applied sciences

MDPI

Article

A Novel Fault-Tolerant Approach for Dynamic Redundant Path Selection Service Migration in Vehicular Edge Computing

Jiale Zhao [1], Yong Ma [2,*], Yunni Xia [1,*], Mengxuan Dai [2], Peng Chen [3], Tingyan Long [1], Shiyun Shao [4], Fan Li [5], Yin Li [6] and Feng Zeng [7]

1 School of Computer, Chongqing University, Chongqing 400044, China
2 School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China
3 School of Computer and Software Engineering, Xihua University, Chengdu 610039, China
4 Department of Computer Science and Operations Research, University of Montreal, Montreal, QC H3T 1N8, Canada
5 Key Laboratory of Fundamental Synthetic Vision Graphics and Image Science for National Defense, Sichuan University, Chengdu 610065, China
6 Institute of Software Application Technology, Guangzhou and Chinese Academy of Sciences, Guangzhou 511458, China
7 Discovery Technology (Shen Zhen) Limited, Shenzhen 518126, China
* Correspondence: may@jxnu.edu.cn (Y.M.); pentaxm42@cqu.edu.cn (Y.X.)

Abstract: Vehicular Edge Computing (VEC) provides users with low-latency and highly responsive services by deploying Edge Servers (ESs) close to applications. In practice, vehicles are usually moving rapidly. To ensure the continuity of services, edge service migration technology is in high need, by which an application, infrastructure or any edge-hosted applications or services are not locked into a single vendor and allowed to shift between different edge resource vendors. Nevertheless, due to their complex and dynamic nature, real edge computing environments are error and fault prone and thus the reliability of edge service migrations can be easily compromised if the proactive measures are not taken to counter failures at different levels. In this paper, we propose a novel fault-tolerant approach for Dynamic Redundant Path Selection service migration (DRPS). The DRPS approach consists of path selection algorithm and service migration algorithm. The path selection algorithm is capable of evaluating time-varying failure rates of ESs by leveraging a sliding window-based model and identifying a set of service migration paths. The service migration algorithm incorporates resubmission and replication mechanisms as well and decides edge service migration schemes by choosing multiple redundant migration paths. We also conduct extensive simulations and show that our proposed method outperforms traditional solutions by 17.45%, 13.17%, and 7.22% in terms of ACT, TCR, and AFC, respectively.

Keywords: vehicular edge computing; service migration; fault-tolerant; dynamic redundant path selection; time-varying failure rates

Citation: Zhao, J.; Ma, Y.; Xia, Y.; Dai, M.; Chen, P.; Long, T.; Shao, S.; Li, F.; Li, Y.; Zeng, F. A Novel Fault-Tolerant Approach for Dynamic Redundant Path Selection Service Migration in Vehicular Edge Computing. *Appl. Sci.* **2022**, *12*, 9987. https://doi.org/10.3390/app12199987

Academic Editor: Ming Liu

Received: 12 August 2022
Accepted: 27 September 2022
Published: 4 October 2022

1. Introduction

Mobile Edge Computing (MEC) is of great significance in real-time computing services among various branches [1–5]. Many typical computation-intensive applications, e.g., face identification, interactive game, auto navigation, augmented reality, and remote control aircraft, benefit from the mobile edge computing paradigm and its capability of handling distributed computing with high speed and scale [6–9]. Due to the popularization of new energy technologies, Vehicle Edge Computing (VEC) technology based on intelligent vehicles has received extensive attention [10]. While vehicles always move around, limited coverage of each ES can result in the dramatic drop of Quality of Service (QoS) or even service interruption [11]. To this end, service needs to be transferred from the source ES to the target one (the closest one), i.e., service migration in vehicular edge computing [12].

In practice, edge service migration is in high need, by which an application, infrastructure or any edge-hosted applications or services are allowed to shift between different resource vendors or edge nodes when the original vendors or nodes are non-functional or overloaded [13,14].

As the edge computing systems continue to grow in scale and complexity, it is of critical importance to ensure the stability, availability, and reliability in such systems. However, due to its intrinsic dynamic and decentralized nature, the edge computing infrastructure itself is fault and error prone [15]. The reliability of edge service migration can therefore be easily compromised if the proactive measures are not taken to tackle against the possible failures occurring at varying levels of the edge computing system. To guarantee reliable service migration, and as a countermeasure for faults and failures, the edge service providers have to enforce fault tolerance [16–19].

Among various fault tolerance strategies in this direction, replication and resubmission are two widely used ones that are proved to be highly capable in mobile edge computing [20]. The replication-based fault-tolerance refers to the technique of running multiple redundant replicates for each and every service. Thus, if one part of the system goes wrong, it has other instances that can be placed instead of it to keep it running. Although running redundant replicas generally increases the operational overhead, it effectively prevents failures of components or individual resources from interrupting system-level task flows, thereby improving the efficiency of service execution. The resubmission-based fault-tolerance refers to re-executing a failed task on the same or another processing unit. Although re-executing a task costs extra time, it can improve the system resource utilization and effectively prevent the service interruption caused by task execution failure. Researchers have proposed many fault-tolerant schemes based on replication and resubmission [16,21–26].

This paper focuses on service migration in VEC environments and proposes a novel fault-tolerant approach for dynamic redundant path selection service migration in mobile edge computing, called DRPS, by synthesizing the replication and resubmission-based fault-tolerance techniques.

The main contributions of the paper are as follows:

- It presents the VEC system of a fault-tolerant environment, including the service migration model, and edge server fault model for the first time.
- Propose a path selection algorithm that combines the sliding window model to evaluate the time-varying failure rate of ESs and obtain the high-quality migration path.
- Propose a service migration algorithm. To cope with edge node failure in the VEC environment, this paper uses the advantages of replication and resubmission strategies to ensure the reliability of service migration.
- To verify the performance of the DRPS approach, we also conducted simulation experiments on two real edge computing datasets. The experimental results demonstrate that, compared with the traditional method, our proposed DRPS approach has a better task completion rate and average task completion time.

The remaining chapters of this paper are structured as follows. In Section 2, we review related work and analyze its limitations. The system model and problem description are presented in Section 3. In Section 4, we describe the proposed DRPS approach in detail. We describe experiments based on the real mobile edge computing dataset and analyze the results in Section 5. Finally, Section 6 presents the main conclusions and future work.

