future internet

MDPI

Article

Minimum-Cost-Based Neighbour Node Discovery Scheme for Fault Tolerance under IoT-Fog Networks

Premalatha Baskar and Prakasam Periasamy *

School of Electronics Engineering, Vellore Institute of Technology, Vellore 632014, Tamilnadu, India; premalatha.b@vit.ac.in

* Correspondence: prakasamp@gmail.com

Abstract: The exponential growth in data traffic in the real world has drawn attention to the emerging computing technique called Fog Computing (FC) for offloading tasks in fault-free environments. This is a promising computing standard that offers higher computing benefits with a reduced cost, higher flexibility, and increased availability. With the increased number of tasks, the occurrence of faults increases and affects the offloading of tasks. A suitable mechanism is essential to rectify the faults that occur in the Fog network. In this research, the fault-tolerance (FT) mechanism is proposed based on cost optimization and fault minimization. Initially, the faulty nodes are identified based on the remaining residual energy with the proposed Priority Task-based Fault-Tolerance (PTFT) mechanism. The Minimum-Cost Neighbour Candidate Node Discovery (MCNCND) algorithm is proposed to discover the neighbouring candidate Fog access node that can replace the faulty Fog node. The Replication and Pre-emptive Forwarding (RPF) algorithm is proposed to forward the task information to the new candidate Fog access node for reliable transmission. These proposed mechanisms are simulated, analysed, and compared with existing FT methods. It is observed that the proposed FT mechanism improves the utilization of an active number of Fog access nodes. It also saved a residual energy of 1.55 J without replicas, compared to the 0.85 J of energy that is used without the FT method.

Keywords: fault tolerance; fog computing; Internet of Things; neighbour node discovery; pre-emptive forwarding

Citation: Baskar, P.; Periasamy, P. Minimum-Cost-Based Neighbour Node Discovery Scheme for Fault Tolerance under IoT-Fog Networks. *Future Internet* **2024**, *16*, 123. https://doi.org/10.3390/fi16040123

Received: 13 March 2024
Revised: 29 March 2024
Accepted: 2 April 2024
Published: 3 April 2024

1. Introduction

The exponential growth in network traffic has led to various emerging wireless technologies such as 5G and Beyond-5G. It is anticipated that these new wireless technologies will grow in importance across multiple industries and make more strides in portable device applications. Smart home systems, productivity, animal farms, agriculture, environmental monitoring, e-health services, commercial and industrial, motor vehicles and transportation, space exploration, and wearable devices are only a few of the numerous sectors in which IoT has generated an unending number of new applications in smart cities. Everything is now easier than ever with the help of IoT-related services. In the Fog network, the data generated from the IoT/User devices are collected based on priority using big data from an urban environment [1]. The gathered data must be sent from the source to the destination for task-processing with the best possible accuracy and efficiency. Dependence on the internet and internet-based services is proliferating worldwide. The volume of data generated by such electronic devices will increase as more and more devices with internet access become a part of our daily lives. Global mobile data traffic is forecasted by ITU, which indicates that 607 Exa-Bytes (EBs) will be reached in 2025 and 5016 EBs in 2030 with M2M data. New approaches to analysing these enormous data streams are being studied. For instance, the Constrained Application Protocol (CoAP) [2] was recently developed to serve as a link between a wireless network and the global web (Internet). Additionally, sending data to the cloud will increase the time gap between a task's occurrence

and its completion. The current cloud service design is highly centralized, meaning that numerous applications can only be run in one location, called the data centre. Because of the rapid progress in cloud computing, the number of data centres is rapidly expanding. The energy demand of data centres will grow from 200 TW/h in 2018 to 2965 TW/h in 2030 [3]. Integrating the power of IoT, the Cloud, and Fog computing provides a significant opportunity to take educational institutions to the next level in terms of technology. Cloud-based data collection from sensors and learning platforms allows educators to track student progress and identify areas needing improvement. These technologies can create a more engaging, efficient, and accessible learning environment, empower educators, and improve the overall educational experience [4].

In response to these issues, Fog computing was introduced, which moves computation from the cloud to the edge device. This may result in shorter delays and less traffic to or from the cloud. A FC network allows for IoT applications by selectively finding nearby heterogeneous devices like PCs, data centres, gateways, and servers to overcome time-sensitive (high-priority) transmission and bandwidth issues [5]. One of the significant issues in FC is identifying the faults that occur in resource blocks (RBs), which act as a connecting path between an IoT layer and fog layer with multiple Fog nodes (FNs) to offload tasks among them during processing and provide reliable and fault-free transmission to all the connected IoT devices. However, identifying faulty resources is comparatively easy when determining the Fog node's fault. The resource block faults were identified and rectified with the existing OEeRA algorithm to provide effective task-offloading [6]. As an extension of the previous research, identifying the faults that occur in Fog nodes and providing a fault-free transmission is the focus of this research article. Due to the depletion of energy levels in the Fog node, neighbour node discovery is required to identify the alternate Fog nodes for reliable transmission in multi-hop Fog networks [7].

Overall, these strategies can be categorized into two categories based on whether they add efficient nodes or remove ineffective nodes. There are two approaches for adding efficient active FNs: the extraction of awaking nodes and collaborative neighbouring discovery nodes. The proposed method uses both approaches, resulting in conflict with other discovery methods. This indicates that two or more types of technologies can be combined. The inefficient FNs are replaced with efficient active FNs, which have a higher residual energy. Here, the FNs are subdivided into a fog control node for monitoring and fog access node for task-processing. The main motive of this work is to deploy the FoG–IoT network in a fault-free environment for task-processing using adequate fog access nodes. This encourages us to suggest a minimum-cost-based neighbouring candidate node discovery method for replacing the faulty nodes with alternate fog access nodes, where the faulty fog nodes are identified based on residual energy. To track the information request received from IoT by the newly assigned fog node, the replication and pre-emptive forwarding method is also suggested for task-processing, either in the same fog server or a neighbouring fog server.

In the proposed method, the aforementioned replacement and removal of inefficient Fog access nodes with a threshold ($Er < 0.1$) are monitored by the Fog control nodes present in the network. Fog control nodes identify active fog access nodes, which have a minimum cost, and act as an intermediary to offload the tasks received from IoT devices among other fog access nodes with appropriate residual energy. The proposed architecture aims to enhance the task-offloading process in a fault-free fog network by optimizing the energy level and cost. The main contributions of this research article are as follows:

- Propose a Priority Task-based Fault-Tolerance (PTFT) mechanism in Fog networks, which identifies the faulty Fog nodes with minimal residual energy.
  - Update the energy level of Fog access nodes automatically after a task is executed by the Fog control nodes or the Fog server.
  - The Fog control nodes also help to find the cost function for each Fog node with a minimum number of hops connected between them.

  - Identify and remove the faulty Fog nodes if the residual energy (Er) of the Fog node is less than 0.1.
  - Update the cost function of the Fog network.
- Propose a Minimum-Cost-based Neighbour Candidate Node Discovery (MCNCND) approach, which identifies the linked neighbouring candidate Fog access nodes with minimal costs for task-offloading.
  - Initially, the source Fog access node will broadcast its own cost to the linked neighbouring candidate Fog access nodes, and will receive the cost of all linked Fog access nodes.
  - Then, it will offload the task to the linked neighbouring Fog access node with the minimum cost.
  - This will be repeated until it finds the candidate Fog access nodes with the minimum cost in the Fog network.
- Develop and deploy the Replication and Pre-emptive Forwarding (RPF) process, which tracks the source information of the Fog access node.
  - Pre-empt and forward the source Fog access node information to the linked neighbouring candidate Fog access node during the task-offloading process within the source Fog server.
  - Replicate the source Fog node information in the neighbour(s) Fog server with minimum cost when all Fog access nodes are busy/faulty in the source Fog server.

This research work is organized as follows: Section 2 contains a detailed literature review of various existing node discover, replication, and forwarding approaches for reliable fault-free transmission in Fog computing. The proposed system model is discussed in Section 3, and Section 4 elaborates on the proposed algorithms. The experimental analysis and discussion are presented in Section 5, and Section 6 concludes the proposed research and provides directions for future research.

2. Related Works

The most important challenges addressed in this paper are how to identify the faulty Fog access nodes using a priority task-based fault-tolerance algorithm and reassign the task-processing to the new Fog access node with a minimum-cost-based neighbouring candidate node discovery algorithm. The paper also addresses how to select a Fog access node in a neighbouring server for task-processing via replication and a pre-emptive forwarding algorithm. Multiple techniques were suggested to ensure fault tolerance in Fog computing; some FT techniques employ task replication, which reduces the cloud's utilization of resources. Some other methods improve the mean response time by using checkpoint recovery to overcome faults.

2.1. Fault Tolerance and Node Discovery

The reactive fault-tolerant systems respond to failures after the occurrence of a fault. The failures are rectified after the reception of the request service, and then the responses are implemented. To detect the fault, the cloud or Fog statuses are regularly monitored and various responses, such as replication, checkpoint, and resubmission, are recorded [8]. Gabriele et al. [9] suggested the use of a Fault-Tolerance Generic Adaptive Interaction Architecture (FT-GAIA) software-based FT method for parallel and distributed environments. Mainly, this method deals with Byzantine faults and crash errors with the help of server restoration in the cloud layer, and this is achieved with the help of a replication mechanism. Semmoud et al. [10] suggested the use of a Replication and Pre-emptive Migration based Fault-Tolerance (RPMFT) technique to identify this fault through a distributed load-balancing algorithm by combining proactive and reactive fault-tolerance techniques. The authors in [11,12] suggested a fault-tolerance approach to reductions in cost and deadlines in order to reduce the failure rate in clustering networks. Peng et al. [13] proposed a WOA algorithm based on a multi-objective model without a fault-tolerance mechanism

for optimal task allocation via maximized system performance and efficiency. The authors in [14] proposed a scheduling algorithm that is efficient, low-cost, and FT to minimize the uncertainty and cost of these resources. They used spot and block-spot instances as hybrid instances to reduce the execution costs. Ghanavati et al. [15] suggested a new task scheduling strategy based on Dynamic Fault-Tolerant Learning Automata (DFTLA). They used variable-structure learning automata to identify the most efficient task assignment to fog nodes. Ramzanpoor et al. [16] proposed a multi-objective fault-tolerant optimization algorithm to reduce bandwidth wastage in a distributed environment.

