gives a brief overview of the related studies in fog-based IoT systems and an explanation of RL with focusing on *Q*-learning. Section III introduces the system model. In Section IV, the structure of ReLIEF will be described in detail. The evaluation of our primary–backup technique and its comparison with the state of the art is discussed in Section V. Finally, this article is concluded in Section VI.

II. Background and Related Work

A. Related State-of-the-Art Techniques

This section describes state-of-the-art techniques and their limitations based on delay, energy, and reliability. The majority of research has been on energy and delay issues in fog-based systems. For instance, Min et al. employed energy harvesting to power IoT nodes. They suggested *Q*-learning-based and deep reinforcement learning-based algorithms to determine the offloading rate and offloading destination [14].

In another study, with the help of the Lyapunov optimization technique, an online algorithm is proposed in order to reduce total energy consumption while reducing the average service delay in fog nodes [15]. Hu et al. [16] provided a method with the use of two heuristics to schedule real-time tasks in a manner to reduce energy consumption. Abdel-Basset et al. [10] proposed a metaheuristic method based on the Harris Hawk optimization algorithm for scheduling tasks on virtual machines (VMs) in a fog environment to optimize energy consumption, makespan, cost, and carbon dioxide emission rate.

Recently, a number of techniques and algorithms for real-time tasks in fog computing have been proposed in various publications [24], [25], [26]. The proposed strategy introduced by [17] prioritizes tasks according to their deadline. After prioritizing, the algorithm chooses the ideal computing device based on resource availability and transmission time. Swain et al. [18] suggested a task offloading technique based on matching theory to reduce latency, total system energy consumption, and the number of tasks exceeding the deadline in an IoT–fog network. In the fog federation framework, Sarkar et al. [19] developed a deadline-aware task offloading mechanism. This technique classifies real-time tasks according to their task utilization and then assigns them to fog devices with the goal of decreasing latency and meeting deadlines. Another strategy is proposed in the fog–cloud environment by Adhikari et al. [27]. In this strategy, a suitable device for processing a real-time task is selected based on resource utilization and total cost using the accelerated particle swarm optimization technique. Hazra et al. [8] proposed a heuristic-based transmission scheduling and computation offloading technique for real-time tasks in industrial fog networks.

In fog computing, the tradeoff between reliability and system cost is also an important issue. Ghanavati et al. [20] proposed a runtime task scheduling mechanism for fog computing platforms to be fault tolerant while maintaining a low cost of fog resources and optimizing QoS and energy consumption. In this work, when the waiting time for processing expires and the node does not respond, a backup strategy is exploited. Yao and Ansari [28] proposed a modified best fit decreasing algorithm for allocating tasks to VMs in the fog node in order to optimize system reliability while minimizing system costs.

TABLE I
Comparison of Existing Strategies for Task Placement and Reliability Management

Existing works	Reliability	Real-time tasks	Delay	Workload balancing	Learning-based
[14]	✗	✗	✓	✗	✓
[10]	✗	✗	✓	✓	✗
[17]	✓	✓	✓	✗	✗
[18]	✓	✓	✓	✗	✗
[19]	✓	✓	✓	✗	✗
[8]	✓	✓	✓	✗	✗
[20]	✓	✗	✓	✗	✓
[21]	✓	✗	✓	✗	✗
[22]	✓	✗	✓	✗	✓
[23]	✓	✗	✓	✗	✗
Our work	✓	✓	✓	✓	✓

Some research on fog and cloud-based networks takes into account both delay and reliability. A dynamic computational offloading method based on Lyapunov's optimization theory is proposed in [29]. In terms of both task running delay and task running failure, it optimized offloading options while also minimizing the amount of time spent processing tasks and ensuring that the data transfer procedure was successful. Hou et al. [21] suggested a mechanism for task allocation that decreased energy consumption while maintaining delay and reliability. They solved it using the proximal Jacobi alternating direction method of multipliers. Wang et al. [23] suggested an offloading policy for heterogeneous fog nodes and a remote cloud that takes into account the reliability of communication and delay-sensitive requirements. In another research, Zhu et al. [30] proposed a task scheduling strategy in a multicloud environment that optimizes the delay and total cost while considering security and reliability constraints. Sun et al. [22] presented a distributed task replication scheme for vehicular edge computing that enables any vehicle to distribute task replicas among many candidate nodes in order to reduce the average offloading delay under task failure constraints. As it is illustrated in Table I, few works paid attention to the failure in real-time tasks, but still, there are open challenges in the reliability of fog-based IoT systems with a dynamic number of fog nodes and tasks in runtime. To address these issues, in this article, we developed a learning-based technique that considers the reliability of tasks in both terms of deadline and functionality by using the primary–backup technique in fog-based IoT systems. Moreover, as we utilize RL to select appropriate fog nodes, this method can adapt dynamically to the different number of fog nodes and tasks. The workload distribution is also improved over previous works. Due to this improved workload distribution, the number of tasks in the queues decreases. Therefore, the delay reduces, and more tasks could be completed before the deadline.

B. Overview of Q-Learning

RL is the process of learning what to perform in order to maximize a numerical reward signal [31]. The learner is not informed about taking which action; rather, it must experiment to determine which acts produce the greatest reward.

In the most fascinating and demanding circumstances, actions may have an effect on not only the immediate reward but also the subsequent situation and, thus, all subsequent rewards. These two characteristics (trial-and-error, search, and delayed reward) are the two most important distinguishing features of RL. The fundamental principle behind RL is to capture the most critical components of the real problem that a learning agent faces when interacting with its environment over time in order to accomplish a goal. A learning agent must be able to perceive the state of its environment, and it must be able to execute actions that have an impact on the state of the environment. Additionally, the agent must have a goal or set of goals related to the environment's state.

Q-learning is a type of model-free RL [32]. It enables agents to learn to act optimally in Markovian domains through experience with the consequences of their actions without requiring them to develop domain maps. The main parts of *Q*-learning are as follows.

1) Agent: This is the entity that makes the decisions. In this research, the agent is a broker or one of the fog nodes responsible for managing task assignments.

2) Environment: This is where the agent works. For example, a fog-based environment with all its devices can be the working environment.

3) State: This is the status of the system in its environment. In other words, the state can be considered as a signal that conveys to the agent some sense of "how the environment is" at a given time. In fog environments based on the problem that RL wants to solve.

4) Action: This is the agent's next move. It can be defined as selecting operating frequency or execution tasks for each fog node in a fog environment.

5) Reward: This is the feedback that the agent receives from the environment as a result of taking that action.

6) Reward Function: A function that calculates reward based on action and system parameters. For example, an equation with a combination of delay and energy can be used as a reward function. It is discussed in greater detail in Section IV-B3.

7) Policy: This is a mapping function between the agent's states and actions. It is a simple lookup table witch called *Q*-table. It will function as the agent's brain.

After the completion of each action, the *Q*-table updates according to the following equation:

$$Q'(s_k, a_k) = Q(s_k, a_k) + \beta_k\left[\left(r_{k+1} + \gamma \times \max_{a\in A} Q(s_{k+1}, a)\right) - Q(s_k, a_k)\right]. \quad (1)$$

Actions are selected based on Bellman's principle of optimality as follows [33]:

$$\pi(s) = \arg\max_a\left(Q'(s, a)\right) \quad (2)$$

where $Q'(s, a)$ is illustrated in (1). Moreover, $\pi(s)$ demonstrates the best action a in state s.