2. Related Work

The path selection problem stems from the basic trade off between the cost of service migration (transmission cost and migration cost) and the improvement of users' expectation on QoS that can be achieved after migration. Recently, extensive efforts are paid to enforcing QoS guarantees and fault tolerance in edge service migrations [27]. For instance, Machen et al. [28] proposed a layered migration framework that divided the base package into three layers: base layer, application layer, and instance layer. A copy of

the base layer is stored on each ES, so it is unnecessary to transfer it on every migration. The application layer contains the basic data information required for service operation. The running state of the service is placed in the instance layer. When migrating a service, the application layer is first migrated, then the service is suspended, and the instance layer is transferred to the target ES. Finally, the base layer, application layer, and instance layer are combined in the target ES to reconstruct the running service. Ha et al. [29] performed delta encoding, deduplication, and data compression operations in sequence before service data migration to reduce the amount of data that needs to be transmitted, thereby improving the efficiency of service migration. However, the data will be distorted to a certain extent after being compressed, and the compression time of the data will also affect the QoS of service migration. Zhao et al. [30] proposed an optimization method for user overhead and provider operating costs in a large-scale edge computing environment. Firstly, the network and user movement space are modeled. Then, random global search schemes such as simulated annealing, tabu search, greedy algorithm, and binary search are combined to search for the best path for migration, thereby reducing data migration overhead and saving costs. Qiao et al. [31] proposed a collaborative edge caching scheme to jointly optimize task placement and migration in vehicle edge computing. This scheme formulates the joint optimization problem as a dual-time-scale Markov Decision Process (MDP) according to the relationship between the task migration time scale and the vehicle movement time scale, which minimizes the task migration cost while satisfying the task migration delay constraint. However, when the MDP state space is large, the cost function and transition probability of MDP may change with time, and the processing capacity of ES is limited, and the resulting problems become difficult to solve. Wang et al. [32] approximate the underlying state space by the distance between users and service locations, which is much faster than traditional methods based on standard value or policy iteration and can efficiently compute service migration policies for 2D mobility models. Representing task migration as an online multidimensional integer linear programming problem, Peng et al. [33] proposed a decentralized reactive method that generates online migration decisions by employing a dynamic learning mechanism. Different from [33], Liang et al. [34] first relax the binary constraints of the transfer decision, and transform the problem into a series of convex problems that can be optimally solved. Then, a rounding method for binary decision solution recovery based on the nature of the problem is proposed to maximize the weighted unloading rate for all users while minimizing the migration cost as much as possible.

In recent years, with the rise of the third wave of artificial intelligence technology, many researchers have begun to combine Reinforcement Learning (RL) technology to deal with the problem of service migration. Gao et al. [35] designed a reinforcement learning-based framework for a single-user edge computing service migration system, which first assumed that the user's moving trajectory was fixed and known, and then used RL methods to find the optimal communication path between two edge servers for service migration. For the multi-user random motion scenario, Chen et al. [36] used the container as the target of service migration to minimize the average perceived delay of users and the average system energy, and proposed a Deep Reinforcement Learning algorithm based such a Multi-user Server Migration strategy (DRLMSM). The DRLMSM algorithm uses Software Defined Network (SDN) to arbitrarily segment traffic on any path, and allows multiple containers to be migrated simultaneously through multiple routing paths, enabling quick decision-making in unstable MEC environments. Chen et al. [37] applied containers in industrial Internet edge computing, and proposed an Improved Genetic-Simulated Annealing Algorithm (IGSAA). The algorithm can solve the problems of container deployment failure and slow deployment caused by hardware failure by improving the initialization, crossover, and mutation operations of the genetic algorithm, so that the edge computing system can still provide services when the container deployment fails, and the reliability of the service is improved. Similar to [31], Li et al. [38] also use edge caching technology and proposed a cooperative edge caching strategy based on the energy-

delay balance to solve the problem of service interruption caused by user movement. First, the cache value model is trained by deep learning, and the content with higher potential cache value is selected for caching in the region, then the service migration problem is modeled as MDP, and finally the Deep Q-Network (DQN) algorithm to find the optimal solution of the model. Although deep reinforcement learning can adapt to complex state-space environments, it is difficult for a single super-agent to learn large-scale decentralized policies as the number of participants grows exponentially. Therefore, Yuan et al. [39] proposed a Multi-Agent Deep Reinforcement Learning (MADRL) algorithm, decomposes a single overall agent into multiple simpler agents to reduce the dimensionality of state space and action space, and constructs a deep q-network based on bi-branch convolution to minimize service migration cost and migration time. However, most of the above methods ignore the impact of ESs performance fluctuations in edge computing on service migration.

Long et al. [40] synthesized the Primary-Backup (PB) and DQN models for developing a novel fault-tolerant service composition method. Wang et al. [41] extended the traditional PB fault-tolerant model based on the idea of containers, and reduced the delay by changing the execution time of task copies to achieve reliable task services. An adaptive adjustment mechanism of container resources is also proposed, which can adaptively supplement the resources of ESs with insufficient processing capability during task execution, so as to improve the resource utilization of edge nodes. Tuli et al. [42] proposed a service migration strategy that can proactively predict failures (PreGAN) based on Generative Adversarial Networks (GAN) to achieve active fault tolerance in containerized edge deployments. PreGAN uses Graph Attention Networks (GAT) and Gated Recurrent Unit (GRU) for feature extraction, and a multi-head attention and a prototype prediction decoder to detect and classify faults. PreGAN utilizes a generator model to utilize anomaly class prototypes to output a delta scheduling decision to correct faults and improve QoS. Xu et al. [43] considered time and cost, and the Pareto optimal method was used to select several suitable migration paths for simultaneous migration to achieve the purpose of fault tolerance. Still, the change of ESs failure rate was not considered. To avoid single points of failure and data loss, ES can replicate data to alternate hosts for fault tolerance, but replicating data at the speed that data arrives is inefficient and will consume significant network bandwidth resources. Thus, Wang et al. [44] proposed a fault-tolerant method that adaptively replicates data according to the application's requirements for data loss and latency to meet the required level of data loss tolerance and timeliness. Jhawar et al. [45] proposed to ensure the QoS of the system through a fault-tolerant strategy provided by a third party. However, it is difficult and impractical for ordinary users to choose a suitable third-party service. Different from the above research that adopted a single fault tolerance strategy, Plankensteiner et al. [46] proposed a replication and resubmitted each service in edge computing simultaneously and adjusted the size of the copy by using the impact of service resubmission to make a trade-off between the two fault tolerance strategies. Similarly, Yao et al. [47] proposed a fault-tolerant scheduling algorithm for the workflow based on unbalanced resources by combining two fault-tolerant strategies of replication and resubmission, assuming that the failure rate of ESs is fixed.

It can be observed that, although the current research on service migration has made some important progress, existing studies are still limited in several ways:

- The performance fluctuations of ES are not taken into account. In the VEC environment, the processing capability and operational stability of ES are always affected by natural or human factors, but the existing research has not fully considered this issue.
- The fault tolerance mechanism is single. Each fault tolerance mechanism has its applicable scenarios and shortcomings. Most of the existing researches are based on one fault tolerance mechanism for service migration, which is difficult to apply to other scenarios.

The above limitations could be well avoided by using the DRPS approach proposed in this paper.

3. System Model and Problem Description

In this section, the architecture of the proposed VEC environment for service migration is described. The environment consists of a base station, network router, and edge user (vehicle). Key notations are given in Table 1.

Table 1. Summary of notations.