Zareie et al. [17] used a hierarchical ranking algorithm with several centrality variables to find the appropriate means of (key) Fog node discovery in complicated networks for task-processing after the occurrence of a fault. Wang et al. [18] developed a new technique to measure the importance of key nodes in networks with more than two layers. They also checked the measurements in single/multi-relationship networks and combined networks. Ali Jaddoa et al. [19] proposed the Multi-critEria DecIsion support meChanism for IoT offloading (MEDICI) method to make a dynamic decision regarding task-processing, which could be either edge or cloud processing based on the requirements. The authors of [20] applied sequential decision-making with the Markova process, improved using the Lyapunov optimization method, to minimize an IoT system's operational costs while offering strict performance assurances.

2.2. Task Scheduling Based on Cost and Power Consumption

Skarlat et al. [21] proposed a genetic algorithm (GA) to solve the Fog task assignment and scheduling problem with the help of control nodes, which assign a service to the Fog control nodes or the Fog cells. The main motive of this optimization is to maximize the number of task assignments in the Fog network (instead of the cloud) to meet the requirements of user applications. In [22], the authors suggested a method for creating an environment that integrates scheduling, sequencing, and partitioning algorithms while ensuring a multi-objective optimization of the competing needs of users and providers. A dynamic-threshold-based task scheduling technique was presented to reduce the transmission and energy consumption of the IoT devices [23]. Misirli and Casalicchio [24] highlighted the need for efficient task scheduling in FC to address the challenges modelled by the distributed and resource-constrained environments. They provided various methods and metrics that can be employed to achieve optimal resource allocation and application performance. These earlier studies did not take fault tolerance into account, making them unsuitable for reliability applications. The Honey-Bee-Inspired Load-Balancing (HBI-LB) algorithm was proposed [25] to examine the load-balancing in the IoT-FoG network. Ranjan et al. [26] proposed a shortest-path resource allocation algorithm for the optimal allocation of resources in D2D, which could handle multiple users simultaneously. A component-based throttled load-balancing technique was proposed in [27], involving VM readers, a free VM owner, and free VM management elements. The VM readers read all VMs that are available. The free VM element temporarily stores free VMs until they are relocated to the free VM management element. The delay energy-balanced task-scheduling (DEBTS) algorithm was suggested to reduce average service latency and delay jitter by minimizing overall energy consumption [28]. Suleiman [29] proposed the framework to address the challenge of balancing cost, energy-efficiency, and QoS in cloud–Fog environments. They provided a method for scheduling tasks that minimizes the overall cost while ensuring service quality.

In [30], the authors stated that the processing power is the most critical factor in the Fog network when attempting to accomplish particular tasks. In the Fog network hierarchy, the network communication and storage resources are a few constraints that may differ during the efficient distribution of resources. Bozorgchenani et al. [31] examined partial task-offloading by assessing the use of centralized/distributed network in edge computing while considering the trade-off between energy consumption and Fog node delays. In [32], the authors developed a simulation of the sound classification system by combining Fog

computing and cloud computing for urban sound classification. To ensure the scalability of the proposed sensing system, they focused on the trade-off between power consumption, task runtime, and server latency. Jiang et al. [33] examined mobile devices in Fog systems and started a task-offloading strategy that reduces the energy consumption of the mobile devices, as well as the end-to-end response time, for task-processing.

3. Materials and Methods

3.1. Network Model

In this section, the IoT-FoG network model, which is considered for analysis, is presented. The general network model used for the IoT-FoG network is illustrated in Figure 1. It has three layers, namely, the cloud, Fog, and IoT layers. The Fog layer is composed of Fog servers and Fog nodes, which are connected to each other for efficient task-offloading activities.

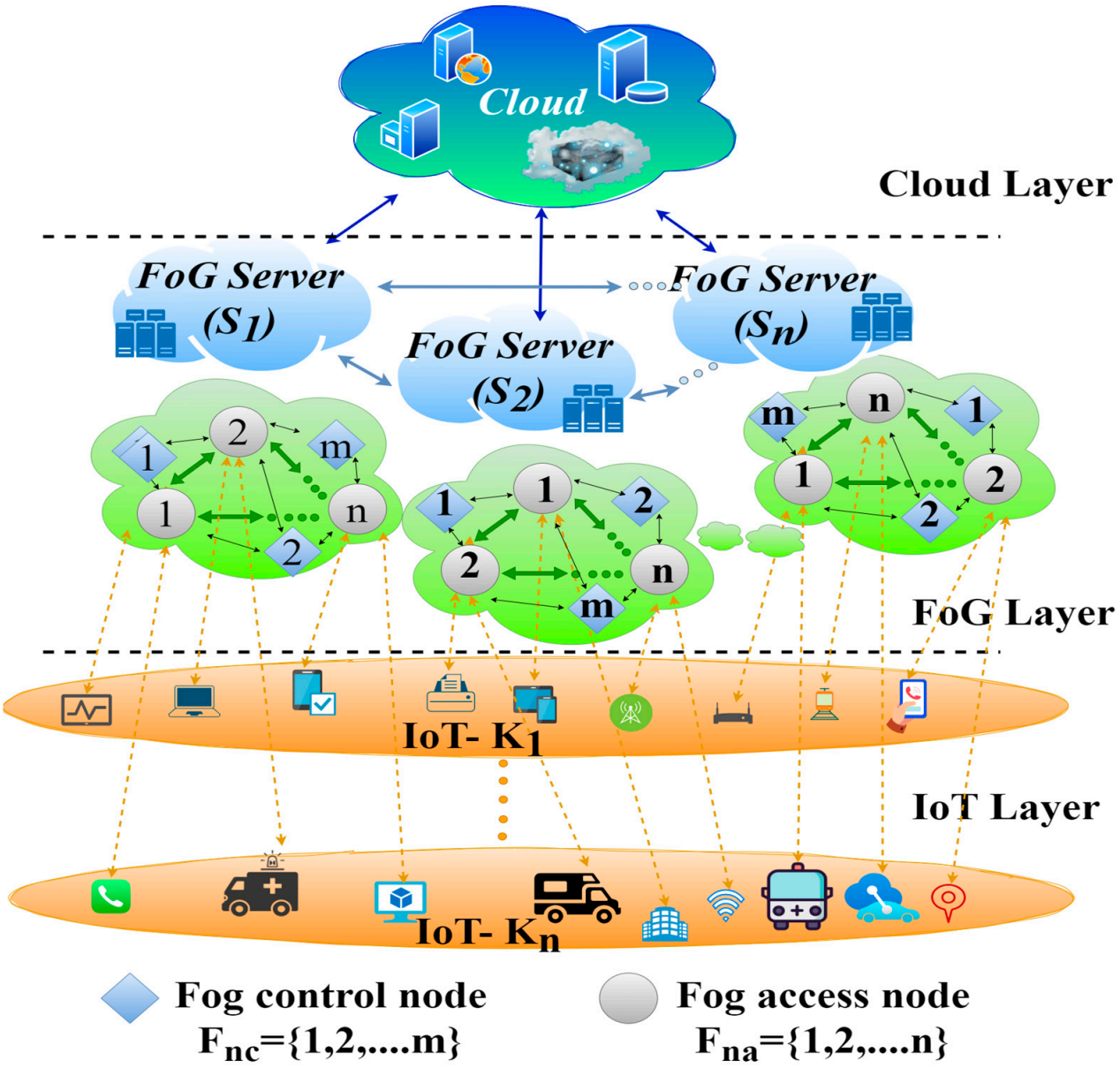


Figure 1. Model of the IoT-FoG network.

Table 1 represents the symbols and notations used in the proposed methods.

Table 1. Notations and symbols used.

Symbol	Meaning
BW	Bandwidth
$c_{k,na}$	Channel cost between device and Fog access node
N_0	Channel noise power
CC_i^{max}	Computational complexity
C_{lai}	Cost of initiator/source Fog access node
C_{laj}	Cost of linked neighbouring candidate Fog access node
C_f	Cost of the Fog node
C_{Sj}	Cost of the neighbouring Fog server
D_i	Deadline
E_{exe}	Energy consumption for the task during its execution
E_{up}	Energy consumption of the task while transmitting from K_k to $\mathbf{F}_{na}$
T_{exe}	Execution time
$N(F)$	Expected number of faults
FAt_n	Fog node active time
$\mathbf{F}_{lai}$	Initiator/source Fog access node
$\mathbf{F}_{si}$	Initiator/source Fog server
$\mathbf{F}_{lan}$	Linked neighbouring candidate Fog access node for task-processing
BL	Maximum battery life of the Fog nodes
$_f(t)$	Mean failure rate
$\mathbf{F}_{laj}$	Neighbouring Fog access node
$\mathbf{F}_{sj}$	Neighbouring Fog server
d_{ij}	Number of hopes
$\mathbf{K}_k$	Number of user/IoT devices
p_{na}	Power consumption of Fog access node
p_k	Power consumption of user/IoT device
P_i	Priority
δ	Priority threshold level
$\mathfrak{p}$	Probability of fault occurrence
E_r	Residual energy
σ_j	Selectivity of all possible neighbouring candidate Fog access nodes
σ_c	Selectivity of the Fog control node
t_i	Set of attributes
$\mathbf{F}_{na}$	Set of Fog access nodes
$\mathbf{F}_n$	Set of Fog nodes
$\mathbf{F}_{nc}$	Set of the Fog control nodes
$\mathbf{F}_s$	Set of the Fog servers
SC_i^{max}	Storage capacity
t_c	Task computation
X_i	Task size/length of the task
t_y	Task type
T_{up}	Time taken to forward the task from IoT device to the Fog access nodes
BC_i^{max}	Total battery capacity
$E_{consumed}$	Total energy consumption
T_{tot}	Total time
$C_{t/(k\rightarrow na)}$	Transmission channel capacity

The set of Fog servers is denoted as $\mathbf{F}_s$ = {S_1, S_2, ……., S_s}, the Fog Node (FN) set is defined as $\mathbf{F}_n$ = {N_1, N_2, ……, N_n}, and the set of user/IoT devices are denoted as $\mathbf{K}_k$ = {K_1, K_2,…..K_k}. In this proposed model, the task-offloading will take place in three different stages, as follows:

1. Task-offloading within a single Fog server;
2. Task-offloading by a neighbour or another Fog server;
3. Task-offloading by the cloud centre.