TABLE II
NOTATIONS UTILIZED IN THIS ARTICLE

Symbol	EXPLANATION
$\mathcal{I}$	Set of IoT devices
$\mathcal{F}$	Set of fog nodes
T	Set of Tasks
$\mathcal{B}$	Broker
i_i	*i*th IoT device
f_i	*i*th Fog node
τ_i	*i*th task
s_i	Size of the *i*th task
a_i	Arrival time of the *i*th task
l_i	Length of the *i*th task
d_i	Deadline of the *i*th task
fr_j	Frequency of the *j*th fog node
$T_{\mathcal{B}}$	Broker's processing time
$T^u_{f_j,\tau_i}$	Transmission time for the *i*th task from the broker to the *j*th fog node
$T^e_{f_j,\tau_i}$	Execution time for the *i*th task by the *j*th fog node
$T^Q_{f_j,\tau_i}$	Queuing time for the *i*th task in the *j*th fog node'queue
D_i	Total delay of the *i*th task
R^c	Computational reliability
λ_i	Failure rate of the *i*th fog node
R^l	Communication reliability
μ_i	Failure rate of the communication link between *i*th fog node and broker
R^0	Reliability of a fog node
R_i	Reliability of the *i*th task
R	Reliability of the system
TPQ	Number of tasks in the primary queue
TBQ	Number of tasks in the backup queue
W_{f_i}	Total workload in the *i*th fog node
WL	Workload balancing
s_i	State i
a_i	Action i
r_k	Reward in the stage k

III. System Architecture

In this section, we describe the system, delay, reliability, and workload model utilized in the proposed method. Table II summarizes the notations used in this article.

A. System and Workload Model

Fig. 3 illustrates the system model. The system is made from a set of fog nodes $\mathcal{F} = \{f_1, f_2, \ldots, f_{|F|}\}$, which are heterogeneous, a broker $\mathcal{B}$, and a set of IoT devices $\mathcal{I} = \{i_1, i_2, \ldots, i_{|I|}\}$. The broker is responsible for assigning the tasks generated by IoT devices to the proper fog nodes. The broker could be every single one of the fog nodes. To prevent a single point of failure, in the beginning, we make a list, and in that list, we specify the order of the next broker. In case of failure in the broker, the next fog node on the list will be the new broker, and by a message from the new broker, all of the devices on the network will become aware. Fog nodes are positioned at the edge of the network. Distance between IoT devices and the broker is often just one hop; therefore, it is possible to connect IoT devices to the broker over a high-speed broadband wireless connection [34].

Each fog node has two queues for tasks to be processed. Primary tasks are stored in the primary queue of fog nodes, and backup tasks are stored in the backup queue of fog nodes. Queued tasks are executed in accordance with the task's deadline. Tasks in the backup queue have a higher priority than

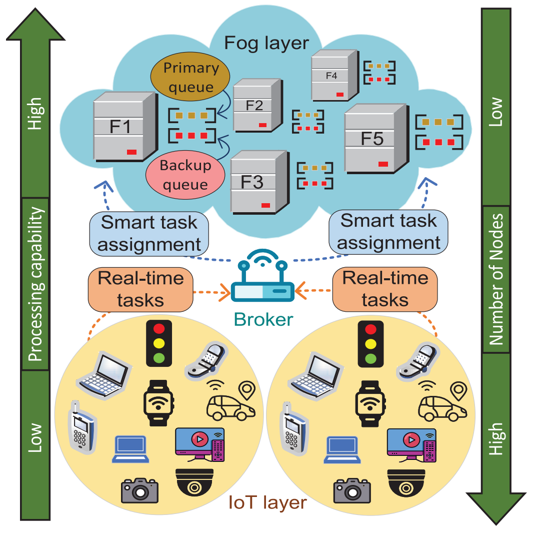


Fig. 3. Hierarchical model of the system in terms of processing capability and the number of nodes.

tasks in the primary queue, as the tasks in the backup queue close their deadlines. If any task exists in the backup queue, it will be executed before the tasks in the primary queue.

We employ a similar workload to that described in [19] and [35]. To be more precise, each IoT device $i_m \in \mathcal{I}$ has a set of tasks T_m $(\tau_1, \tau_2, \ldots, \tau_n)|n \geq 1$. A task is a piece of data transmitted from an IoT device to the broker. These tasks are processed on the fog layer. After processing the tasks, each fog node transmits the results to the broker, and the broker sends the results to the IoT device i_m. These tasks are distinct and inseparable. Each task $\tau_i \in T_j$ has a number of attributes: $\tau_i = \{s_i, a_i, l_i, d_i\}$, where s_i denotes the task's size (in bits), a_i denotes the arrival time of the tasks, l_i denotes the length of each task in CPU-cycle, and d_i is the deadline of each task. Additionally, each task has its own unique id, which is a combination of its sequence number and the id of the IoT device from which it originates.

B. Delay Model

In this study, four types of delays are considered as follows: 1) broker's processing time; 2) transmission delay; 3) execution time; and 4) queuing delay. These delays will be discussed in detail in the following.

1) Broker's Processing Time: T_b denotes the broker's processing time. In other words, this is the period of time during which the Q-learning model selects the primary and backup nodes for each task. The value of T_b is almost always the same, but it changes based on the specifications of the broker, so it is considered as an ε.

2) Transmission Delay: The term transmission delay refers to the sum of the time required to send the tasks to the fog nodes and the time required to deliver the results to the IoT nodes. In most cases, the result is a small packet. Therefore, the time required to deliver the results is neglected [36].

Based on [37], the data transmission rate between the broker $\mathcal{B}$ and the fog node f_j can be expressed as follows:

$$TR_{\mathcal{B},f_j}(t) = \omega \log_2\left(1 + \frac{h_{\mathcal{B},f_j}(t).p}{N}\right) \tag{3}$$

where ω is the transmission bandwidth, $h_{\mathcal{B},f_j}(t)$ is the channel power gain and it is equal to $\upsilon_1 \mathcal{Z}_{B,f_j}{}^{-\upsilon_2}$, where $\mathcal{Z}_{B,f_j}$ is the distance between the broker B and fog node f_j. Also, υ_1 is the path-loss constant and υ_2 is the path-loss exponent, p denotes the transmission power, and N is the noise. As a result, the transmission delay between the broker and a fog node f_j for task τ_i is denoted by

$$T^u_{f_j,\tau_i} = \frac{s_i}{TR_{\mathcal{B},f_j}(t)}. \tag{4}$$

3) Executing Time: The time for executing the task τ_i on a specific fog node f_j is expressed as

$$T^e_{f_j,\tau_i} = \frac{l_i}{frj} \tag{5}$$

where fr_j is the frequency of the fog node f_j.

4) Queuing Delay: $T^Q_{f_j,\tau_i}$ indicates how long a task must wait in the queue before being processed by a fog node. This delay is calculated as the sum of the processing times of tasks in the primary and backup queues that have a deadline earlier than task τ_i.

As a result, total delay D for task τ_i can be calculated as follows:

$$D_i = T_b + T^u_{f_j,\tau_i} + T^e_{f_j,\tau_i} + T^Q_{f_j,\tau_i}. \tag{6}$$

C. Reliability Model

Due to the diversity of environments and working scenarios for fog nodes, the failure rate for some tasks could be high. Accordingly, reliability is a critical issue that must be considered.