Notation	Definition
B	The set of base stations
N	The set of network routers
R	The set of tasks
M^{r_i}	The set of Migration paths for task r_i
MT_i	The Migration time of task r_i
b_i	The i-th base station in B
n_i	The i-th network router in N
r_i	The i-th task in R
$m_j^{r_i}$	The j-th Migration path in M^{r_i}
l_i^b	The coordinates of b_i
f_i^b	The signal coverage radius of b_i
e_i^b	The edge server of b_i
c_i^b	The residual computing power of e_i
sw_i^b	The task information performed on the e_i in the recent period
l_i^n	The coordinates of n_i
f_i^n	The failure rate of n_i
a_i^r	The arrival time of r_i
d_i^r	The amount of data transmitted in r_i
td_i^r	The maximum tolerable time of r_i
l_i^r	The coordinates of r_i
p_i^r	The amount of calculation of r_i
t_i^j	The execution time of task r_i on e_j
$t_i^{j'}$	The normalized data of t_i^j
y_i^j	The number of edge servers in $m_j^{r_i}$
bw_{xy}	The bandwidth between node x and node y
$d_i^{r'}$	The calculation result of d_i^r
β	The channel bandwidth of the base station
ρ_i	The signal transmission power of r_i
λ^2	The Gaussian noise power of the device

3.1. System Model

As illustrated in Figure 1, in an VEC environment, ESs are deployed on signal Base Stations (BSs) near users. This paper uses the set $B = \{b_1, b_2, ..., b_m\}$ to represent m BSs deployed in an area, each BS is represented by a triple (l_i^b, f_i^b, e_i^b), where l_i^b represents the geographic location of the BS b_i, f_i^b represents the signal coverage radius of the b_i, $e_i^b = (c_i^b, sw_i^b)$ represents the ES in the b_i, where c_i^b represents the remaining computing power of the e_i^b. sw_i^b means the task information performed on the e_i^b in the recent period. Use the set $N = \{n_1, n_2, ..., n_k\}$ to represent k Network Routers (NRs) deployed in an area; each NR is represented by a two-tuple (l_i^n, f_i^n), where l_i^n represents the geographic location of the NR n_i, f_i^n represents the failure rate of the n_i, the failure rate is assumed to be constant. Use the set $R = \{r_1, r_2, ..., r_n\}$ to represent the services that need to be migrated of Edge Users (EUs). In this model, services are migrated in the form of tasks. Each task is defined by a quintuple $(a_i^r, d_i^r, td_i^r, l_i^r, p_i^r)$, where a_i^r represents the arrival time of task r_i, d_i^r represents the amount of data transmitted, td_i^r represents the maximum tolerable time, and l_i^r represents the user's position; p_i^r represents the amount of computation required to complete task r_i.

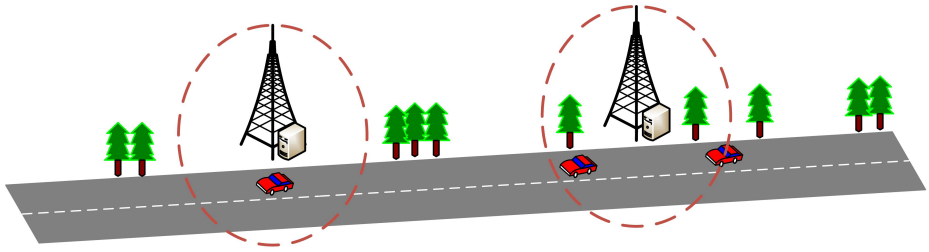


Figure 1. Vehicle edge computing environment.

In the proposed system model, it is assumed that each BS is equipped with only one ES, and the performance of all ES is exactly the same. BS can forward and compute tasks (ES is responsible for the computation), while NR can only forward tasks and does not participate in the computation of tasks. The failure rate of the BS is assumed to change in real time, the failure rate of the NR remains constant all the time, and the EU always communicates with the nearest BS.

This paper abstracts the transmission link between each node (ES and NR) in mobile edge computing network into an undirected graph. It transforms the problem into how to transmit tasks in the undirected graph efficiently. As illustrated in Figure 2, when EU moves from left to right, the possible service migration paths are: ES1-NR1-NR5-ES5, ES1-NR1-NR4-ES5, ES1-NR2-ES3-NR5-ES5, and so on. How to determine the direction of EU motion belongs to the problem of trajectory prediction, which is not the content of this paper. In this paper, we focus on the problem of task fault-tolerant migration when the origin and destination of the EU are known. We set $M^{r_i} = \{m_1^{r_i}, m_2^{r_i}, ..., m_s^{r_i}\}$ is used to represent s migration paths of a task r_i, and each path $m_j^{r_i}$ is composed of a multiple node. The performance of ESs in a real VEC environment is not constant. Still, it changes dynamically with the influence of the time and external factors (e.g., weather, temperature, etc.), which leads to the traditional shortest path-based service migration strategy not always achieving better results. If the failure rate of NR1 in Figure 2 is much greater than that of ES3, the migration effect of selecting the path ES1-NR2-ES3-NR5-ES5 is significantly better than that of the paths ES1-NR1-NR5-ES5 and ES1-NR1-NR4-ES5.

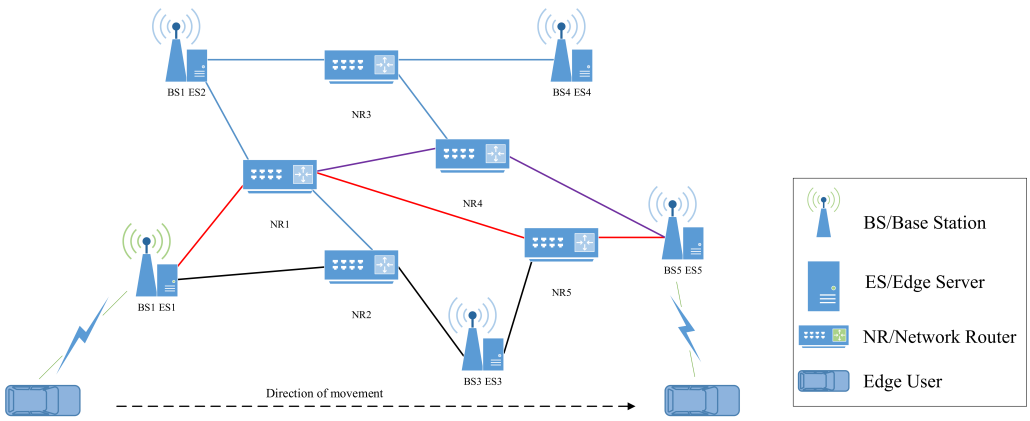


Figure 2. Abstract VEC network structure.

Based on the sliding window mechanism, this paper updates the failure rate of each ESs in real-time to ensure that the transmission path of each task is high-quality. The probability of a failure of task r_i executed on edge server e_j^b is:

$$P_i^j(F \mid t_i^{j'}) = \frac{P_i^j(t_i^{j'} \mid F) P_i^j(F)}{P_i^j(t_i^{j'})} \tag{1}$$

$P_i^j(t_i^{j'} \mid F)$, $P_i^j(F)$, and $P_i^j(t_i^{j'})$ can be calculated according to the data in the sliding window sw_j^b of e_j^b. Where F represents the migration failure, and $t_i^{j'}$ represents the execution time t_i^j of task r_i on e_j^b to normalize data, the normalization process is expressed as follows:

$$t_i^{j'} = \frac{t_i^j - t_{min}}{t_{max} - t_{min}} \tag{2}$$

where t_{max} and t_{min} respectively represent the longest and shortest execution time of tasks in sw_j^b on e_j^b, and the execution time t_i^j is calculated as follows:

$$t_i^j = \frac{p_i^r}{y_i^k c_j^b} \tag{3}$$

where y_i^k represents the number of ES in the migration path $m_k^{r_i}$, p_i^r represents the amount of computation required to complete the task r_i, and c_j^b represents the remaining computation force in e_j^b.