Efficient task-offloading is obtained based on the availability of active Fog access nodes in the proposed model. The Fog node (FN) can be subdivided into two types: (1) Fog control nodes ($\mathbf{F}_{nc}$) and (2) Fog access nodes ($\mathbf{F}_{na}$). One or more Fog control nodes will control

every Fog access node. The Fog control node gathered the details of residual energy levels and the task priorities of user/IoT data to process the task received through the corresponding Fog server. According to this, it will control the task-offloading in Fog access nodes for efficient task-processing. The Fog control nodes will estimate their own cost and find the cost of all possibly linked neighbouring candidate Fog access nodes with a minimum number of hops between the Fog control nodes and the corresponding candidate Fog access nodes.

3.2. Communication Model

The tasks generated from any IoT/User devices are represented as a set of $T = \{t_1, t_2, \ldots, t_n\}$. In this, each element, t_i, will describe as a set of attributes, as follows:

$$t_i = \{X_i, t_c, t_y, D_i\} \quad (1)$$

where (i = 1, 2,…, n); X_i stands for the size/length of the task to be transmitted to the Fog access nodes for task-processing; t_c represents the required computations for task execution; t_y denotes the task type (i.e., time-sensitive or time-tolerance); D_i represents the task deadline for completion. The Fog nodes ($\mathbf{F}_n$) set will be divided further into Fog control nodes, $\mathbf{F}_{nc} = \{N_{c1}, N_{c2}, \ldots\ldots, N_{cm}\}$, for monitoring and controlling actions and Fog access nodes, $\mathbf{F}_{na} = \{N_{a1}, N_{a2}, \ldots\ldots, N_{an}\}$, for task-offloading, which can also be represented by the following details:

$$Fn = \{SC_i^{max}, CC_i^{max}, BC_i^{max}\} \quad (2)$$

where i = 1,2,…n, SC_i^{max} refers to the storage capacity, CC_i^{max} gives the computing capacity, and BC_i^{max} denotes the total battery capacity of the Fog access nodes. The total time taken for user/IoT task assignment to the Fog access nodes is described as follows:

$$T_{tot} = T_{up} + T_{exe} \quad (3)$$

where T_{up} denotes the time taken to forward and offload tasks to a Fog access node within the same or a neighbouring server, and T_{exe} refers to the execution time of the task in $\mathbf{F}_{na}$. T_{up} is related to the transferral of data from the user/IoT device to the selected Fog access node. From this, the T_{up} is defined as the ratio of task size/length (X_i) to the transmission capacity ($C_{t/(k\to na)}$) between the corresponding user/IoT device (k) and Fog access node (na), and is expressed as follows:

$$T_{up} = \frac{X_i}{C_{t/(k\to na)}} \quad (4)$$

where

$$C_{t/(k\to na)} = BW \times \log_2\left(1 + \frac{p_k \times c_{k,na}}{N_0 \times BW}\right) \quad (5)$$

$C_{t/(k\to na)}$ is calculated by Shannon's formula, where p_k indicates the power consumption of the user/IoT device, $c_{k,na}$ refers to the channel cost between the $\mathbf{K}_k$ and $\mathbf{F}_{na}$, N_0 gives the channel noise power (dBm/Hz), and BW denotes the bandwidth (Hz). The execution time of the task on the Fog access node is defined as the number of tasks and their size/length, multiplied by the required computation and divided by the computational complexity $\left(CC_i^{max}\right)$ of the Fog access node, and is expressed as follows:

$$T_{exe} = X_i \times \frac{t_c}{CC_i^{max}} \quad (6)$$

where the computational complexity of the proposed methods mainly depends on the number of freely available candidate fog access nodes in a fog network, and this can be represented as $O\{\mathbf{F}_s[\min(\mathbf{F}_{lan})]\}$. This will find the minimum-cost Fog access node within a server that can be used for task-processing.

3.3. Energy and Priority Model

The total energy consumption of task-processing is obtained using three phases: energy consumption due to task transmission between IoT and FN, task-offloading at FN, and task transmission from FN to IoT after processing. The consumption of energy while transmitting the task from IoT device to the Fog access node is denoted as E_{up}, and the consumption of energy during task execution is denoted as E_{exe}. This can be represented mathematically as follows:

$$E_{consumed} = E_{up} + E_{exe} \tag{7}$$

where

$$E_{up} = T_{up} \times p_k \tag{8}$$

$$E_{exe} = T_{exe} \times p_{na} \tag{9}$$

where p_k and p_{na} denote the power consumption of the IoT device and F_{na}, respectively, during task execution.

When the IoT devices generate a task for processing, the corresponding Fog server identifies the priority level of the task and monitors the residual energy level for each FN. The faulty FNs are determined based on the residual energy level and removed from the Fog networks before task-offloading activities. The residual energy (Er) is obtained by subtracting the energy consumed by the Fog access nodes during task-processing from the initial energy, and the energy level rate can be obtained by dividing the obtained result by the initial energy.

$$E_r = \frac{E_{initial} - E_{consumed}}{E_{initial}} \tag{10}$$

The proposed model considered the characteristics of residual energy for fault identification. The residual energy mainly depends on the battery life of the Fog nodes, which is represented by Equation (11):

$$BL = \frac{E_r}{I_{avg}} \tag{11}$$

where BL defines the battery life of the Fog nodes, and this can be obtained by dividing the residual energy by the average current. The residual energy is also used to estimate the active time of the Fog nodes during task-processing, and these relations are represented below:

$$FAt_n = \frac{E_r}{X_i \times \text{duty cycle}} \tag{12}$$

The energy level of each node [28] will be categorized into three levels, as follows:

1. $Er \geq 0.3 \rightarrow$ used for executing high-priority (time-sensitive) tasks from the user/IoT devices.
2. $(0.1 \leq Er < 0.3) \rightarrow$ used for executing low-priority (time-tolerance) tasks, and may also act as a reserve node for some forms of task-processing.
3. $Er < 0.1 \rightarrow$ identified as faulty FN because the battery life is too short, and this indicates that the FNs will not complete the task efficiently with in the battery's lifespan.

3.4. Fault Model

Faults have become common in the context of the IoT-FoG networks. As a result, some Fog access nodes may experience partial or whole faults during task-offloading activities, based on their minimum energy levels after the execution of multiple tasks within a particular period. In this proposed method, fault identification depends on the residual energy that remains in the Fog access nodes during task-processing. The number of faulty nodes within a specified period and the fault rate are identified based on the residual energy in the fog access nodes, which can be determined by continuously monitoring energy levels via fog control nodes. As a result, the expected number of faults during a specific time interval, from (0 to t), can be obtained using the following relation:

$$N(F) = \int_0^t \mu_f(t)dt \tag{13}$$

where $\mu_f(t)$ is the mean fault rate of the Fog nodes, and the probability of fault occurrence (ƥ) in the Fog nodes during the period of task-offloading, obtained using the total number of faults that occurred $T(ƥ)$ within a specified time, with a range of $0 \leq P(ƥ) \leq 1$, is given as follows:

$$P(ƥ|\mu_f) = \frac{{\mu_f}^{ƥ}}{T(ƥ)} e^{-\mu_f} \tag{14}$$

3.5. Cost Model

The following equation can be used to obtain the cost of Fog control and access nodes:

$$C_f = \sigma_c r_i + \sum_{j \in c} \sigma_j d_{ij} \tag{15}$$

where σ_c is the selectivity of the Fog control nodes, and σ_j represents the selectivity of all possibly linked neighbouring candidate Fog access nodes for task-offloading. Here, the selectivity of all Fog access/control nodes should be maintained as one ($\sigma_j = 1$ and $\sigma_c = 1$); r_i indicates the number of hops between the source Fog control node $(r_i = 0)$ and d_{ij} denotes the minimum number of hops between the Fog control nodes and the Fog access node. To estimate the minimal cost of a Fog network, a minimum of three Fog control nodes are considered to be necessary for efficient cost estimation, which can be obtained by selecting the minimum number of hops between the nodes. The minimum-cost Fog access node is selected for task-offloading via the minimum-cost-based neighbour candidate discovery method for efficient task-processing. In some cases, the source server itself has minimum-cost fog access nodes and can be selected using Equation (16). If the source server has busy nodes, then it will select the minimum-cost fog access nodes from the neighbouring server using Equation (17):

$$sel(C_f) = \min\left\{ \sigma_c r_i + \sum_{j \in c} \sigma_j d_{ij}; \text{Source } F_s \right\} \tag{16}$$

$$sel(C_f) = \min\left\{ \sigma_c r_i + \sum_{j \in c} \sigma_j d_{ij}; \text{NeighbourF}_s \right\} \tag{17}$$

3.6. Problem Formulation and Objective Function

In the network model considered in this research, the tasks generated by the IoT devices are evaluated and processed based on this multi-objective optimization problem, such as optimizing the cost, fault, energy, etc. The issues encountered during task-offloading are addressed by two objective functions, as follows:

(1) Fault minimization—the prior identification of faulty nodes and immediate task-offloading to the alternate candidate Fog access nodes, either in the same Fog server or a neighbouring Fog server, without disconnecting the task-processing activities.
(2) Residual energy optimization—the continuous monitoring of residual energy levels after the completion of each task by candidate fog access nodes. The optimization can be achieved by selecting the maximum residual energy for high-priority tasks and minimum residual energy for low-priority tasks.

$$\min(N(F)) = \min\left\{ \int_0^t \mu_f(t)\, dt; \text{Candidate } F_{na} \right\} \tag{18}$$

$$Opt(E_r) = \max\{E_r; \text{high priority}(X_i)\} \tag{19}$$

$$Opt(E_r) = \min\{E_r; \text{low priority}(X_i)\} \tag{20}$$

This is subject to:

$$\left.\begin{array}{rr} C1: & F_{nc} > 3 \\ C2: & E_r \geq 0.1 \\ C3: & F_{na} > 1 \\ C4: & F_s > 1 \\ C5: & F_{na} > F_{nc} \\ C6: & \sum_{i=1}^{n}\sum_{j=1}^{m} F_{nai}\, F_{ncj}; E_r \geq 0.3; \mathit{Active\ fog} \\ C7: & \sum_{i=1}^{n}\sum_{j=1}^{m} F_{nai}\, F_{ncj}; 0.1 \leq E_r \leq 0.3; \mathit{sleepy\ fog} \\ C8: & \sum_{i=1}^{n}\sum_{j=1}^{m} F_{nai}\, F_{ncj}; E_r < 0.1; \mathit{faulty\ fog} \end{array}\right\} \quad (21)$$

The multi-objective function of the formulation of problems during task-offloading in the distributed IoT-FoG networks is presented in Equations (18)–(20). The proposed methods will minimize the occurrence of faults in the network by continuously monitoring the residual energy levels of all nodes and task-offloading based on the priority level.