Our reliability model is based on [38]. The system's reliability is defined as the product of the probability that each fog node will be operational during the execution time of the assigned tasks and the probability that each communication link will be operational during the data transmission period. The Poisson process [38] affects the failures of fog nodes and communication links. Additionally, the failure rates of the fog node f_i are denoted by λ_i, while the failure rate of the communication lines between f_i and $\mathcal{B}$ is denoted by the letter μ_i. As a result, the computation reliability of each fog node f_i is

$$R^c_{f_i} = e^{-\lambda_i \frac{l_i}{f_i}}. \tag{7}$$

And also, the link reliability is

$$R^l_{f_i} = e^{-\mu_{i,j} \frac{s_i}{TR_{\mathcal{B},f_i}(t)}}. \tag{8}$$

Then, the reliability of a task τ_i that transmitted from $\mathcal{B}$ to f_i is denoted by

$$R^0_i = R^c_{f_i} * R^l_{f_i} = e^{-\lambda_i \frac{l_i}{f_j} - \mu_{i,j} \frac{s_i}{TR_{\mathcal{B},f_j}(t)}}. \tag{9}$$

As we use the primary–backup technique for each task, the reliability of each task consists of three disjoint events.

1) Both primary and backup nodes are functional.
2) The primary node is functional, but the backup node has failed.
3) The primary node has failed, and the broker does not receive the result, but the backup node is functional.

Because these three events are collectively exhaustive and mutually exclusive, according to Bayes' rule, it is possible to consider the reliability of each task as the sum of these three events [39].

As a result, the reliability of each task τ_i by utilizing the primary–backup technique is calculated as follows:

$$R_i = R_i^{0^2} + R_i^0\left[1 - R_i^0\right] + R_i^0\left[1 - R_i^0\right] \tag{10}$$

$$R_i = 2R_i^0 - R_i^{0^2}. \tag{11}$$

Therefore, the total reliability of the system is

$$R = \prod_{i=1}^{|T|} R_i. \tag{12}$$

D. Load Distribution Model

To measure and compare the workload distribution, the total workload in each fog node can be compared to the average workload of all fog nodes. The total workload for each fog node f_i is calculated based on

$$W_{f_i} = \mathrm{TPQ} + \mathrm{TBQ}. \tag{13}$$

In (13), TPQ and TBQ are the total amounts of CPU cycles required to process all of the tasks in the primary queue and backup queue of the fog node f_i

$$\mathrm{TPQ} = \sum_{p=1}^{P_i} l_p \tag{14}$$

$$\mathrm{TBQ} = \sum_{b=1}^{B_i} l_b \tag{15}$$

where P_i and B_i are the numbers of tasks in the primary and backup queue of the fog node f_i.

Therefore the workload distribution is calculated as follows:

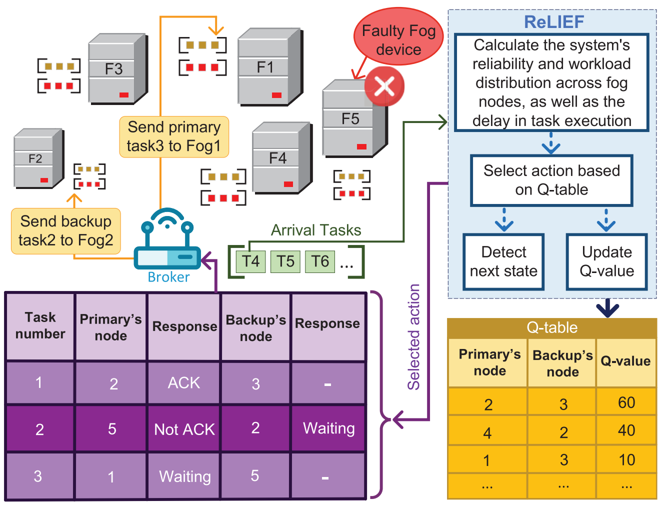
$$W_{\mathrm{AVG}} = \frac{\sum_{i=1}^{|F|} W_{f_i}}{|F|} \tag{16}$$

$$WL = \sum_{i=1}^{|F|} \left|W_{f_i} - W_{\mathrm{AVG}}\right| \tag{17}$$

where $|F|$ is the number of fog nodes, and WL is the workload distribution. The closeness of the WL to zero shows better workload balancing.

IV. Proposed Method

This section demonstrates the proposed task assignment technique based on RL, and the primary–backup approach [40], [41]. Fig. 4 provides an overview of our proposed method. To achieve a higher level of reliability, we utilize the primary–backup technique. The primary–backup technique is one of the most important techniques used in fault-tolerant systems. In this technique, there is a primary task and a backup task. If the primary task fails, then the backup task is executed. Due to the real-time nature of the tasks in our system, selecting nodes for the execution of the primary and backup tasks is an important issue. For this selection, we employ the Q-learning


ReLIEF
Calculate the system's reliability and workload distribution across fog nodes, as well as the delay in task execution
Select action based on Q-table
Detect next state
Update Q-value
Faulty Fog device
F1
F2
F3
F4
F5
Send primary task3 to Fog1
Send backup task2 to Fog2
Broker
Arrival Tasks
T4
T5
T6
...
Selected action
Q-table

Task number	Primary's node	Response	Backup's node	Response
1	2	ACK	3	-
2	5	Not ACK	2	Waiting
3	1	Waiting	5	-

Primary's node	Backup's node	Q-value
2	3	60
4	2	40
1	3	10
...	...	...

Fig. 4. Overview of our proposed method (ReLIEF).

method. As previously stated, Q-learning requires three critical parameters. These parameters and problem formulation will be discussed in greater detail in this section.

A. Problem Formulation

In this section, we formulate the problem by utilizing integer linear programming (ILP). The following notations are used in this formulation. n is the number of tasks that are ready to be executed. y is the number of available fog pairs, and t is the number of time slots in each hyper period.

1) The matrix $RR \in \mathbb{R}^{n\times y\times t}$ represents the reliability of the system. In this matrix, each element RR_{ikl} represents the reliability of the task τ_i or τb_i (τ_i is the original task and τb is the backup) in time slot l, which is executed on fog device k.
2) The task-to-fog-pair mapping is represented by the matrix $X \in \{0, 1\}^{n\times y\times t}$. At the time slot l, the task τ_i or τb_i is mapped to the fog device k if and only if $X_{ikl} = 1$.

The final objective of our method is to improve the total reliability of the system and keep it higher than a specific level while meeting the task's real-time constraints.

Objective: The total reliability of the system (i.e., the product of the reliability of all running tasks in each time slot l) should be more than a specific reliability level R_l

$$\forall i : \prod_{i=1}^{n} X_{ikl} RR_{ikl} \geq R_l. \tag{18}$$

Consequently, this is a 0-1 assignment problem, and we have the following:

$$\forall i, k, l : X_{ikl} \in \{0, 1\}. \tag{19}$$

Constraint: All tasks in the task set must complete all their execution phases before the deadline

$$\forall i, k, l : X_{ikl} \leq d_i. \tag{20}$$

B. Q-Learning Parameters

The following sections define the state space, action space, and reward function. It should be noted that the Q-values are initially determined randomly. In consequence, the model's outputs become less reliable and more unpredictable with time. This issue is referred to as the "cold-start issue" [42]. In order to address the cold-start issue, the model was run in a simulated environment prior to being used in runtime to verify that the Q-values are more realistic than the random version.