MT_i^k of the total time of service migration of task r_i on path $m_k^{r_i}$ consists of task calculation time Z_i^k and data transmission time D_i^k:

$$MT_i^k = Z_i^k + D_i^k \tag{4}$$

Since NR only forwards data and does not participate in calculation, all the tasks are calculated in ES:

$$Z_i^k = \sum_{x=1}^{y_i^k} t_i^x + FZ_i^k \tag{5}$$

where FZ_i^k represents the redundant calculation time, an optional item, and only needs to be calculated when ES fails.

D_i^k consists of the following four parts:

$$D_i^k = DU_i^k + DT_i^k + DD_i^k + FD_i^k \tag{6}$$

DU_i^k represents the uplink transmission time of d_i^r from user equipment to ES:

$$DU_i^k = \frac{d_i^r}{\beta \log_2(1 + \frac{\rho_i}{\lambda^2})} \tag{7}$$

where β is the channel bandwidth of the BS, ρ_i is the signal transmission power of the mobile device, and λ^2 is the Gaussian noise power of the device.

DT_i^k represents the transit time of d_i^r between ESs and NRs:

$$DT_i^k = \sum_{(x,y) \in m_k^{r_i}} \frac{d_i^r}{bw_{xy}} \tag{8}$$

$(x, y) \in m_k^{r_i}$ represents the record of task r_i from node x to node y, bw_{xy} represents the bandwidth between x and y.

DD_i^k indicates the downlink transmission time for the data calculation result to be returned from the ES to the user equipment:

$$DD_i^k = \frac{d_i^{r'}}{\beta \log_2(1 + \frac{\rho_i}{\lambda^2})} \tag{9}$$

where $d_i^{r'}$ represents the calculation result of task r_i.

FD_i^k represents the redundant calculation time, an optional item, and only needs to be calculated when ES or NR fails. In the actual scenario, EU must upload some instruction information in addition to computing tasks, but the amount of data of this extra information is small, so it is not considered in our model.

3.2. Problem Description

Based on the above system model, this paper aims to solve the following problems: given an area where multiple ESs and NRs are deployed, the task will pass through h nodes in the migration path p in this area, among which there are m ESs and how to choose the appropriate migration path to reduce the failure rate of service r_i migration to minimize migration time, the problem can be described as:

$$Min : \prod_{x=1}^{m} P_i^x(F \mid t_i^{x'}) \prod_{y=1}^{h-m} f_y^n \tag{10}$$

$$Min : MT_i^p \tag{11}$$

$$s.t : g(r_i) \in M \tag{12}$$

$$h(r_i) \in M \tag{13}$$

$$MT_i^p \leq td_i^r \tag{14}$$

$$0 \leq P_i^x(F \mid t_i^{x'}) < 1 \tag{15}$$

$$0 \leq f_y^n < 1 \tag{16}$$

Functions $g(r_i)$ and $h(r_i)$ represent task r_i arriving at source ES and migrating to target ES, respectively, M represents the resource pool composed of m ESs, and MT_i^p represents the migration time of the task. As shown in (10) and (11), the problem to be solved in this paper is to minimize the failure rate and migration time during task migration, where (12) and (13) limit the data transmission between the user and the ES to the base station's within the signal coverage, (14) indicates that the task should be migrated within the maximum tolerable time of the user, (15) and (16) indicate that the failure rates of ES and NR are within a reasonable range.

4. DRPS: Fault-Tolerant Approach for Dynamic Redundant Path Selection Service Migration

Based on the system model proposed in Section 3, for the service migration problem in the VEC environment, this paper considers the performance fluctuation of ESs and proposes a fault-tolerant approach called DRPS, which consists of path selection algorithm and service migration algorithm. This section describes the method in detail.

4.1. Path Selection

In this paper, the service migration whose execution time does not exceed a certain threshold is called a service migration success. Mobile users will not feel the change of QoS obviously at this time. If the service migration time exceeds a certain threshold, the service migration is considered a failure because mobile users can feel the decrease of QoS obviously at this time. The traditional service migration [43] mainly studies the migration of task data and does not consider the calculation of tasks during the migration process. However, with the increase in the complexity of mobile applications, simple data migration is gradually difficult to meet user needs. Therefore, this paper considers the case where services are migrated and calculated simultaneously.

As shown in Algorithm 1, this paper uses the Dijkstra algorithm to find the migration path between the source ES and the target ES. When looking for the migration path of task r_i, the weight of the ESs node in graph G is the failure rate of r_i when it is executed on e_j^b, i. e., $P_i^j(F \mid t_i^{j'})$. First, initialize the migration path set M^{r_i} and weight graph G (lines 1–2), then find the path m^{r_i} with the lowest failure rate according to the Dijkstra algorithm,

add m^{r_i} to the set M^{r_i}, delete the data link in the path m^{r_i}, and cyclically query the all paths (lines 3–9), and finally determine whether the service is successfully migrated (lines 10–21). The task migration time MT_i is the shortest migration time of task r_i in all feasible paths, which will be described in detail later. Algorithm 1 is based on Dijkstra, so the time complexity is $O(kn^2)$, where k is the number of all feasible migration paths and n is the number of all nodes in G.

Algorithm 1 Path Selection Algorithm

Input: Task r_i; source edge server B_s; target edge server B_t.
Output: T or F, where T indicates that the task migration succeeded, and F indicates that the task migration fail.

1: Initialize the set of migration paths $M^{r_i} \leftarrow \{\}$.
2: Abstracts the transmission link between B_s and B_t as a weighted graph G, and the failure rate is the weight.
3: **while** True **do**
4: apply Dijkstra algorithm to the weighted graph G to find the shortest path m^{r_i} from B_s to B_t
5: **if** m^{r_i} not exists **then**
6: break
7: **else**
8: $M^{r_i} \leftarrow M^{r_i} \cup \{m^{r_i}\}$
9: delete all nodes in path m^{r_i}
10: **end if**
11: **end while**
12: **if** $M^{r_i} == \varnothing$ **then**
13: return F
14: **else**
15: compute the task migration time MT_i
16: **if** $MT_i > td_i^r$ **then**
17: return F
18: **else**
19: return T
20: **end if**
21: **end if**

It should be noted that to ensure the success rate of service migration, the path selection algorithm finds all feasible paths in G for migration, which is not suitable for all cases. If all paths are found for migration in a dense network, the cost of migration will be immeasurable. Still, with a slight modification of the path selection algorithm, it can be applied to a broader range of situations; e.g., limiting the number of migration paths or limiting the number of nodes in the migration path, etc., can be a good trade-off for the problem of excessive migration costs.