4. Proposed Fault-Tolerance Schemes

This section discusses the various proposed mechanisms and algorithms for efficient fault-free task-offloading activities in Fog networks.

4.1. Priority Task-Based Fault-Tolerance (PTFT) Algorithm

The PTFT algorithm is proposed to identify the faulty Fog access nodes based on the residual energy, and the Fog control nodes are used to select the appropriate Fog access node for task-processing. The task generated by the user/IoT devices is initially received by the nearest Fog access node (source node). Based on the remaining residual energy in the fog access nodes and the priority level of the task, the Fog control nodes will identify the appropriate Fog access nodes for task-offloading with the help of the neighbouring candidate node discovery method. This proposed method uses parallel task-processing activities via the effective utilization of resource-sharing and nodes during task-offloading in an IoT-FoG network. In the first case, for time-sensitive (high-priority) tasks, if the source node has residual energy levels that are more significant than 0.3, then the source node itself acts as a processing node for task-processing. Otherwise, if the residual energy is less than 0.3, it will identify the next neighbouring candidate Fog access node in the same server or a neighbouring server for task-processing through the MCNCND and RPF algorithms. A few of these nodes can also be reserved as auxiliary nodes for reliable task-offloading. Finally, if the residual energy is less than 0.1, it is identified as a faulty node, and a new node will be discovered through the MCNCND algorithm for task-processing. In the second case, for time-tolerant (low-priority) tasks, if the source node has residual energy levels that more significant than 0.3, then those nodes will be utilized for a higher-priority task. The source node will forward the task to the neighbouring candidate Fog access nodes with residual energy ranging from 0.1 to 0.3 for task-processing. Otherwise, if the residual energy ranges from 0.1 to 0.3, then the node itself acts as a processing node.

The proposed PTFT is summarized in Algorithm 1. For a better understanding of the PTFT algorithm and a more accurate cost estimation of the Fog nodes, a simple example is proposed, as shown in Figure 2. The use-case network model contains two Fog servers, a few control nodes, and a Fog access node. The diamond shape represents the Fog control nodes, and the circle represents the Fog access nodes. Initially, the Fog server will estimate the residual energy level of the Fog nodes, and the Fog control nodes will identify the faulty Fog access nodes with the help of a threshold value (Er < 0.1). Once the faulty node is identified, it will immediately be removed from the Fog network. The cost of all Fog control and access nodes will be calculated by "(15)". In this model, the cost of the Fog nodes is calculated via the minimum number of hops between the Fog control nodes through the

Fog access nodes. For example, the initial cost of Fog control node N7 is calculated as follows:

$$C_7 = \sigma_c r_i + \sum_{j \in c} \sigma_j d_{ij} = 0 + \sum (1).d_{ij} = 0 + d_{7,4} + d_{7,3}$$

$$C_7 = 0 + 2 + 3 = 5$$

where $\sigma_c = 1$ and $\sigma_j = 1$ for all cases. For Fog control nodes, the initial source cost is 0 (i.e., $r_i = 0$), and the minimum number of hops between the next two Fog control nodes is added (i.e., the minimum number of hops between N7 and N4 is 2, and that between N7 and N3 is 3). The initial cost of Fog access node N9 is calculated as follows:

$$C_9 = \sum (1).d_{ij} = d_{9,7} + d_{9,11} + d_{9,4}$$

$$C_9 = 1 + 2 + 3 = 6$$

here, for Fog access nodes, the term $\sigma_c r_i$ is not considered because the selectivity of the Fog control node is zero and the minimum number of hops between three Fog control nodes was added (i.e., the minimum number of hops between N9 and N7 is 1, that between N9 and N11 is 2, and that between N9 and N4 is 3).

Algorithm 1 Priority Task-Based Fault-Tolerance (PTFT) Algorithm

Initialization: IoT Device, K_k = {K_1, K_2, …… K_k} and Fog Node, $\mathbf{F}_n$ = {N_1, N_2, ……, N_n}
Begin
Collect input from K_k
Check the task priority (Pi) and measure the residual energy (Er) of the FNs
If FN has a higher-priority task (Pi $\geq \delta$) // *Time-sensitive task*
If residual energy Er ≥ 0.3
Declare source Fog node as the processing node
Else
Forward the task to the neighbouring candidate Fog access node through MCNCND and RPF algorithms 2 and 3
End if
If residual energy ($0.1 \leq$ Er < 0.3)
Identify the next neighbouring candidate (auxiliary/reserve) node
Reserve and hold the auxiliary node until the execution of the current task
Else residual energy Er < 0.1
Declare it a faulty node
Remove the faulty node
Forward the task to the reserve node, called pre-emptive forwarding
New node is updated
End if
End if
If FN has a lower-priority task (Pi $\leq \delta$) //*Time-tolerance task*
If residual energy Er ≥ 0.3
Forward the task to the next neighbouring candidate node via Algorithm 2
New node is updated
Else residual energy ($0.1 \leq$ Er < 0.3)
The corresponding node itself processes the task and declares it a processing node
End if
If residual energy Er < 0.1
Declare it a faulty node
Remove the faulty node
Identify and forward the task to the next neighbouring candidate node via Algorithm 2
Pre-emptive forwarding process occurs (Algorithm 3)
New node is updated
End if
End if
End

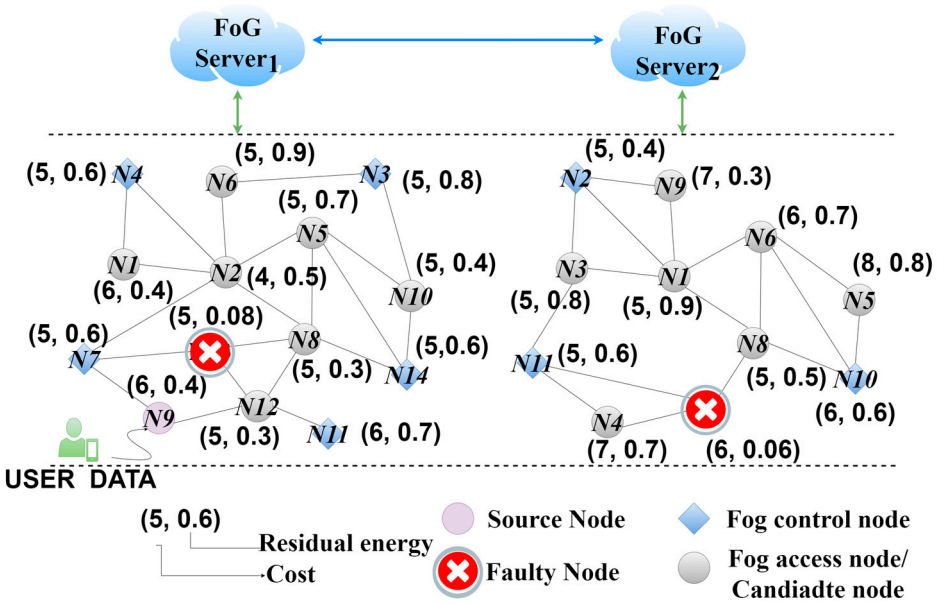


Figure 2. Identification of faulty Fog access nodes.

4.2. Minimum-Cost-Based Neighbour Candidate Node Discovery (MCNCND) Algorithm

The MCNCND algorithm is proposed to discover the alternate candidate Fog access node to achieve task-offloading with minimal cost, which the Fog control nodes will compute. The initial cost of the Fog nodes is calculated and broadcast using all linked neighbouring Fog access nodes. Then, the algorithm will identify whether the node is in a candidate (ready to process) or busy (already processing) state. In the first case, the source server has candidate Fog access nodes; once the task is generated by the IoT/user device, the nearest Fog access node will gather the task for processing, and the cost of all linked neighbouring Fog access nodes is received. The candidate Fog access node with the minimum cost or the exact cost is selected. The task will be offloaded to the neighbouring candidate Fog access node with the minimum cost. This process will be repeated until the selection of the minimum-cost candidate Fog access node in the Fog network. Then, the finalized minimum-cost candidate Fog access node will be declared a processing node.

In the second case, the source server does not have a candidate node, and the generated task is directly offloaded to the source server. It identifies the neighbouring Fog server with candidate Fog access nodes for task-processing. Finally, the task-processing is achieved by the candidate Fog access node with the lowest cost in the neighbouring server. It declares the corresponding node the processing node for task-processing.

To obtain a better understanding of the first case presented in Algorithm 2, a simple example is illustrated in Figure 3, which contains two Fog servers, along with their Fog nodes. Server 1 has 13 Fog nodes, with 5 Fog control nodes and 8 Fog access nodes, and server 2 has 10 Fog nodes, with 3 Fog control nodes and 7 Fog access nodes.

Initially, the user's device sends the task for processing to Fog access node N9; then, this node will act as an initiator (source node), and receives the cost value for all linked neighbouring candidate Fog access nodes by sending a request. In this scenario, Fog access node N9 has the linked neighbouring candidate Fog access node, denoted as N12, and also has a reduced cost compared to N9. Therefore, the task is offloaded to N12, and again will request the cost of the linked neighbouring candidate Fog access nodes. Then, it is identified as node N8, with the exact cost. To further check the remaining Fog access nodes, the task is offloaded to the N8 node. Again, it identifies the minimum cost from N2 for task-offloading. Finally, N2 is declared a processing node for task-processing because it has the minimum cost compared with other linked neighbouring candidate Fog access nodes.