1) State-Space: Several system metrics, such as node temperature, power consumption, bandwidth, and workload distribution, can be utilized to determine the current status of the system. However, using all of these parameters at the same time is not necessary, and also, using all of them may result in increasing the model complexity. Therefore, only a subset of these metrics must be chosen. The reliability and workload distribution of the system are used to determine the state of the system. The reason for choosing these two metrics is that our primary objective is to improve reliability while maximizing the workload balancing of the system. Therefore, these two metrics were selected among the others. The value of state variables, such as reliability and workload distribution, is continuous and can take any value; representing the state using continuous values may lead the system to significant computational complexity and reduce convergence speed. To alleviate this problem, all state values are treated as a number of discrete values [43]. Therefore, the actual state values are mapped to discrete values depending on their proximity to the discrete value. As a result, the system may have one of k discrete and specific states represented by $s_1, s_2, \dots, s_k$. Each state is a representation of the fog node's reliability and workload balancing. States are denoted by $s_i = \{R_i, WL_i\}$, where R_i is the reliability of fog node i and WL_i is the workload balancing.

2) Action-Space: There are only a limited number of distinct actions available for the model to search in. Actions are denoted by $A = \{a_1, a_2, \dots, a_{|A|}\}$, where each a_i represents the id of two fog nodes. One of them is the id of a fog node that is responsible for the execution of the primary task, and the second one is the id of a fog node that is responsible for the execution of the backup task.

3) Reward-Function: As discussed earlier, the reward function directs the learning process by giving points to the RL agent. The reward function must be defined in terms of the state of the RL agent as well as the action executed by the RL agent. The reward function must be constructed in conjunction with the action, state, and other aspects that must be minimized or balanced. As a result, the reward function for the model that transients between states s_i and s_j is as follows:

$$r_{k+1}|\left(s_i, a_m, s_j\right) = \rho_1(\Delta WL/WL_k) + \rho_2\left(\Delta D/D_{i_k}\right) + \rho_3(\Delta R/R_K). \quad (21)$$

Equation (21) expresses the amount of reward that the model will receive for doing action a_m from state s_i. Combining multiple metrics is not straightforward due to their distinct behaviors and cardinality. To address these issues, Swaminathan et al. [44] have proposed the use of principal component analysis. This is effective but is limited by a number of factors, such as the nonlinear or orthogonal relationship between workload distribution, delay, and reliability, as well as the complexity required to operate in the utilized scenario.

Algorithm 1 Task Scheduler in Fog Nodes

```
Input: task $\tau_i$.
Output: The result of the task $\tau_i$.
 1: primary_execution_queue = [ ]
 2: backup_execution_queue = [ ]
 3: for each input task do
 4:     if $\tau_i$ is primary then
 5:         primary_execution_queue.append ($\tau_i$)
 6:     else
 7:         backup_execution_queue.append ($\tau_i$)
 8:     end if
 9:     if backup_execution_queue is not empty then
10:         for each task $\tau_i$ in backup_execution_queue do
11:             EDF (backup_execution_queue)
12:         end for
13:     else
14:         EDF (primary_execution_queue)
15:     end if
16: end for
```

Consequently, we consider the variation in the WL relative to the target WL, the variation in the D related to the target D, and the variation in the R related to the target R. Therefore, $\Delta WL/WL_k$, $\Delta D/D_{i_k}$, and $\Delta R/R_k$ represent the change in the workload distribution, delay, and reliability with respect to the target value of these variables, respectively. The parameters ρ_1, ρ_2, and ρ_3 correspond to the learning parameters discovered through a vast number of experiments [45]. In our case, ρ_1 equals 0.36, ρ_2 equals 0.27, and ρ_3 equals 0.29.

C. Design Methodology

Our proposed method has two distinct parts. One part executes on fog nodes based on Algorithm 1, and the other one executes on the broker as discussed in Algorithm 2. These two algorithms will be discussed in more detail in Sections IV-C1 and IV-C2.

1) Broker: The pseudocode in Algorithm 2 illustrates the decision-making process in the broker for finding the appropriate nodes for primary and backup tasks. In the first stage of Algorithm 2, the broker calculates reliability and workload distribution as in (9) and (17), respectively (lines 1 and 2). After that, the state of the system must be determined based on the calculated reliability and workload distribution (line 5). After determining the state of the system, the best policy must be chosen based on Bellman's principle of optimality (2) (line 6). Then, the primary task must be sent to the selected node based on the chosen action (line 7). Following that, the broker waits for the results of the task τ_i. If the broker receives the result successfully, then the task must be deleted from the list Ψ (lines 8–10). Otherwise, if the result is not received prior to a certain time, a backup is sent to the specified node (lines 11–15). That specific moment can be estimated using the formula in line 11. Then, the reward is calculated by the reward function as in (21) (line 18). Finally, new Q-values are determined using (1) (line 19), and the Q-table will be updated based on new Q-values.

Algorithm 2 Q-Learning-Based Reliability Management in Broker

```
Input: Task set $\mathcal{T} = \{\tau_1, \tau_2, ..., \tau_{|T|}\}$, a set of Fog nodes $\mathcal{F} = \{f_1, f_2, ..., f_{|F|}\}$
Output: Determining the primary and backup nodes.
1: while $\Psi$ is not empty do
2:   for $\tau_i$ in $\Psi$ do
3:     Calculate reliability as in Eq. (12).
4:     Calculate the workload distribution based on Eq. (17).
5:     Determine the system's state, which corresponds to the task's reliability and the computed workload distribution.
6:     Choose an action that has the best policy in terms of Eq. (2).
7:     Send primary task to selected node based on chosen action.
8:     while $task\_remaining\_deadline > transmission\_time + processing\_time$ do
9:       if The result of the task $\tau_i$ received then
10:        Remove the task from $\Psi$;
11:        break;
12:      end if
13:    end while
14:    if The result of the task $\tau_i$ NOT received then
15:      Send the backup version of the task to the chosen node;
16:    end if
17:    Calculate the reward using Eq. (21).
18:    As in Eq. (1), set the new values of Q-values.
19:  end for
20: end while
```

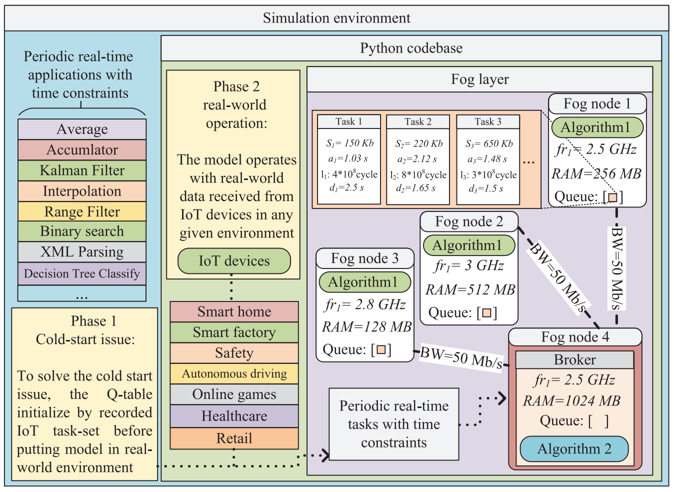


Fig. 5. Overview of our simulation environments.