4.2. Service Migration

As shown in Algorithm 2, the service migration algorithm first copies the task into many copies and migrates in all feasible paths simultaneously. There is no need to worry about the problem that ESs repeatedly calculates multiple identical tasks during the migration process. First of all, when searching for the migration path in Algorithm 1, all duplicate data links have been deleted (line 9), so there will not be the same data links in the migration path. Secondly, suppose two replicated tasks arrive at the same ES simultaneously (through different links, although the probability of this happening is very low). In that case, the ES will only calculate any one of the tasks and copy the calculation result, as both are sent to the next different node, respectively. Because the two tasks are the same, the calculation results are also the same, and the cost of copying the calculation results is meager, which does not affect the final result. The significance of copying tasks to all migration paths is to reduce the migration failure rate and speed up the migration. In service migration

phase, the redundancy idea commonly used in fault tolerance mechanism is combined. If the BS or NR in a certain path fails, other migration paths without failure can ensure the successful migration of this task. The downside is obvious - it consumes more resources. However, the user experience can be improved by making a trade-off between the number of migration paths and the resource consumption.

Algorithm 2 Service Migration Algorithm

Input: Task r_i; s feasible migration paths $\{m_1^{r_i}, m_2^{r_i}, ..., m_s^{r_i}\}$.
Output: The task final migration time MT_i.

1: Copy the tasks r_i to be migrated into s copies;
2: **do in parallel**
3: compute the migration time mt_i^1 of task r_i on path $m_1^{r_i}$. If the task is successfully migrated first on this path, it will send a termination migration signal to other paths. At this time $MT_i = mt_i^1$;
4: compute the migration time mt_i^2 of task r_i on path $m_2^{r_i}$. If the task is successfully migrated first on this path, it will send a termination migration signal to other paths. At this time $MT_i = mt_i^2$;
5: ...
6: compute the migration time mt_i^s of task r_i on path $m_s^{r_i}$. If the task is successfully migrated first on this path, it will send a termination migration signal to other paths. At this time $MT_i = mt_i^s$;
7: return MT_i

During task migration, if the ESs fails, the task will be rolled back to the previous node and resend to the ES for processing. When the task on a path in M^{r_i} reaches the target ES successfully, a termination instruction will be sent to terminate the service migration on other paths to save costs. This is easy to implement because each task knows all migration paths. The transfer time of the termination instruction is negligible, so it can quickly terminate the migration of tasks on other paths. The service migration algorithm minimizes the task migration time based on the above resubmission and replication strategy.

Algorithm 3 describes the migration process of tasks on each migration path and the calculation method of migration time. The migration time consists of computation time and transmission time. When the ES or NR fails and the migration fails, the task is rolled back to the previous node for retransmission. If the source edge server fails, it means that the task cannot be migrated at this time, that is, the migration fails. ES and NR will retain the task information (task size, execution time, failure or failure duration, etc.) of tasks that have been migrated on this device in the recent period to provide necessary information for the next round of task migration. It should be noted that, as shown in lines 7–17 and 20–21, if a task fails to be rolled back, its final migration time will need to be added with the rollback calculation time, which is reflected in the form of the redundant calculation time in this model. Service migration is computed in parallel, so the time complexity is $O(n)$, where n represents the number of nodes in a single path.

Algorithm 3 Service Migration Time

Input: Task r_i; The migration path $m_s^{r_i}$.
Output: The migration time mt_i^s of the task r_i on path $m_s^{r_i}$.

Initialize the task transfer time $D_i^{m_s^{r_i}} = 0$
Initialize the task execution time $Z_i^{m_s^{r_i}} = 0$
for all $n \in m_s^{r_i}$ **do**
  if n is the edge server **then**
    $D_i^{m_s^{r_i}} = D_i^{m_s^{r_i}} + D_i^n \leftarrow$ compute the task transfer time of r_i on n according to (6)
    $Z_i^{m_s^{r_i}} = Z_i^{m_s^{r_i}} + Z_i^n \leftarrow$ compute the task execution time of r_i on n according to (5)
    if n fail **then**
      if n is source edge server **then**
        $D_i^{m_s^{r_i}} = \infty$
        break
      else
        add the information of task r_i to sw_n^b
        roll back to the previous node
      end if
    else
      add the information of task r_i to sw_n^b
    end if
  else
    $D_i^{m_s^{r_i}} = D_i^{m_s^{r_i}} + D_i^n \leftarrow$ compute the task transfer time of r_i on n according to (6)
    if n fail **then**
      roll back to the previous node
    end if
  end if
end for
return $mt_i^s = D_i^{m_s^{r_i}} + Z_i^{m_s^{r_i}}$

5. Performance Evaluation

To verify the performance of the DRPS approach, this section uses two real mobile edge computing datasets for simulation experiments. Two real datasets and experimental settings are first introduced, followed by comparison algorithms and evaluation metrics. Finally, the experimental results are summarized and analyzed.

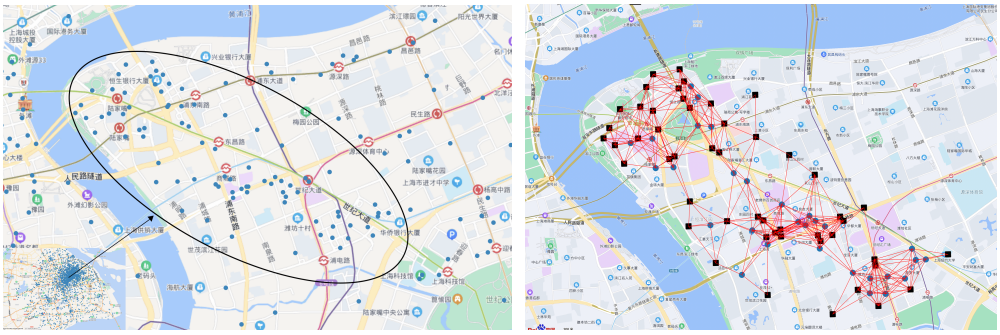
5.1. Experiment Setting

The simulation experiment is based on the real edge environment data sets Telecom (http://sguangwang.com/TelecomDataset.html, accessed on 3 August 2022) [48–50] and Taxi (https://cse.hkust.edu.hk/scrg/, accessed on 3 August 2022) [51]. The Telecom data set records the geographic location information of 3233 base stations in downtown Shanghai, and the Taxi data set records the driving track information of 4316 taxis in downtown Shanghai. To compare the performance of different algorithms objectively, this section treats the two datasets as follows.

(1) As illustrated in Figure 3a, 80 base stations in Pudong Central Business District are selected in the Telecom data set for simulation experiment, and 30 of them are randomly chosen as network routers, which are only responsible for forwarding data without calculation.The communication range of each base station and network router is set as 250 meters, and the initial failure rate is evenly distributed between 0.1 and 0.15. All base stations are equipped with the same ES specifications and support parallel computing. The processed Telecom dataset is illustrated in Figure 3b, with circles representing network routers, rectangles representing base stations, and red connection lines representing data links.