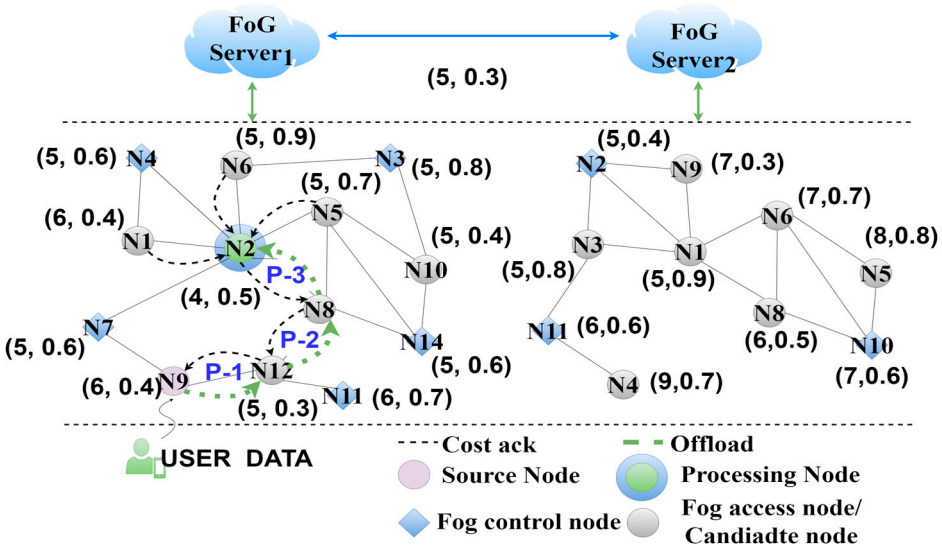


Figure 3. Case 1: Neighbouring candidate node discovery within the source Fog server.

For the second case in Algorithm 2, a simple example is taken from case 1 of the MCNCND algorithm by considering all Fog access nodes and the busy states in the source Fog server. The structure of this is illustrated in Figure 4. Initially, the user device sends the task for processing to Fog access node N9, which acts as an initiator. The task will be offloaded to the neighbouring Fog server, which contains a minimum cost or the exact cost. Here, there are only two servers, so it offloads the task to server 2. Next, server 2 will find the next candidate Fog access node for task-processing; here, Fog access node N9 is identified as the initiator. By using the MCNCND algorithm, the node for task-processing will be found. Fog access node N9 will request the list of all linked neighbouring candidate Fog access nodes' costs; the cost of N1 is shown to be the minimum, so it then offloads the task to N1 for processing. Then, N1 will request the cost list and identify itself having the minimum cost as compared with other linked neighbouring candidate Fog access nodes. Finally, N1 is declared as the processing node for task-processing. The proposed MCNCND algorithm is summarized in Algorithm 2.

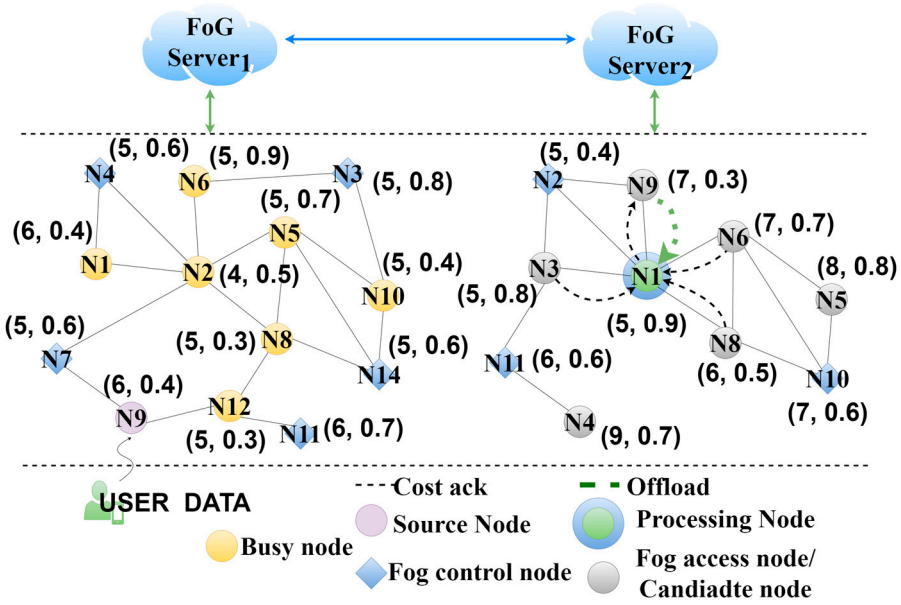


Figure 4. Case 2: Neighbouring candidate node discovery in neighbouring Fog server.

Algorithm 2 Minimum-Cost-Based Neighbour Candidate Node Discovery (MCNCND) Algorithm

1: **Initialization**: Fog server, $\mathbf{F}_s = \{S_1, S_2, \ldots\ldots\ldots, S_s\}$,
2: Fog Node, $\mathbf{F}_n = \{N_1, N_2, \ldots\ldots, N_n\}$;
3: Fog control node, $\mathbf{F}_{nc} = \{N_{c1}, N_{c2}, \ldots\ldots, N_{cm}\}$,
4: Fog access node, $\mathbf{F}_{na} = \{N_{a1}, N_{a2}, \ldots\ldots, N_{an}\}$ with the condition of (a > c)
5: **Begin**
6: **For** i = 1,2,3, …., k
7: Receive task from IoT/user device
8: If $\mathbf{F}_{si} \neq \mathbf{F}_{lan}$ {Φ }//*Source server with the neighbouring candidate Fog access nodes*
9: Calculate and broadcast the initial cost of Fog control node $C_{c1}, C_{c2}, \ldots, C_{cm}$
10: Initiator (i) (source node) computes its own cost, C_{lai}
11: Initiator sends a request to every linked neighbouring candidate Fog access node '$\mathbf{F}_{lai}$' in the same server
12: Fog control nodes ($\mathbf{F}_{nc}$) will compute the cost function of all possible linked fog access nodes as follows: $C_{la1}, C_{la2}, \ldots, C_{lan}$
13: The initiator node will receive the cost list of all linked neighbouring nodes and Identify the Fog access node, stating whether the node is a candidate node or a busy node.
14: Select the min-cost candidate node, $C_{laj} = \min\{ C_{la1}, C_{la2}, \ldots, C_{lan} \}$, and select the corresponding candidate Fog access node $\mathbf{F}_{laj}$ for task-processing
15: If $\mathbf{F}_{lai}$ (C_{lai}) < $\mathbf{F}_{laj}$ (C_{laj})//*cost of initiator is less than neighbouring nodes*
16: Declare the initiator node as $\mathbf{F}_{lai}$ (C_{lai}), a processing node
17: Otherwise:
18: Consider the min-cost neighbouring candidate Fog access node as the initiator node for further processing (replace $\mathbf{F}_{laj}$ with $\mathbf{F}_{lai}$)
19: End if:
20: Repeat from line 13 to find the min-cost candidate Fog access node $\mathbf{F}_{lan}$
21: Otherwise: //*Source server with no candidate Fog access nodes*
22: For $\mathbf{F}_{sj}$, j = 1, 2, 3, …. , n−1 //*neighbouring Fog server*
23: Calculate the cost of all neighbouring Fog servers $\mathbf{F}_{sj}$ (C_{Sj})
24: End for:
25: Identify the next min-cost neighbouring $\mathbf{F}_s$, $C_{Sj} = \{\min (C_{S1}, C_{S2}, \ldots, C_{Ss})\} \rightarrow \mathbf{F}_{sj}$
26: Select the min-cost neighbouring Fog server ($\mathbf{F}_{sj}$) with candidate Fog access nodes for task-processing
27: Select the nearest candidate Fog access node ($\mathbf{F}_{lai}$) as the initiator in the respective Fog server ($\mathbf{F}_{sj}$)
28: Fog control nodes ($\mathbf{F}_{nc}$) will compute the cost function of all possible linked Fog access nodes as follows: $C_{la1}, C_{la2}, \ldots, C_{lan}$
29: The initiator node receives the list of the cost of all linked neighbouring nodes and identifies the Fog access node states, and whether the node is a candidate node or a busy node.
30: Select the min-cost candidate node, $C_{laj} = \min\{ C_{la1}, C_{la2}, \ldots, C_{lan} \}$, and select the corresponding candidate Fog access node $\mathbf{F}_{laj}$ for task-processing
31: If $\mathbf{F}_{lai}$ (C_{lai}) < $\mathbf{F}_{laj}$ (C_{laj})//*cost of initiator is less than neighbouring nodes*
32: Declare the initiator node, $\mathbf{F}_{lai}$ (C_{lai}), a processing node
33: Otherwise:
34: Consider the min-cost neighbouring candidate Fog access node as the initiator node for further processing (replace $\mathbf{F}_{laj}$ with $\mathbf{F}_{lai}$)
35: End if:
36: Repeat from line 29 to find the min-cost candidate Fog access node $\mathbf{F}_{lan}$
37: End if:
38: **End for:**
39: **End**

4.3. Replication and Pre-Emptive Forwarding (RPF) Algorithm

The RPF algorithm is proposed to forward the task, carrying information from the source Fog access node to the selected neighbouring candidate Fog access node for task-offloading with the same Fog server or a neighbouring Fog server. The proposed Replication and Pre-emptive Forwarding (RPF) algorithm is summarized in Algorithm 3.