2) Fog Nodes: Algorithm 1 specifies the order of execution of tasks in each fog node. In fog nodes, there are two execution queues: 1) primary and 2) backup (lines 1 and 2). Each input task τ_i appends to one of these queues. If the input task is primary, it appends to the primary queue, and if the input task is backup, it appends to the backup queue (lines 4–8). Execution priority is always with the backup queue because these tasks have no other backup, and they are so close to the deadline. As a result, if there are no tasks in the backup queue, the execution of tasks in the primary queue begins based on EDF (lines 10–15). The tasks are executed based on the partitioned earliest-deadline-first (EDF) scheduling policy. The EDF policy is a well-known dynamic priority scheduling algorithm used in real-time systems to schedule tasks in a priority queue. Whenever a scheduling event occurs (a task finishes, a new task is released, etc.), the priority queue will be searched for a task with the earliest deadline. After a task is done, the result is sent back to the broker, which sends it to the IoT devices that generated it.

V. Evaluation

In this section, we discuss the simulation environment and analyze the effectiveness of the proposed method in terms of 1) computational delay; 2) transmission delay; 3) queuing delay; 4) reliability of the system in a variety of scenarios; and 5) utilization using the fog computing framework. Fig. 5 illustrates the evaluation setup. Moreover, we compared our proposed method with the following baseline strategy.

1) *Random Action:* In the random action strategy, the primary fog node and the backup fog node for each task are chosen randomly.
2) *Without Backup:* In this strategy, each task has only one chance to be processed. More precisely, each task is only assigned to one fog node, and there is no backup fog node.
3) *Fast Offloading:* In the fast offloading strategy, each task is assigned to the fog node with the shortest delay, and the backup task is assigned to the next fog node with the shortest delay.

Additionally, to demonstrate the proposed strategy's efficacy, we compare it to three states of art methods, i.e., DDTP [19], MOO [27], and DPTO [17].

A. Simulation Environment

For simulation, we considered 30 IoT nodes that generate multiple periodic real-time tasks. The task set for this work contains real-time tasks, which means that all of them have time constraints that must be considered. These tasks need more computational power to be executed on IoT devices. Therefore, all of these tasks need to be offloaded to fog devices. The number of fog nodes can be dynamically changed, and the proposed method can adapt to the different numbers of fog nodes. The entire simulation is run on an Intel Core i7-6700HQ processor @ 3.5 GHz $\times$ 8 with 16-GB RAM using the Ubuntu 20.04 LTS operating system. The simulation is fully implemented with Python programming language. All of the experimental processes are repeated 2500 times, and the average result is reported. Table III demonstrates the rest of the simulation parameters in detail [17].

B. Experimental Results

We validate our reliability approach by comparing the proposed (ReLIEF) with the mentioned state-of-the-art methods and baseline strategies.

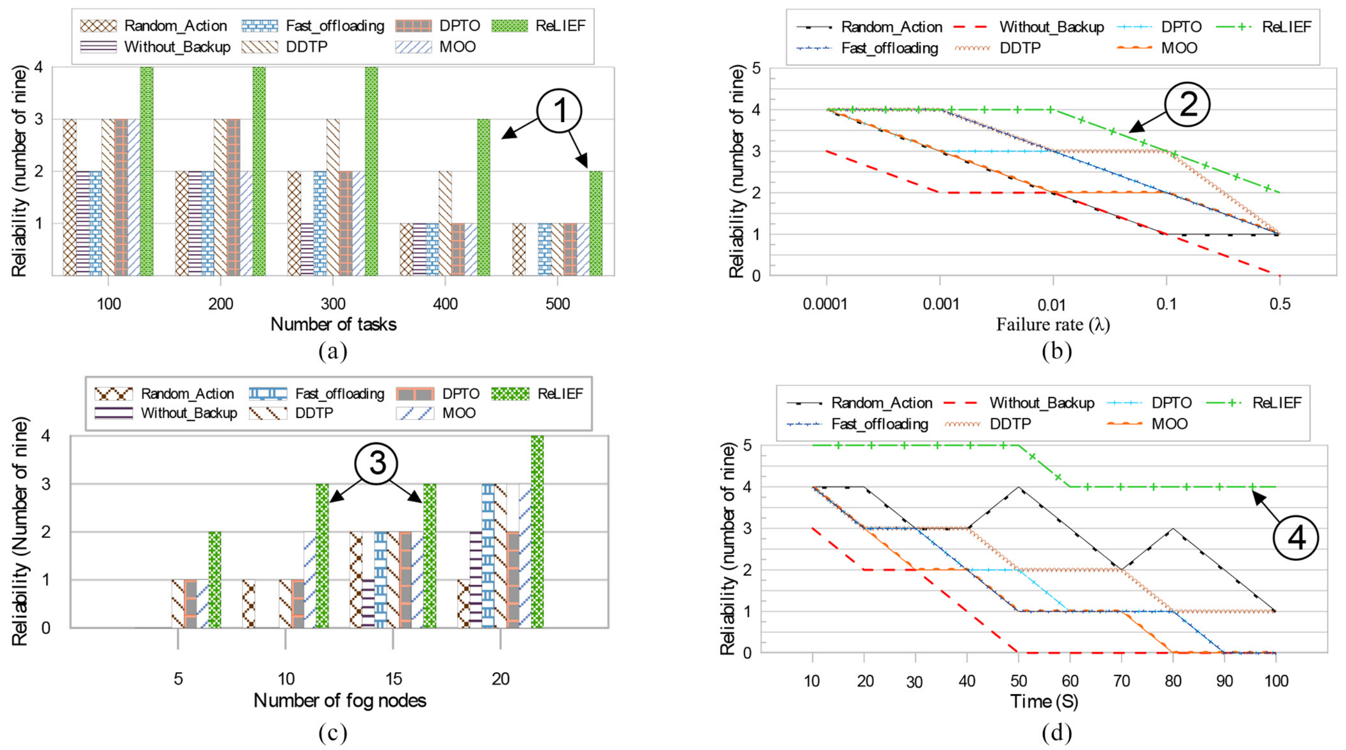


Fig. 6. Comparison between proposed (ReLIEF) and state-of-the-art methods and also baseline strategy. The proposed method (ReLIEF) was compared to state-of-the-art techniques and a baseline strategy based on 2500 experiments across various tasks with varying computational needs. (a) Average reliability for different number of tasks. (b) Average reliability for different fault rates. (c) Reliability with different numbers of fog nodes. (d) Average reliability during 100 s with $\lambda = 0.0001$.