(2) In the Taxi dataset, the driving trajectories of 100, 300, 500, 700, and 900 taxis in the Pudong Central Business District in the same period were selected as the movement trajectories of edge users. The taxi drives at a constant speed. The base station closest to the initial position of the taxi is set as the source base station, and the base station closest to the taxi position after the maximum tolerable time is set as the target base station. Each taxi only needs to migrate one task.

All simulation experiments were carried out on a computer with an 3.2GHz AMD Ryzen 5800H processor and 16GB of RAM. All experimental codes were implemented on in the PyCharm software using Python3.9.



(**a**) The base station distribution (**b**) Abstract base station network structure

Figure 3. Pudong Central Business District (The blue dots in (**a**) represent base stations; Black squares in (**b**) indicate base stations, blue dots indicate network routers, and red lines indicate that devices can communicate with each other).

5.2. Baseline Algorithms and Metrics

The proposed algorithm and four comparison algorithms are as follows.

(1) DRPS: It is the algorithm proposed in this paper. A novel fault-tolerant approach for redundant-path-enabled service migration in mobile edge computing.
(2) DRPS/F: It is a simplified version of the DRPS approach, where the failure rate of ESs is constant.
(3) Greedy-SP: It is a traditional greedy algorithm, which first finds the current closest path and performs service migration based on the resubmission strategy.
(4) Greedy-CP: It is a traditional greedy algorithm, which first find ES with the most remaining computing force within the current coverage range and performs service migration based on the resubmission strategy.
(5) PLP [43]: It is based on weak Pareto optimization and selects multiple paths for service migration.

The migration paths of the DRPS approach and the other four comparison algorithms are illustrated in Figure 4. The movement paths of edge users (vehicles) are represented by red dotted lines, and the migration paths of services are represented by solid lines. As can be observed:

- As illustrated in Figure 4a, for independent tasks, the migration paths of DRPS and DRPS/F approach are the same, because both algorithms choose the migration path according to the failure rate of ES. However, for the task flow that is continuously reached within a period of time, because the DRPS approach adjusts the failure rate of ES dynamically, and the failure rate of the DRPS approach will be lower than that of DRPS/F, which has also been confirmed in subsequent experiments.
- As illustrated in Figure 4b, PLP algorithm always selects the path with the least time and consumption for migration, but it may increase migration time and energy consumption due to the ES failure.

- As illustrated in Figure 4c, the Greedy-SP algorithm selects the closest ES or NR for service migration, but it is easy to fall into the local high-quality solution and cannot find the high-quality migration path.
- As illustrated in Figure 4d, similar to Greedy-SP, Greedy-CP is also based on greedy thinking, but it always prefers the ES with the highest remaining computing power within the scope for service migration.

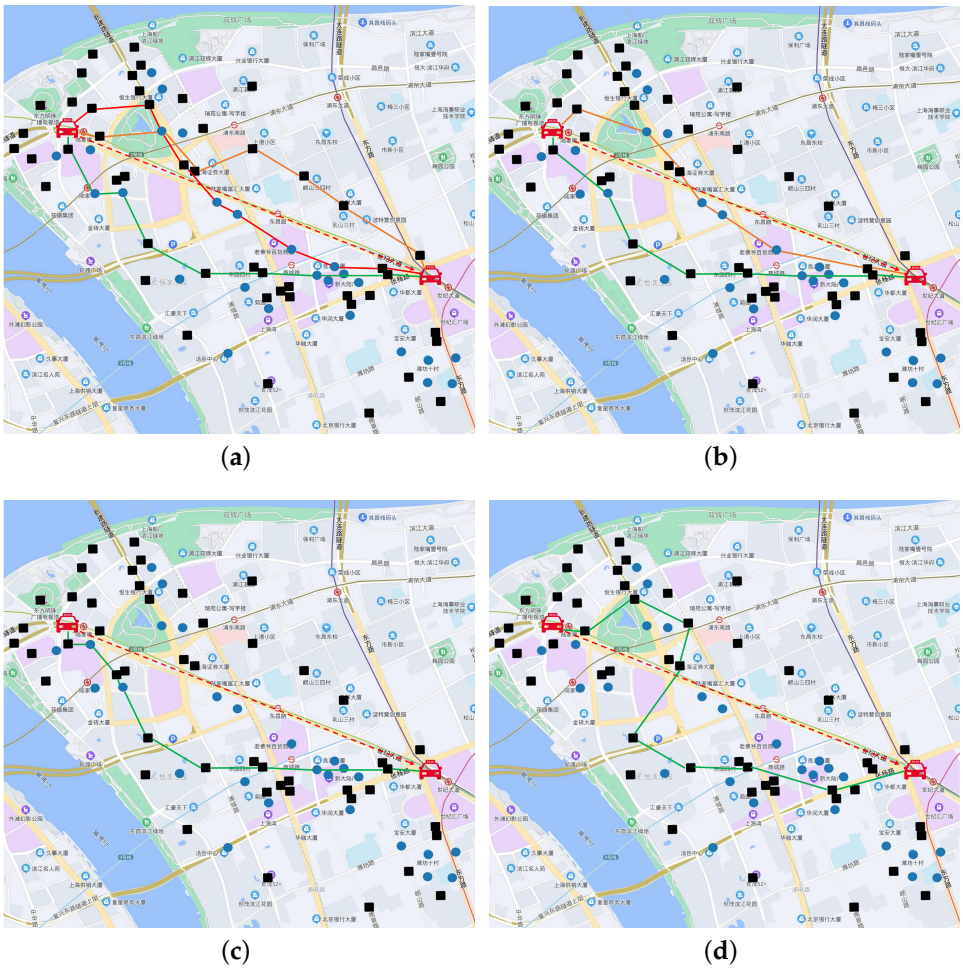


Figure 4. Service migration path in VEC environment (black squares indicate base stations, blue dots indicate network routers, dotted lines indicate vehicle movement directions, and solid lines indicate task migration paths.) (**a**) DRPS&DRPS/F; (**b**) PLP; (**c**) Greedy-SP; (**d**) Greedy-CP.

This section uses the following three indicators to evaluate the algorithm's performance.

(1) Average Completion Time (ACT): Time to migrate services from source ES to target ES, including data transfer time and task computation time.
(2) Task Completion Rate (TCR): The success rate of service migration. If the service is migrated to the targeted ES within the user's maximum tolerable time, the service is successfully migrated.
(3) Average Failure Count (AFC): The number of service failures during migration. The more failures, the higher the cost of the service migration.

It should be noted that because DRPS and DRPS/F choose multiple paths for service migration simultaneously, DRPS and DRPS/F are subdivided into DRPS/OP, DRPS/AP, DRPS/F/OP, and DRPS/F/AP in the experiment to evaluate the average failed times of service migration. DRPS/AP and DRPS/F/AP indicate the average number of migration failures in all migration paths, and DRPS/OP and DRPS/F/OP indicate the average number of migration failures in successful migration paths.