Algorithm 3 Replication and Pre-Emptive Forwarding (RPF) Algorithm

Initialization: Fog server, $\mathbf{F}_s = \{S_1, S_2, \ldots\ldots\ldots, S_s\}$, and Fog node, $\mathbf{F}_n = \{N_1, N_2, \ldots\ldots, N_n\}$

//Replication from S to S—replicating the entire source task information for fault-free task-processing

//Forwarding occurs from N to N within the same server; S→ Fog server and N→ Fog access node

Begin

If $\mathbf{F}_{si} \neq \mathbf{F}_{lan}$ {Φ}//*task-processing within the same server*

Find min-cost candidate fog access node $\mathbf{F}_{laj}$ $\{C_{laj}\} = \{\min (C_{la1}, C_{la2}, \ldots\ldots, C_{lan})\}$ from Algorithm 2

Check whether if $\mathbf{F}_{laj}$ is min-cost compared with other fog access nodes $(C_{laj} \leq C_{la1}, C_{la2}, \ldots\ldots, C_{lan})$

Otherwise, conduct a pre-emptive self-forwarding within the source server

Send the task to the neighbouring candidate Fog access node with min-cost $\mathbf{F}_{laj}$ $\{ C_{laj} \}$ for task-processing

End if:

Otherwise, $\{ \mathbf{F}_{lai} \rightarrow \mathbf{F}_{sj} \}$, conduct a replication//*initiator forwards the task to neighbouring Fog server*

Find $\mathbf{F}_{sj}$ $\{C_{Sj}\} = \{\min (C_{S1}, C_{S2}, \ldots\ldots, C_{Ss})\}$//*Fog server with min-cost*

Replicate and pre-emptively forward the task to neighbouring server $\{\mathbf{F}_{si} \rightarrow \mathbf{F}_{sj}\}$

Find min-cost neighbouring Fog access node $\mathbf{F}_{laj}$ $\{C_{laj}\} = \{\min (C_{la1}, C_{la2}, \ldots\ldots, C_{lan})\}$ and update the process as $\{\mathbf{F}_{sj} \rightarrow \mathbf{F}_{laj}\}$ replication

Declare min-cost neighbouring Fog access node $\mathbf{F}_{laj}$ as a processing node

End if:

End

In case 1 of the MCNCND algorithm, the task-offloading is achieved with the source server itself by finding the minimum-cost candidate Fog access node for task-offloading, and this process is called pre-emptive self-forwarding. The replication will be performed in case 2 of the MCNCND algorithm due to the unavailability of candidate Fog access nodes within the corresponding source Fog server. In this condition, the task will be directly replicated with the neighbouring Fog server that has the minimum cost and the most appropriate candidate Fog access node will be selected for task-processing.

5. Results and Discussion

This section explains the experimental parameter setup and provides a comparative result analysis of the proposed algorithms for various scenarios using existing methods. The proposed algorithms are simulated and analysed with the MATLAB R2022a software platform in Intel Core i5 HP computers with 8 GB RAM to evaluate the performance of task-offloading activities, and this is compared with other existing fault-tolerant algorithms. Table 2 represents the simulation parameters used for the proposed algorithms.

Table 2. Simulation parameters.

S.No	Parameter	Value
1	Fog server	5–50
2	Fog control node	20–100
3	Fog access node	50–5000
4	User device	5–1000
5	Number of tasks	50–1500
6	Task length	1000–10,000
7	Node capacity	1500 MIPS
8	Initial node energy	5 J
9	Task selection from IoT/user device	Priority based
10	Cloud data-center	1
11	Bandwidth	500
12	Processing speed	1200 MIPS
13	Fog memory space	1024 MB
14	Deadline of the task	Length based

5.1. Average Response Time for Various Tasks with Different Lengths

A Fog environment is created with a Fog server, Fog control nodes, Fog access nodes, and users to estimate the performance of the proposed PTFT algorithm.

In this research work, the proposed PTFT algorithm is tested and implemented for 1000 users with 50–1500 tasks, with the task length varying from 1000 to 10,000 bytes, 20 Fog servers, 100 Fog control nodes, and 1000 Fog access nodes in order to analyse the scalability of the proposed Fog-IoT networks. In this simulation environment, it is clearly observed that the proposed methods will efficiently work in a large-scale environment. The performance of the proposed PTFT algorithm in terms of the average response time is illustrated in Figure 5a for various tasks with a variable length, and in Figure 5b for multiple tasks of a variable length.

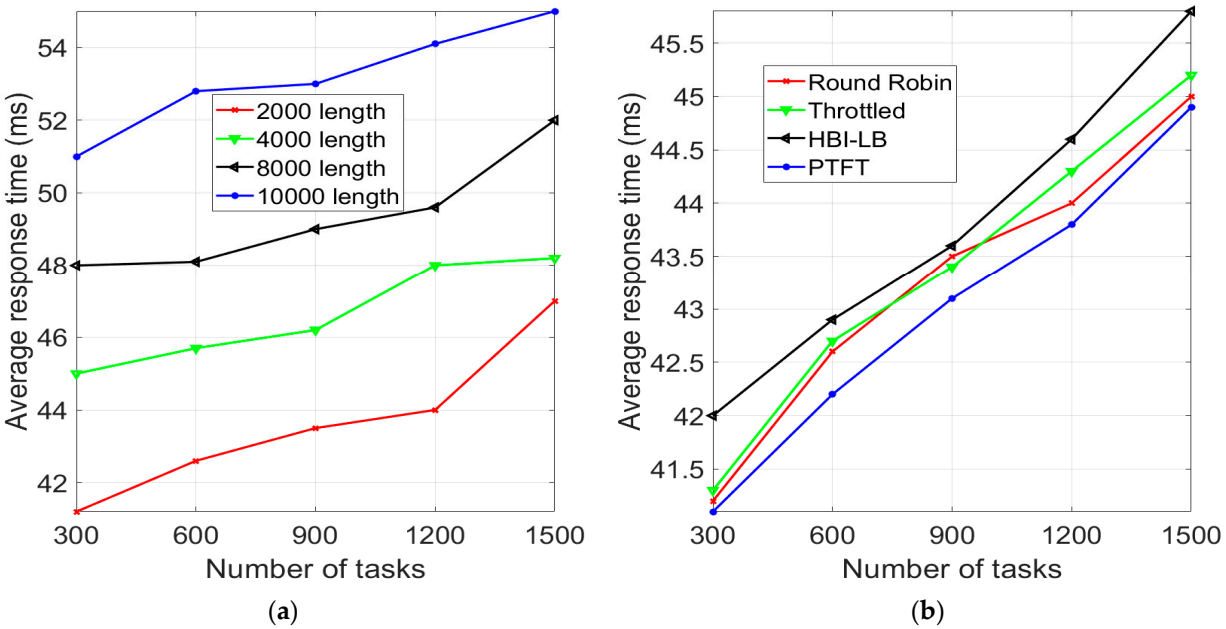


Figure 5. Comparison of average response time vs. number of tasks, (**a**) with a variable length; (**b**) with various algorithms.

It is observed that the average response time increased exponentially, while the tasks generated by the IoT device increased, and task length also increased. The maximum response time, 49.57 ms, occurred at a higher task count of 1500, with a task length of 5000. The proposed method provides shorter response times of 0.9 ms, 0.3 ms, and 0.1 ms for

HBI-LB [25], Throttled, and Round Robin [27], respectively. This is because an increase in both factors will directly affect the load of the Fog network and the availability of Fog access nodes during efficient task-offloading, significantly increasing the average response time. Figure 6a,b present a comparison of average response times with various existing algorithms, such as Round Robin, Throttled [27], and HBI-LB [25], for variable task lengths and multiple tasks, respectively. This is identified as the proposed PTFT algorithm with a shorter average response time of 1 ms, 0.3 ms, and 0.7 ms for HBI-LB [25], Throttled [27], and Round Robin, respectively.

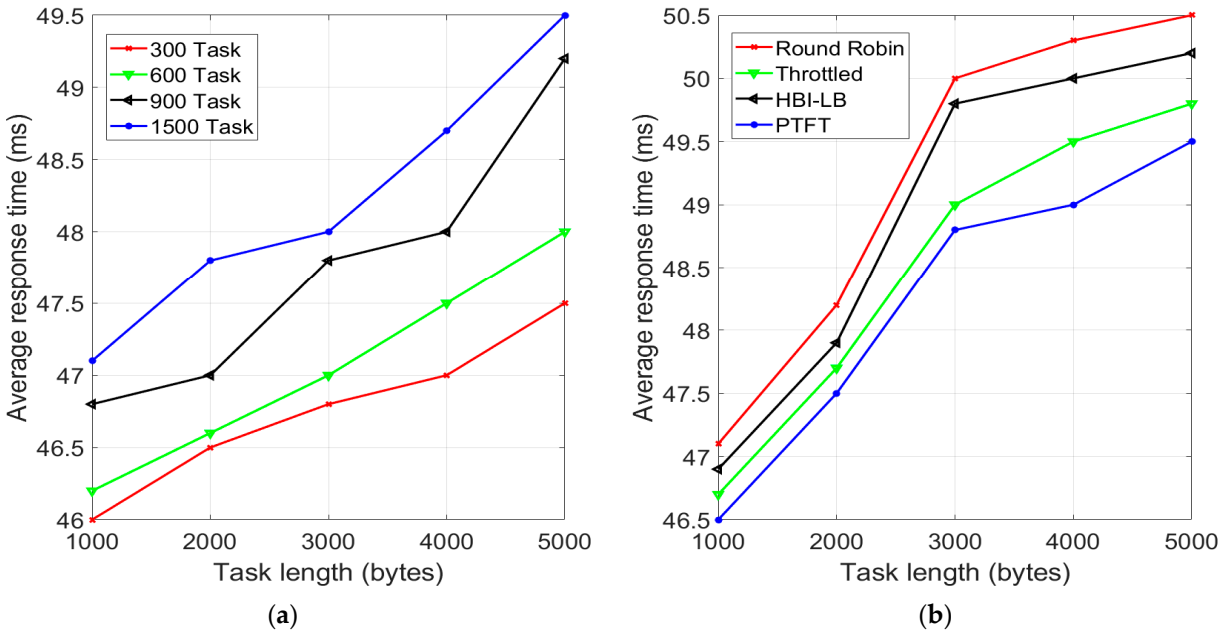
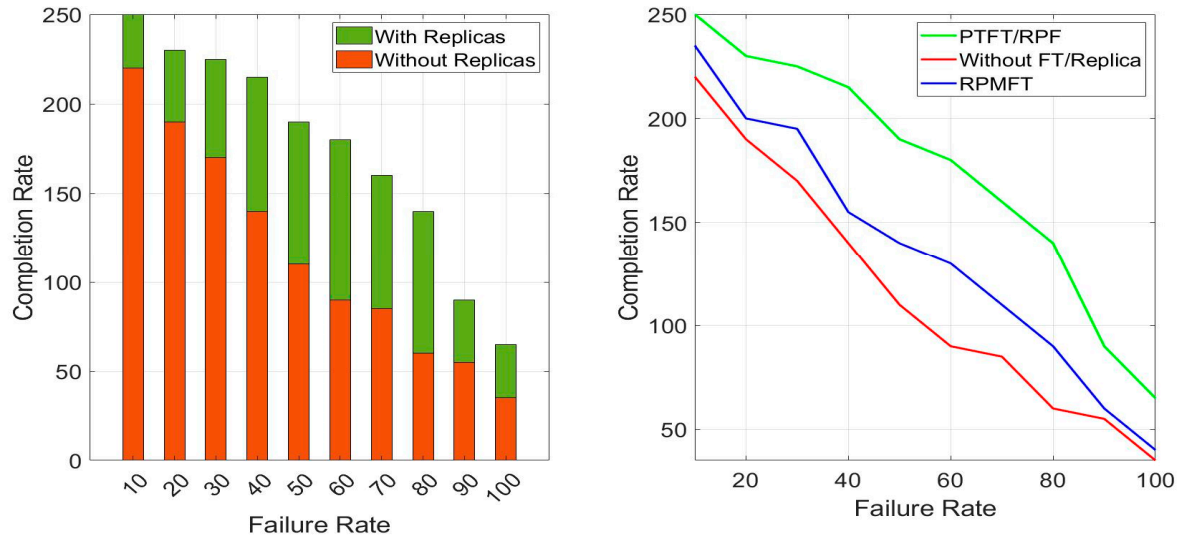


Figure 6. Comparison of average response time vs. task length (**a**) with multiple tasks; (**b**) with various algorithms.