TABLE III
SIMULATION'S PARAMETERS

PARAMETER	Value
Number of IoT nodes	30
Number of fog nodes	$[5 - 15]$
Number of tasks	$[50 - 300]$
Task size	$[100 - 1000]$ Kb
Length of tasks	$[1 - 15] \times 10^8$ cycle
Deadline of tasks	$[100 - 5000]$ ms
$freq_{f_i}$	$[1 - 5] * \times 10^9$ cycle/s
ω	50 Mb/s
Path loss parameters (υ_1, υ_2)	$(10^{-3}, 4)$
p	10^{-3} W
N	10^{-10} W

1) Reliability: Fig. 6(a) demonstrates the average reliability for a different number of tasks based on (12). Reliability is calculated based on the number 9. In this study, *reliability* is defined as a combination of functionality and timing. In other words, due to the real-time nature of the tasks, processing tasks before the deadline is as important as processing tasks correctly. As illustrated in Fig. 6(a), our proposed (ReLIEF) has a higher level of reliability (key observation ①). It may select suitable fog nodes based on the deadline using RL, and the backup ensures that functionality is right. According to our experiments, on average, we have a 62.4% improvement in reliability with a different number of tasks. Another experiment, represented in Fig. 6(b), demonstrated the average reliability for various fault rates. It is obvious that by increasing the fault rate, the reliability in all methods decreases, and the proposed (ReLIEF) method outperforms other state-of-the-art methods (key observation ②) by an average of 71.5% due to the backup mechanism and intelligent selection of fog nodes. As seen in Fig. 6(c), increasing the number of fog nodes increases the reliability. This happens because by increasing the number of fog devices, there are more choices to select. With a good workload-balancing strategy, increasing choices could lead to less crowded queues in fog devices. As a result, a higher number of tasks can reach their deadlines (key observation ③). "ReLIEF" also achieved, on average 84.6% reliability on a different number of fog nodes. Also, Fig. 6(d) indicates that during a specific amount of time, by increasing the number of tasks that comes to the fog nodes from the IoT nodes reliability of real-time tasks can be compromised. In this experiment, we monitored the reliability level of 2500 real-time tasks for 100 s. It shows that smart node selection leads to good workload balancing, and a better level of reliability can be achieved (key observation ④).

The reason that our proposed method (ReLIEF) outperformed other methods in a larger number of fog nodes is that by using RL, ReLIEF can dynamically adapt to the new situation, and by better workload balancing, it reduces the crowd in fog node's queue. Therefore, it reduces the delay.

2) Workload Distribution: One of the other important parameters that affect the performance of a fog system is workload distribution. In this case, (17) reflects the maximum amount of variation in workload across all of the fog nodes. As this value decreases and becomes closer to zero, it indicates that the workload is balanced between fog nodes. As a result, a lower number represents better workload balancing. As seen in

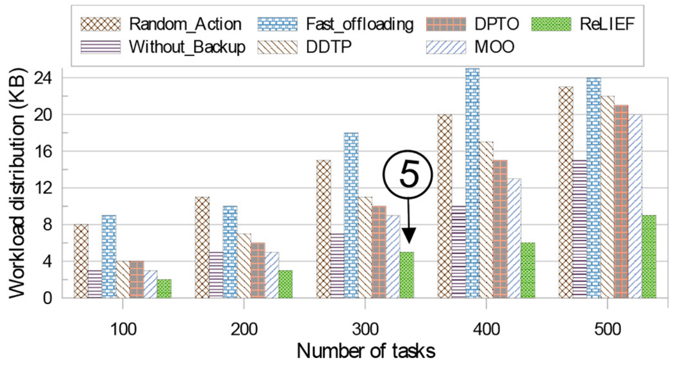


Fig. 7. Workload distribution for different number of tasks (lower is better).

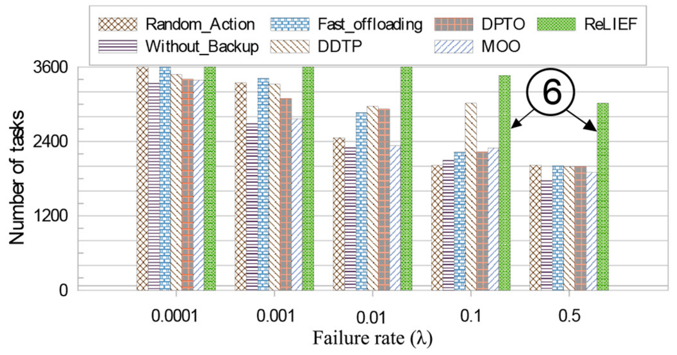


Fig. 8. Number of successfully executed tasks before the deadline of each task out of 3600 tasks with different failure rates.

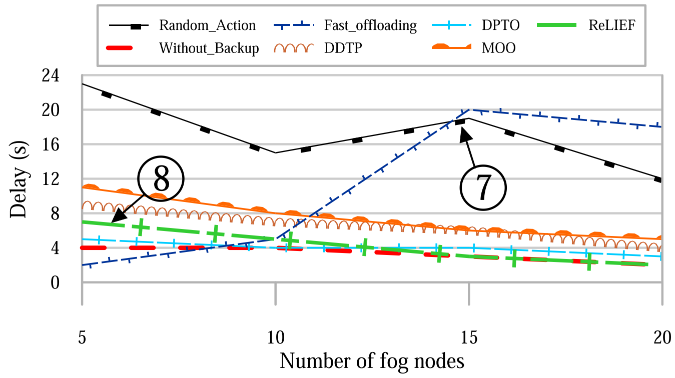


Fig. 9. Delay for a different number of fog nodes.

TABLE IV
IMPACT OF THE HYPERPARAMETERS ON THE PERFORMANCE METRICS

ρ_1	ρ_2	ρ_3	WL	Delay	Reliability
0.34	0.34	0.68	16.88	9.25	5
0.34	0.58	0.33	17.62	8.95	5
0.53	0.42	0.37	4.83	6.12	6
0.33	0.33	0.33	6.73	5.69	7
0.36	**0.27**	**0.29**	**5.56**	**5.3**	**7**

Fig. 7, ReLIEF (key observation ⑤) is 83.3% better in workload balancing compared to the state-of-the-art works because the workload is one of the main parameters in the reward function (21) the model always moves in a direction that has better load balancing. Since load balancing causes workload distribution among fog nodes and uses the resource's all of the fog nodes, energy consumption may increase rather than other methods. However, workload balancing reduces peak power and keeps the temperature of fog nodes from rising.

3) Throughput: The throughput of the system is determined by the number of tasks completed within a specified time period. Fig. 8 illustrates how many of the 3600 tasks were completed successfully prior to the deadline. ReLIEF performs better than other baseline strategies and state-of-the-art methods (key observation ⑥). One of the main reasons for this is using the primary–backup approach. Using this technique, if one of the fog nodes or communication link fails and the result is not received, the backup task in other fog nodes begins to execute in order to deliver the result before its deadline. Another reason is intelligent node selection. The Q-learning approach is used in this article to select appropriate primary and backup nodes depending on the task's deadline, delay, and workload on fog nodes. In this work, balancing the workload among fog nodes is important, and each task assigns to fog nodes based on its workload. Balancing workload prevents congestion on fog nodes and causes tasks to meet their deadlines. All of the above-mentioned factors lead to an average 84% improvement in successfully completed tasks.

4) Delay: The time complexity of this approach can be studied in two phases. Phase one is the phase in which the model tries to converge and generates an effective policy table with higher time complexity than some of the other states of the arts. Phase two is the phase after convergence, and the agent can decide only by looking and searching in the policy table. The complexity of this phase is O(n) which is slightly lower than the state of the arts. The complexity in phase one could not cause any trouble for the system because the tasks are real time, and in real-time tasks, it is only important to finish them before the deadline and not as soon as possible. Our experiments show that even in phase one, with higher time complexity than the state-of-the-art methods, ReLIEF could execute tasks before their deadlines, and it is because of the fare and balance allocation of the tasks. This balance and fare allocation help the system prevent crowded queues in fog nodes.