5.3. Experimental Results

As illustrated in Figure 5, the average completion time of the DRPS approach is the lowest because the DRPS approach performs task migration on multiple paths simultaneously. When the task migration on one of the paths is successful, the tasks on the other paths will be discarded to reduce the cost. The migration time of the DRPS/F approach is the lowest after the DRPS approach because DRPS/F does not take into account the performance fluctuations of ES, resulting in a higher failure rate of task migration than the DRPS approach takes more time. The migration times of Greedy-SP, Greedy-CP, and PLP approach are similar and significantly higher than DRPS and DRPS/F approach. Because the three do not consider the performance fluctuations of ES in the process of task migration, the PLP algorithm is jointly optimized with Pareto, so the migration time is slightly lower than the other two algorithms.

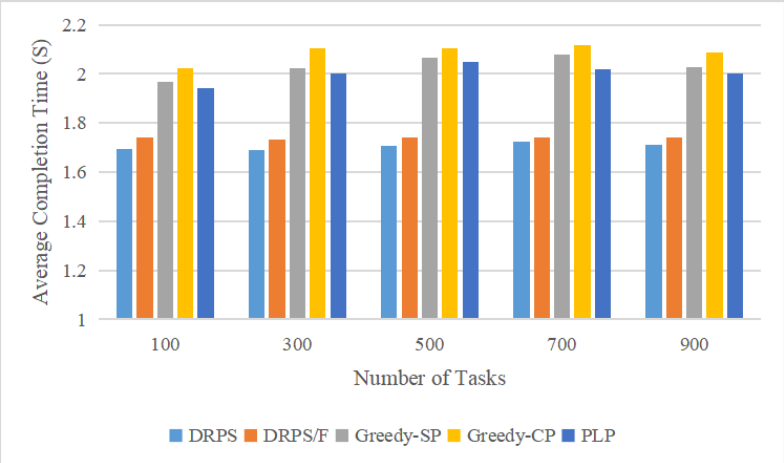


Figure 5. Results of the ACT.

Task completion rate is an important criterion to measure the migration algorithm. A too long task migration time will affect QoS, but task migration failure will have serious consequences. As illustrated in Figure 6, the TCR of the DRPS approach and the DRPS/F approach is significantly higher than that of the other three algorithms. The reason is similar to that of the ACT. The DRPS approach dynamically updates the failure rate of ES in real-time, and the failure rate of each migration path is always the lowest. This ensures that the TCR of the DRPS approach is always better than the comparison algorithm.

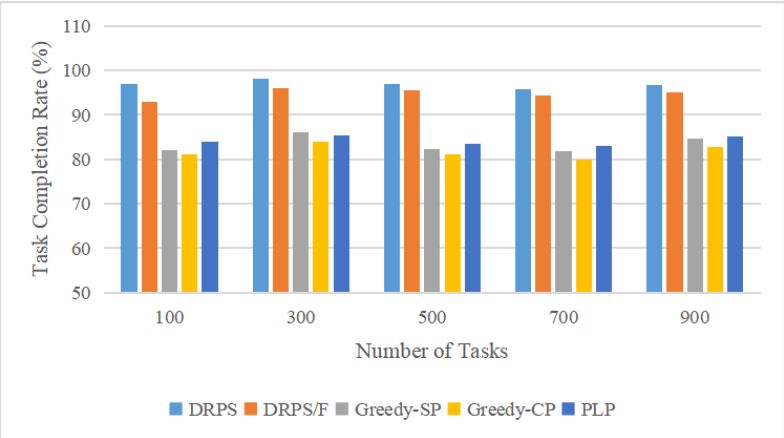


Figure 6. Results of the TCR.

The AFC can reflect the overhead of the algorithm. The higher the AFC, the more tasks that need to be retransmitted during the migration process, and the greater the natural

overhead. In order to measure the performance of the algorithm more accurately, we further divide the DRPS approach and the DRPS/F approach into DRPS/OP, DRPS/AP, DRPS/F/OP, and DRPS/F/AP. As illustrated in Figure 7, DRPS/OP and DRPS/F/OP represent the AFC in the successful migration path. It can be observed that the performance of DRPS/OP is better than other algorithms in five rounds of testing. DRPS/AP and DRPS/F/AP represent the AFCs in all migration paths (including unfinished migration paths), and it can be observed that their performance has decreased. Still, this performance loss is accepted to ensure service migration's reliability.

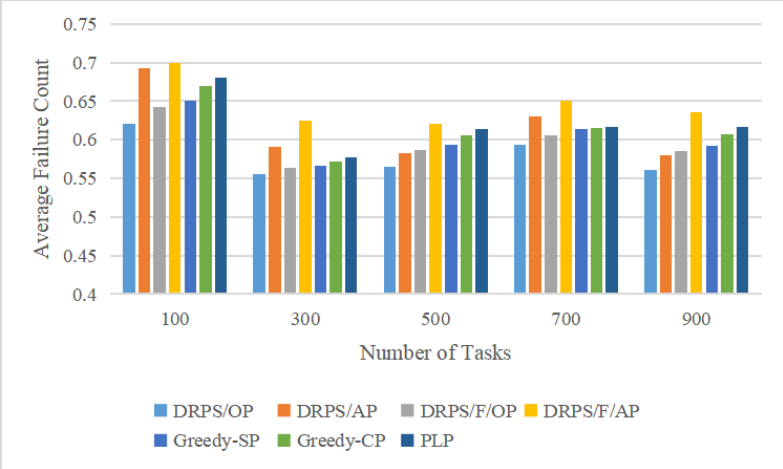


Figure 7. Results of the AFC.

As can be observed from Figures 5 and 6, with the increase in task volume, the performance of DRPS approach remains stable without obvious fluctuation, indicating that the DRPS approach has good robustness in ACT and TCR. Although the performance of DRPS approach on AFC fluctuates with the increase in data volume, it is also within the acceptable range.

Overall, as illustrated in Table 2, compared with the DRPS/F approach, the performance of ACT and TCR of the DRPS approach is improved by 2.03% and 2.13%, respectively; compared with Greedy-SP, the performance of ACT, TCR, and AFC is improved by 19.17%, 14.02%, and 4.18%, respectively; compared with Greedy-CP, the performance of ACT, TCR, and AFC is improved by 22.41%, 15.61%, and 6.01%, respectively; compared with the PLP algorithm, the performance of ACT, TCR, and AFC was enhanced by 17.45%, 13.17%, and 7.22%, respectively; compared with the DRPS/F/OP and DRPS/F/OP approach, the performance AFC was enhanced by 2.97% and 11.6%, respectively (compared with the DRPS/OP). This is because the DRPS approach selects a set of paths for service migration according to the real-time error rate of ESs, and combines replication and resubmission strategies during the migration process to ensure that the service always has the least migration time, compared with traditional PLP and Greedy, etc. The ACT, TCR, and AFC performance of the DRPS approach has been significantly improved.

Table 2. Compared with baseline algorithms, the performance improvement of DRPS on three metrics.