This is achieved because the lowest-cost path with appropriate residual energy levels are selected based on task priority and the free availability of alternate candidate Fog access nodes for task-offloading following fault occurrence, while also keeping track of task-offloading activities in Fog network using RPF algorithm to avoid the under- and overutilization of Fog access nodes.

5.2. Task Completion Rate

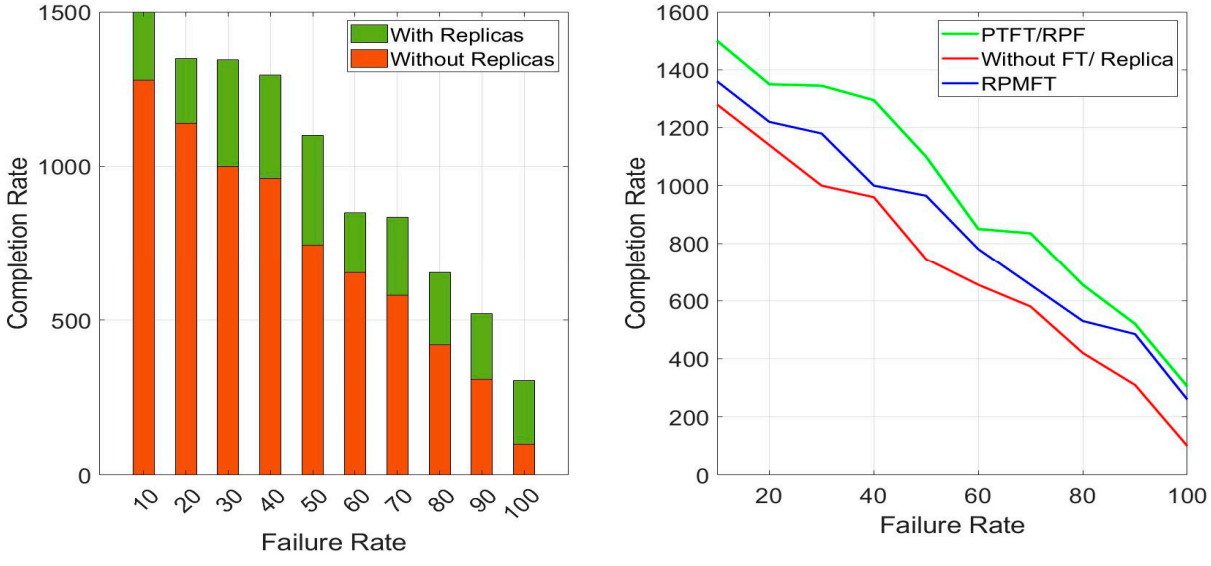
The proposed PTFT/RPF algorithm is simulated based on the task deadline (length) and the obtained results are compared with the existing RPMFT [10] method and replication without a fault-tolerance mechanism. Figure 7a,b illustrate the performance of the proposed PTFT method in the first scenario, showing the completion rate and failure rate for 250 tasks of random length generated by the IoT/user devices. It is observed that the proposed replication and pre-emptive forwarding method achieved a higher completion rate than other methods. Figure 7a indicates the importance of task completion using replication methods, and it is observed that, when 50%, 60%, and 80% failure rates are obtained, replicas are used to achieve efficient task completion during task-offloading activities in Fog networks. From Figure 7b, that the completion rate is shown to be improved for the proposed method, and an improvement in performance of 62.5% was observed compared to the RPMFT [10] method.



(**a**) (**b**)

Figure 7. Task completion rate vs. failure rate for the task count of 250 (**a**) with and without replicas; (**b**) for various algorithms.

In second scenario, Figure 8a,b present the performance of the proposed PTFT/RPF methods, showing the completion rate and failure rate for 1500 tasks of random length generated by the IoT/User devices. Figure 8a indicates the importance of task completion using replicas, and the 30%, 40%, and 50% failure rates were obtained when more replicas to achieve a better completion rate in Fog networks. Figure 8b shows that the completion rate is improved for the proposed method, and a 17.3% higher performance was reached compared to the RPMFT [10] method because the existing method uses only the replicas of specific tasks in the fault-tolerance mechanism, instead of giving equal priority to all tasks. In both scenarios, a better performance is achieved with the help of the prior availability of Fog access nodes for task-offloading when nodes are in a state of failure, as well as the proper information tracking of source Fog access nodes via replication and pre-emptive forwarding methods.



(**a**) (**b**)

Figure 8. Task completion rate vs. failure rate for the task count of 1500 (**a**) with and without replicas; (**b**) for various algorithms.

5.3. Task Acceptance Ratio and Completion Time

The task acceptance ratio of the proposed PTFT algorithm is compared with the existing fault-tolerance method without replication and forwarding in Fog networks. From Figure 9, it is observed that the proposed method outperformed the other methods, and achieved a higher acceptance of tasks with an increase in the number of Fog access nodes. This can be obtained with the help of efficient task-offloading activities in Fog networks, using replication and pre-emptive forwarding mechanisms, to candidates in the same Fog server or a neighbouring Fog server. In this, the acceptance rate of tasks was increased by 24% compared with existing methods without FT.

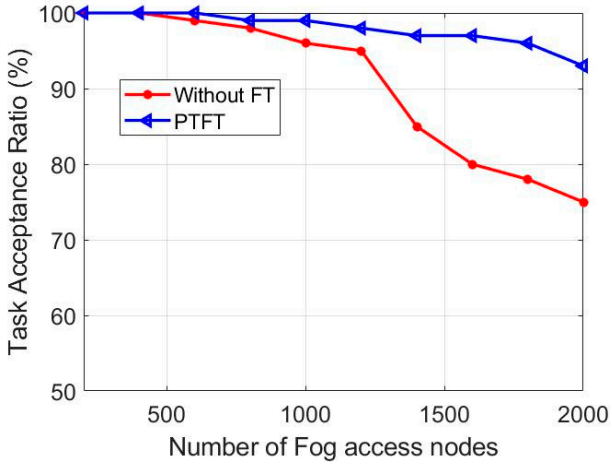


Figure 9. Task acceptance ratio vs. number of Fog access nodes.

The task completion time is measured for the proposed PTFT method, and it is compared to methods without replicas and fault-tolerance methods. Figure 10 shows that the proposed method has a minimum task completion time, and this can only be achieved because of the task-offloading in distributed Fog networks. In contrast, the other methods use a hierarchical placement for task-offloading in Fog access nodes, using the controlling function of the cloud data centres. Here, the task completion time is measured for various Fog access nodes, keeping the task size constant. The proposed method outperforms other methods, with a higher performance of 16.27% compared to those without fault tolerance and 29.84% compared to those without replicas.

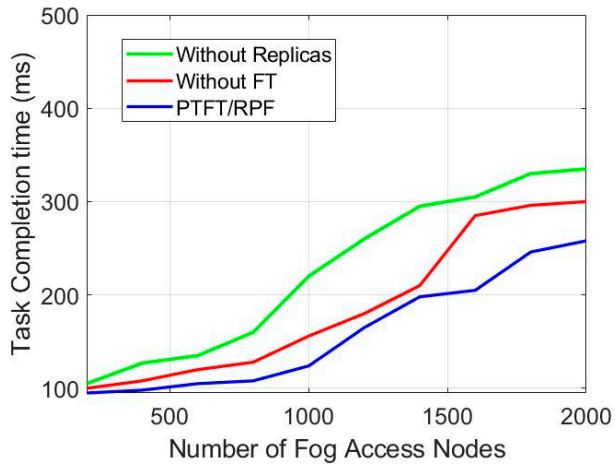


Figure 10. Task completion time vs. number of Fog access nodes.

5.4. Residual Energy

The residual energy of the proposed PTFT algorithm is compared with other methods, and the results are illustrated in Figure 11; the proposed method outperformed other methods and achieved task-offloading activities with lower energy consumption. Initially, each Fog node joins, with an initial residual energy level of 5 J, during the task-offloading process. As time increases, the number of tasks arriving from the user device will increase, and this will directly reduce the residual energy of the Fog access nodes after task-processing. The level of residual energy will be monitored continuously after and during task-processing by the respective Fog control nodes. Here, it is identified that the proposed PTFT method uses less energy for task-offloading and achieves a residual energy saving of 1.55 J compared to methods without replicas and 0.85 J compared to methods without the fault-tolerance method. Because the Fog access nodes select tasks based on priority and their remaining residual energy, this will directly reflect the inefficient energy utilization that occurs during offloading activities in Fog networks compared with other methods.

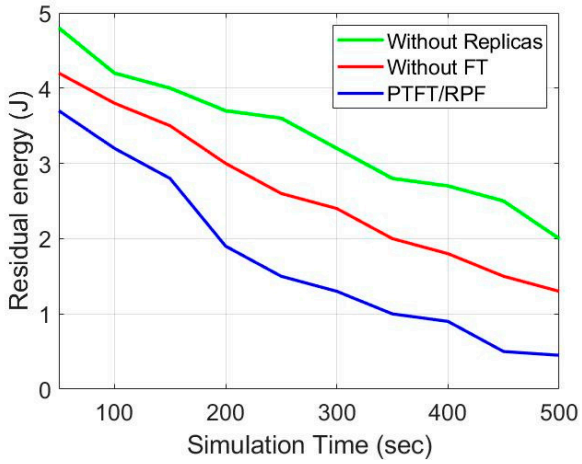


Figure 11. Residual energy vs. simulation time.