As illustrated in Fig. 9, increasing the number of fog nodes while maintaining a fixed number of tasks reduces the time required to process each task. This is because by increasing the number of fog nodes, there are more resources available. By assigning tasks to fog nodes based on workload distribution, processing queues become less congested. Additionally, as one of the most critical components in the reward function is the delay, it has an important role in the selection of the fog nodes. As seen in Fig. 9, in the fast offloading approach, the delay increases with an increasing number of devices (key observation ⑦). This is because this approach just focuses on transmission delay and does not consider queuing delay, whereas ReLIFE (key observation ⑧) considers transmission, queuing, and processing delays. Additionally, the delay varies intermittently in the random action approach and lacks any discernible pattern, as fog nodes responsible for task execution chose arbitrarily without a specified strategy. Nonetheless, when an error occurs in a primary task, the task's response time increases. This is because the execution of the backup task was not initiated until the deadline approached. Sending backup tasks close to the deadline conserves energy and prevents the waste of resources, particularly in environments with a low error probability.

Moreover, selecting different values for learning parameters has an impact on the final results. Therefore, it is important to select these values carefully. Table IV is a part of the experiments to find the best values for learning parameters. Selected values are in the last row of the table.

VI. Conclusion

This article has proposed an RL-based real-time task assignment in fault-tolerant fog networks, called ReLIEF, to increase reliability while meeting real-time constraints. To increase the reliability, we exploited the primary–backup technique. If a failure happens in one of the fog nodes or communication links, this technique ensures that the backup task executes on another fog node to ensure meeting the deadline. Due to the fact that selecting each fog node can have a specific effect on the system's status, it is critical to select suitable fog nodes for executing the primary and backup tasks. Primary and backup nodes are selected by the Q-learning algorithm based on the deadline of the task, delay, and the workload of fog nodes. With a good workload-balancing strategy, the model can lead the system to a status with less crowded queues in fog devices. As a result, the delay can be reduced, and a higher number of tasks can meet their deadlines. Also, the simulation results indicate the proposed ReLIEF strategy's effectiveness in comparison to many state-of-the-art algorithms with a variety of performance metrics.

References

[1] B. Safaei, A. M. H. Monazzah, and A. Ejlali, "ELITE: An elaborated cross-layer RPL objective function to achieve energy efficiency in Internet-of-Things devices," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1169–1182, Jan. 2021.

[2] W. Duo, M. Zhou, and A. Abusorrah, "A survey of cyber attacks on cyber physical systems: Recent advances and challenges," *IEEE/CAA J. Automatica Sinica*, vol. 9, no. 5, pp. 784–800, May 2022.

[3] P. Marwedel, *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*, 2nd ed. Cham, Switzerland: Springer Publ. Company, Incorp., 2010.

[4] P. Zhang et al., "A fault-tolerant model for performance optimization of a fog computing system," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 1725–1736, Feb. 2022.

[5] G. Li, J. Yan, L. Chen, J. Wu, Q. Lin, and Y. Zhang, "Energy consumption optimization with a delay threshold in cloud-fog cooperation computing," *IEEE Access*, vol. 7, pp. 159688–159697, 2019.

[6] Y. Li, B. Yang, H. Wu, Q. Han, C. Chen, and X. Guan, "Joint offloading decision and resource allocation for vehicular fog-edge computing networks: A contract-Stackelberg approach," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 15969–15982, Sep. 2022.

[7] M. Abdel-Basset, R. Mohamed, M. Elhoseny, A. K. Bashir, A. Jolfaei, and N. Kumar, "Energy-aware marine predators algorithm for task scheduling in IoT-based fog computing applications," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 5068–5076, Jul. 2021.

[8] A. Hazra, P. K. Donta, T. Amgoth, and S. Dustdar, "Cooperative transmission scheduling and computation offloading with collaboration of fog and cloud for industrial IoT applications," *IEEE Internet Things J.*, early access, Feb. 9, 2022, doi: 10.1109/JIOT.2022.3150070.

[9] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, 1st ed. Boston, MA, USA: Kluwer Acad. Publ., 1997.

[10] M. Abdel-Basset, D. El-Shahat, M. Elhoseny, and H. Song, "Energy-aware metaheuristic algorithm for industrial-Internet-of-Things task scheduling problems in fog computing applications," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12638–12649, Aug. 2021.

[11] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 3, pp. 361–373, Sep. 2017.

[12] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 10782–10793, Nov. 2020.

[13] P. Zhang, M. Zhou, and G. Fortino, "Security and trust issues in fog computing: A survey," *Future Gener. Comput. Syst.*, vol. 88, pp. 16–27, Nov. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17329722

[14] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[15] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo, and J. Wang, "DEBTS: Delay energy balanced task scheduling in homogeneous fog networks," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2094–2106, Jun. 2018.

[16] B. Hu, Z. Cao, and M. Zhou, "Scheduling real-time parallel applications in cloud to minimize energy consumption," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 662–674, Jan.–Mar. 2022.

[17] M. Adhikari, M. Mukherjee, and S. N. Srirama, "DPTO: A deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5773–5782, Jul. 2020.

[18] C. Swain et al., "METO: Matching-theory-based efficient task offloading in IoT-fog interconnection networks," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12705–12715, Aug. 2021.

[19] I. Sarkar, M. Adhikari, N. Kumar, and S. Kumar, "Dynamic task placement for deadline-aware IoT applications in federated fog networks," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1469–1478, Jan. 2022.

[20] S. Ghanavati, J. Abawajy, and D. Izadi, "Automata-based dynamic fault tolerant task scheduling approach in fog computing," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 1, pp. 488–499, Jan.–Mar. 2022.

[21] X. Hou, Z. Ren, J. Wang, S. Zheng, W. Cheng, and H. Zhang, "Distributed fog computing for latency and reliability guaranteed swarm of drones," *IEEE Access*, vol. 8, pp. 7117–7130, 2020.

[22] Y. Sun, S. Zhou, and Z. Niu, "Distributed task replication for vehicular edge computing: Performance analysis and learning-based algorithm," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1138–1151, Feb. 2021.

[23] J. Wang, K. Liu, B. Li, T. Liu, R. Li, and Z. Han, "Delay-sensitive multi-period computation offloading with reliability guarantees in fog networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 9, pp. 2062–2075, Sep. 2020.

[24] B. Li, G. Wu, Y. He, M. Fan, and W. Pedrycz, "An overview and experimental study of learning-based optimization algorithms for vehicle routing problem," 2021, *arXiv:2107.07076*.

[25] Z. Cao, C. Lin, M. Zhou, and R. Huang, "Scheduling semiconductor testing facility by using cuckoo search algorithm with reinforcement learning and surrogate modeling," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 825–837, Apr. 2019.

[26] Y. Yang, Z. Ding, R. Wang, H. Modares, and D. C. Wunsch, "Data-driven human-robot interaction without velocity measurement using off-policy reinforcement learning," *IEEE/CAA J. Automatica Sinica*, vol. 9, no. 1, pp. 47–63, Jan. 2022.

[27] M. Adhikari, S. N. Srirama, and T. Amgoth, "Application offloading strategy for hierarchical fog environment through swarm optimization," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4317–4328, May 2020.

[28] J. Yao and N. Ansari, "Fog resource provisioning in reliability-aware IoT networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8262–8269, Oct. 2019.

[29] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[30] Q.-H. Zhu, H. Tang, J.-J. Huang, and Y. Hou, "Task scheduling for multi-cloud computing subject to security and reliability constraints," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 4, pp. 848–865, Apr. 2021.

[31] R. Sutton and A. Barto, *Reinforcement Learning, Second Edition: An Introduction* (Adaptive Computation and Machine Learning Series). Cambridge, MA, USA: MIT Press, 2018.

[32] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[33] R. Bellman, *Dynamic Programming* (Princeton Landmarks in Mathematics and Physics). Princeton, NJ, USA: Princeton Univ. Press, 2010.

[34] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[35] P. Cai, F. Yang, J. Wang, X. Wu, Y. Yang, and X. Luo, "JOTE: Joint offloading of tasks and energy in fog-enabled IoT networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3067–3082, Apr. 2020.

[36] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M.-T. Zhou, "FEMTO: Fair and energy-minimized task offloading for fog-enabled IoT networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4388–4400, Jun. 2019.

[37] J.-Y. Baek, G. Kaddoum, S. Garg, K. Kaur, and V. Gravel, "Managing fog networks using reinforcement learning based load balancing algorithm," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2019, pp. 1–7.

[38] S. M. Shatz and J.-P. Wang, "Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems," *IEEE Trans. Rel.*, vol. 38, no. 1, pp. 16–27, Apr. 1989.

[39] K. S. Trivedi, *Probability and Statistics With Reliability, Queuing, and Computer Science Applications*, 2nd ed. New York, NY, USA: Wiley, 2001.

[40] H. Zou and F. Jahanian, "A real-time primary-backup replication service," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 6, pp. 533–548, Jun. 1999.

[41] S. Safari et al., "A survey of fault-tolerance techniques for embedded systems from the perspective of power, energy, and thermal issues," *IEEE Access*, vol. 10, pp. 12229–12251, 2022.

[42] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2002, pp. 253–260. [Online]. Available: https://doi.org/10.1145/564376.564421

[43] S. M. P. Dinakarrao, A. Joseph, A. Haridass, M. Shafique, J. Henkel, and H. Homayoun, "Application and thermal-reliability-aware reinforcement learning based multi-core power management," *J. Emerg. Technol. Comput. Syst.*, vol. 15, no. 4, pp. 1–19, Oct. 2019.

[44] K. Swaminathan, N. Chandramoorthy, C.-Y. Cher, R. Bertran, A. Buyuktosunoglu, and P. Bose, "BRAVO: Balanced reliability-aware voltage optimization," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2017, pp. 97–108.

[45] N. Habib, *Hands-On Q-Learning With Python: Practical Q-Learning With OpenAI Gym, Keras, and TensorFlow*. Birmingham, U.K.: Packt Publ., 2019.

Roozbeh Siyadatzadeh received the B.Sc. degree in computer engineering from Persian Gulf University, Bushehr, Iran, in 2020.

He is admitted as an Exceptional Talented Student with Sharif University of Technology, Tehran, Iran, for M.Sc. programs, in 2020, where he is currently a member of the Embedded Systems Research Laboratory, Department of Computer Engineering. His research interests include embedded systems and cyber–physical systems, low-power design, and machine learning.

Fatemeh Mehrafrooz received the B.S. degree in computer engineering from Amirkabir University of Technology, Tehran, Iran, in 2020. She is currently pursuing the master's degree in computer engineering with Sharif University of Technology, Tehran.

She is a member of the Embedded Systems Research Laboratory, Department of Computer Engineering, Sharif University of Technology. Her research interests include Internet of Things, fog computing, and machine learning.

Mohsen Ansari received the Ph.D. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2021.

He is currently an Assistant Professor of Computer Engineering with Sharif University of Technology. He was a Visiting Researcher with the Chair for Embedded Systems, Karlsruhe Institute of Technology, Karlsruhe, Germany, from 2019 to 2021. His research interests include low-power design, real-time embedded systems, cyber–physical systems, and hybrid systems design.

Dr. Ansari was the Technical Program Committee (TPC) Member of the Asia and South Pacific Design Automation Conference in 2022.

Bardia Safaei received the Ph.D. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2021.

He was a member of the Dependable Systems Laboratory from 2014 to 2017 and Embedded Systems Research Laboratory from 2017 to 2021. As a Ph.D. Visiting Researcher, he was with the Chair for Embedded Systems, Karlsruhe Institute of Technology, Karlsruhe, Germany, from 2019 to 2020. He is currently a Faculty Member of the Department of Computer Engineering, Sharif University of Technology, where he is the Director of the Reliable and Durable IoT Applications and Networks Laboratory. His research interests include power efficiency and dependability challenges in Internet of Things, wireless sensor networks, mobile ad-hoc networks, and fog computing.

Dr. Safaei received the ACM/SIGAPP Student Award in the 34th ACM/SIGAPP Symposium on Applied Computing (SAC'19). He is honored to be selected as a member of the National Elites Foundation from 2016 to 2020. He has served as a reviewer in several prestigious international journals and conferences, such as IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, *ACM Transactions on Storage*, ACM/IEEE DAC, IEEE Sensors Conference, and IEEE WF-IoT.

Muhammad Shafique (Senior Member, IEEE) received the Ph.D. degree in computer science from Karlsruhe Institute of Technology, Karlsruhe, Germany, in 2011.

He was a Full Professor with the Institute of Computer Engineering, TU Wien, Vienna, Austria, from October 2016 to August 2020. Since September 2020, he has been with the Division of Engineering, New York University Abu Dhabi, Abu Dhabi, UAE, and is a Global Network Faculty Member of the NYU Tandon School of Engineering, Brooklyn, NY, USA. His research interests are in system-level design for brain-inspired computing, AI/machine learning hardware, wearables, autonomous systems, energy-efficient and robust computing, IoT, and smart CPS.

Dr. Shafique received the 2015 ACM/SIGDA Outstanding New Faculty Award, the AI 2000 Chip Technology Most Influential Scholar Award in 2020 and 2022, six gold medals, and several best paper awards and nominations. He has given several keynotes, talks, and tutorials and organized special sessions at premier venues. He has served as the PC chair, general chair, track chair, and PC member for several conferences.

Jörg Henkel (Fellow, IEEE) received the Diploma and Ph.D. degrees from the Technical University of Braunschweig, Braunschweig, Germany, in 1996.

He is the Chair Professor of Embedded Systems with Karlsruhe Institute of Technology, Karlsruhe, Germany. Before that, he was a Research Staff Member of NEC Laboratories, Princeton, NJ, USA. His research work is focused on codesign for embedded hardware/software systems with respect to power, thermal, and reliability aspects.

Prof. Henkel has received six Best Paper Awards from, among others, ICCAD, ESWeek, and DATE. He is currently the Editor-in-Chief of the *IEEE Design & Test Magazine*. He served as the General Chair of ICCAD and ESWeek. He coordinates the DFG Program SPP 1500 "Dependable Embedded Systems" and is a Site Coordinator of the DFG TR89 Collaborative Research Center on "Invasive Computing." For two consecutive terms, he served as the Editor-in-Chief for the *ACM Transactions on Embedded Computing Systems*.

Alireza Ejlali received the Ph.D. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2006.

He is currently an Associate Professor of Computer Engineering with Sharif University of Technology. From 2005 to 2006, he was a Visiting Researcher with the Electronic Systems Design Group, University of Southampton, Southampton, U.K. In 2006, he joined Sharif University of Technology as a Faculty Member of the Department of Computer Engineering, where he was the Director of Computer Architecture Group from 2011 to 2015. His research interests include low-power design, real-time embedded systems, and fault-tolerant embedded systems.