6. Conclusions

The importance of fault-tolerance mechanisms in a Fog environment during task-offloading activities was studied, and a relevant PTFT algorithm was proposed, which includes the identification of faulty Fog nodes and priority-based task assignment in Fog networks. Fault-free task assignment is necessary for the efficient and reliable utilization of Fog access nodes in a Fog environment. Considering this, a new MCNCND algorithm was proposed for task-offloading, and the obtained results were validated. The obtained results of the proposed methods show that they achieved a higher performance than existing methods in terms of the following: (i) the active number of Fog access nodes, with an average improvement of 41.2%, 37.82%, and 28.57% compared to the existing methods without FT, NFT-WOA, and DFTLA, respectively, using the minimum number of active Fog access nodes for task-offloading; (ii) average response time with a response time that was shorter by 0.9 ms, 0.3 ms, and 0.1 ms than the HBI-LB, Throttled, and Round Robin methods, respectively; (iii) the completion rate, with a 62.5% higher performance than the RPMFT method; (iv) acceptance ratio, with an improvement of 24%, and completion time, which outperforms the method without FT by 16.27% and the method without replicas by 29.84%; and (v) achieved energy savings, which are improved by 1.55 J compared to the system without replicas and 0.85 J for the system without an FT method in terms of residual energy.

The higher performance of the proposed results was only achieved because prioritized tasks were assigned to the Fog access nodes with the most efficient residual energy for task-processing. The prior availability of candidate Fog access nodes was ensured at the time the fault occurred, using RPF to track source nodes' information and to create a dynamic

network with scalability. Furthermore, in the future, malicious nodes will be identified in a dynamic Fog networking environment, and those nodes will be removed for efficient task-offloading. Additionally, an efficient trust-management scheme can be proposed for use in dynamic Fog-IoT environments to securely offload the task.

Author Contributions: Conceptualization, P.P. and P.B.; methodology and software, P.B.; formal analysis and investigations, P.P. and P.B.; article drafting, P.B.; review, editing and supervision, P.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Badidi, E.; Mahrez, Z.; Sabir, E. Fog Computing for Smart Cities' Big Data Management and Analytics: A Review. *Future Internet* **2020**, *12*, 190. [CrossRef]
2. Tanganelli, G.; Vallati, C.; Mingozzi, E. Edge-Centric Distributed Discovery and Access in the Internet of Things. *IEEE Internet Things J.* **2017**, *5*, 425–438. [CrossRef]
3. Ben Salah, N.; Ben Saoud, N.B. Adaptive data placement in the Fog infrastructure of IoT applications with dynamic changes. *Simul. Model. Pract. Theory* **2022**, *119*, 102557. [CrossRef]
4. Badshah, A.; Rehman, G.U.; Farman, H.; Ghani, A.; Sultan, S.; Zubair, M.; Nasralla, M.M. Transforming Educational Institutions: Harnessing the Power of Internet of Things, Cloud, and Fog Computing. *Future Internet* **2023**, *15*, 367. [CrossRef]
5. Mahdikhani, H.; Lu, R.; Shao, J.; Ghorbani, A. Using Reduced Paths to Achieve Efficient Privacy-Preserving Range Query in Fog-Based IoT. *IEEE Internet Things J.* **2020**, *8*, 4762–4774. [CrossRef]
6. Premalatha, B.; Prakasam, P. Optimal Energy-efficient Resource Allocation and Fault Tolerance scheme for task offloading in IoT-FoG Computing Networks. *Comput. Netw.* **2024**, *238*, 110080. [CrossRef]
7. Chen, L.; Shu, Y.; Gu, Y.; Guo, S.; He, T.; Zhang, F.; Chen, J. Group-Based Neighbor Discovery in Low-Duty-Cycle Mobile Sensor Networks. *IEEE Trans. Mob. Comput.* **2015**, *15*, 1996–2009. [CrossRef]
8. Zhang, J.; Zhou, A.; Sun, Q.; Wang, S.; Yang, F. Overview on Fault Tolerance Strategies of Composite Service in Service Computing. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 9787503. [CrossRef]
9. D'angelo, G.; Ferretti, S.; Marzolla, M. Fault tolerant adaptive parallel and distributed simulation through functional replication. *Simul. Model. Pract. Theory* **2018**, *93*, 192–207. [CrossRef]
10. Semmoud, A.; Hakem, M.; Benmammar, B. A Distributed Fault Tolerant Algorithm for Load Balancing in Cloud Computing Environments. *E3S Web Conf.* **2022**, *351*, 01012. [CrossRef]
11. Zhang, P.; Chen, Y.; Zhou, M.; Xu, G.; Huang, W.; Al-Turki, Y.; Abusorrah, A. A Fault-Tolerant Model for Performance Optimization of a Fog Computing System. *IEEE Internet Things J.* **2022**, *9*, 1725–1736. [CrossRef]
12. Khaldi, M.; Rebbah, M.; Meftah, B.; Smail, O. Fault tolerance for a scientific workflow system in a Cloud computing environment. *Int. J. Comput. Appl.* **2020**, *42*, 705–714. [CrossRef]
13. Peng, H.; Wen, W.-S.; Tseng, M.-L.; Li, L.-L. Joint optimization method for task scheduling time and energy consumption in mobile cloud computing environment. *Appl. Soft Comput.* **2019**, *80*, 534–545. [CrossRef]
14. Kumar, S.M. Cost and fault-tolerant aware resource management for scientific workflows using hybrid instances on clouds. *Multimedia Tools Appl.* **2017**, *77*, 10171–10193. [CrossRef]
15. Ghanavati, S.; Abawajy, J.; Izadi, D. Automata-Based Dynamic Fault Tolerant Task Scheduling Approach in Fog Computing. *IEEE Trans. Emerg. Top. Comput.* **2020**, *10*, 488–499. [CrossRef]
16. Ramzanpoor, Y.; Shirvani, M.H.; Golsorkhtabaramiri, M. Multi-objective fault-tolerant optimization algorithm for deployment of IoT applications on fog computing infrastructure. *Complex Intell. Syst.* **2021**, *8*, 361–392. [CrossRef]
17. Zareie, A.; Sheikhahmadi, A. A hierarchical approach for influential node ranking in complex social networks. *Expert Syst. Appl.* **2018**, *93*, 200–211. [CrossRef]
18. Wang, D.; Wang, H.; Zou, X. Identifying key nodes in multilayer networks based on tensor decomposition. *Chaos Interdiscip. J. Nonlinear Sci.* **2017**, *27*, 063108. [CrossRef]
19. Jaddoa, A.; Sakellari, G.; Panaousis, E.; Loukas, G.; Sarigiannidis, P.G. Dynamic decision support for resource offloading in heterogeneous Internet of Things environments. *Simul. Model. Pract. Theory* **2019**, *101*, 102019. [CrossRef]
20. Urgaonkar, R.; Wang, S.; He, T.; Zafer, M.; Chan, K.; Leung, K.K. Dynamic service migration and workload scheduling in edge-clouds. *Perform. Eval.* **2015**, *91*, 205–228. [CrossRef]
21. Skarlat, O.; Nardelli, M.; Schulte, S.; Borkowski, M.; Leitner, P. Optimized IoT service placement in the fog. *Serv. Oriented Comput. Appl.* **2017**, *11*, 427–443. [CrossRef]

22. Mokni, M.; Yassa, S.; Hajlaoui, J.E.; Omri, M.N.; Chelouah, R. Multi-objective fuzzy approach to scheduling and offloading workflow tasks in Fog–Cloud computing. *Simul. Model. Pract. Theory* **2023**, *123*, 102687. [CrossRef]
23. Luo, J.; Yin, L.; Hu, J.; Wang, C.; Liu, X.; Fan, X.; Luo, H. Container-based fog computing architecture and energy-balancing scheduling algorithm for energy IoT. *Future Gener. Comput. Syst.* **2019**, *97*, 50–60. [CrossRef]
24. Misirli, J.; Casalicchio, E. An Analysis of Methods and Metrics for Task Scheduling in Fog Computing. *Future Internet* **2023**, *16*, 16. [CrossRef]
25. Verma, R.; Chandra, S. HBI-LB: A Dependable Fault-Tolerant Load Balancing Approach for Fog based Internet-of-Things Environment. *J. Supercomput.* **2022**, *79*, 3731–3749. [CrossRef]
26. Ranjan, H.; Dwivedi, A.K.; Prakasam, P. An optimized architecture and algorithm for resource allocation in D2D aided fog computing. *Peer-to-Peer Netw. Appl.* **2022**, *15*, 1294–1310. [CrossRef]
27. Mekonnen, D.; Megersa, A.; Sharma, R.K.; Sharma, D.P. Designing a Component-Based Throttled Load Balancing Algorithm for Cloud Data Centers. *Math. Probl. Eng.* **2022**, *2022*, 4640443. [CrossRef]
28. Yang, Y.; Zhao, S.; Zhang, W.; Chen, Y.; Luo, X.; Wang, J. DEBTS: Delay Energy Balanced Task Scheduling in Homogeneous Fog Networks. *IEEE Internet Things J.* **2018**, *5*, 2094–2106. [CrossRef]
29. Suleiman, H. A Cost-Aware Framework for QoS-Based and Energy-Efficient Scheduling in Cloud–Fog Computing. *Future Internet* **2022**, *14*, 333. [CrossRef]
30. Zhang, C. Design and application of fog computing and Internet of Things service platform for smart city. *Future Gener. Comput. Syst.* **2020**, *112*, 630–640. [CrossRef]
31. Bozorgchenani, A.; Tarchi, D.; Corazza, G.E. Centralized and Distributed Architectures for Energy and Delay Efficient Fog Network-Based Edge Computing Services. *IEEE Trans. Green Commun. Netw.* **2018**, *3*, 250–263. [CrossRef]
32. Baucas, M.J.; Spachos, P. Using cloud and fog computing for large scale IoT-based urban sound classification. *Simul. Model. Pract. Theory* **2019**, *101*, 102013. [CrossRef]
33. Jiang, Y.-L.; Chen, Y.-S.; Yang, S.-W.; Wu, C.-H. Energy-Efficient Task Offloading for Time-Sensitive Applications in Fog Computing. *IEEE Syst. J.* **2018**, *13*, 2930–2941. [CrossRef]