	DRPS/F	**Greedy-SP**	**Greedy-CP**	**PLP**	**DRPS/F/OP**	**DRPS/F/AP**
ACT	2.03%	19.17%	22.41%	17.45%	/	/
TCR	2.13%	14.02%	15.61%	13.17%	/	/
AFC	/	4.18%	6.01%	7.22%	2.97%	11.6%

6. Conclusions

This paper proposes a fault-tolerant service migration approach (DRPS) for service migration in the VEC environment. Compared with the traditional service migration method, DRPS firstly uses the sliding window mechanism to predict the failure rate of ESs in VEC, then calculates a set of high-quality migration paths according to the type of migration task, and, finally, combines the replication and resubmission strategies in the process of service migration to ensure the migration efficiency. To verify the performance of DRPS, we conduct simulation experiments based on two real edge computing datasets. The experimental results show that compared with the traditional service migration method, DRPS can significantly improve the task completion rate and the average task completion time.

The continuous development of artificial intelligence technology and data collaborative processing has become a general trend to apply artificial intelligence technology in VEC. In the next step, we plan to use artificial intelligence technology (e.g., deep learning, reinforcement learning, etc.) with strong robustness and good generalization to solve problems, e.g., load balancing scheduling and ES performance prediction in VEC to further reduce transmission costs to improve the system QoS.

Author Contributions: Conceptualization, J.Z. and Y.X.; Data curation, J.Z., Y.M. and Y.X.; Formal analysis, J.Z., Y.M., Y.X. and M.D.; Software, J.Z.; Writing—original draft, J.Z.; Supervision, Y.M.; Funding acquisition, Y.X.; Validation, M.D.; Methodology, P.C. and S.S.; Investigation, P. C.; Resources and Project administration, T.L.; Visualization, S.S.; Writing—review & editing, F.L., Y.L. and F.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Postgraduate Scientific Research and Innovation Foundation of Chongqing under Grant No. CYB22064; This work is supported by the National Science Foundations under Grant No 62162036.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

1. Song, C.; Liu, M.; Cao, J.N.; Zheng, Y.; Gong, H.G.; Chen, G.H. Maximizing network lifetime based on transmission range adjustment in wireless sensor networks. *Comput. Commun.* **2009**, *11*, 1316–1325. [CrossRef]
2. Xu, F.L.; Guo, S.; Jeong, J.; Yu, G.; Cao, Q.; Liu, M.; He, T. Utilizing shared vehicle trajectories for data forwarding in vehicular networks. In Proceedings of the 2011 IEEE INFOCOM, Shanghai, China, 10–15 April 2011; pp. 441–445.
3. Liu, N.B.; Liu, M.; Wei, L.; Chen, G.H.; Cao, J.N. PVA in VANETs: Stopped cars are not silent. In Proceedings of the 2011 IEEE INFOCOM, Shanghai, China, 10–15 April 2011; pp. 431–435.
4. Liu, M.; Gong, H.G.; Wen, Y.G.; Chen, G.H.; Cao, J.N. The last minute: Efficient data evacuation strategy for sensor networks in post-disaster applications. In Proceedings of the 2011 IEEE INFOCOM, Shanghai, China, 10–15 April 2011; pp. 291–295.
5. Liu, N.B.; Liu, M.; Chen, G.H.; Cao, J.N. The sharing at roadside: Vehicular content distribution using parked vehicles. In Proceedings of the IEEE INFOCOM, Orlando, FL, USA, 25–30 March 2012; pp. 2641–2645.
6. Shi, W.S.; Zhang, X.Z.; Wang, Y.F.; Zhang, Q.Y. Edge computing: State-of-the-art and future directions. *J. Comput. Res. Dev.* **2019**, *56*, 69.
7. Hu, Y.C.; Patel, M.; Sabella, D.; Sprecher, N.; Young, V. Mobile edge computing—A key technology towards 5G. *ETSI White Paper* **2015**, *11*, 1–16.
8. Deng, S.G.; Huang, L.T.; Wu, H.Y.; Tan, W.; Taheri, J.; Zomaya, A.Y.; Wu, Z.H. Toward Mobile Service Computing: Opportunities and Challenges. *IEEE Cloud Comput.* **2016**, *3*, 32–41. [CrossRef]
9. Chung, J.M.; Park, Y.S.; Park, J.H.; Cho, H.J. Adaptive cloud offloading of augmented reality applications on smart devices for minimum energy consumption. *KSII Trans. Internet Inf. Syst. (TIIS)* **2015**, *8*, 3090–3102.
10. Xiong, K.; Leng, S.; Hu, J.; Chen, X.S.; Yang, K. Smart network slicing for vehicular fog-RANs. *IEEE Trans. Veh. Technol.* **2019**, *4*, 3075–3085. [CrossRef]
11. Wang, S.Q.; Urgaonkar, R.; He, T.; Chan, K.; Zafer, M.; Leung, K.K. Dynamic service placement for mobile micro-clouds with predicted future costs. In Proceedings of the 2015 IEEE International Conference on Communications (ICC), London, UK, 8–12 June 2015.

12. Kekki, S.; Featherstone, W.; Fang, Y.G.; Kuure, P.; Li, A.; Ranjan, A.; Purkayastha, D.; Jiangping, F.; Frydman, D.; Verin, G.; et al. MEC in 5G networks. *ETSI White Paper* **2018**, *28*, 1–28.
13. Satyanarayanan, M.; Bahl, P.; Caceres, R.; Davies, N. The Case for VM-based Cloudlets in Mobile Computing. *IEEE Pervasive Comput.* **2011**, *8*, 14–23. [CrossRef]
14. Khayyat, M.; Elgendy, I.A.; Muthanna, A.; Alshahrani, A.S.; Alharbi, S.; Koucheryavy, A. Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks. *IEEE Access* **2020**, *1*, 137052–137062. [CrossRef]
15. Pezoa, J.E.; Dhakal, S.; Hayat, M.M. Maximizing service reliability in distributed computing systems with random node failures: Theory and implementation. *IEEE Trans. Parallel Distrib. Syst.* **2010**, *21*, 1531–1544. [CrossRef]
16. Chen, C.A.; Won, M.; Xie, G.G. Energy-efficient fault-tolerant data storage & processing in dynamic networks. In Proceedings of the Fourteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing, Bangalore, India, 29 July–1 August 2013; pp. 281–286.
17. Chen, C.A.; Stoleru, R.; Xie, G.G. .Energy-efficient and fault-tolerant mobile cloud storage. In Proceedings of the 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), Pisa, Italy, 3–5 October 2016; pp. 51–57.
18. Satria, D.; Park, D.; Jo, M. Recovery for overloaded mobile edge computing. *Future Gener. Comput. Syst.* **2017**, *1*, 138–147. [CrossRef]
19. Long, T.Y.; Ma, Y.; Xia, Y.N.; Xiao, X.; Peng, Q.L.; Zhao, J.L. A Mobility-Aware and Fault-Tolerant Service Offloading Method in Mobile Edge Computing. In Proceedings of the 2022 IEEE International Conference on Web Services (ICWS), Barcelona, Spain, 10–16 July 2022; pp. 67–72.
20. Poola, D.; Ramamohanarao, K.; Buyya, R. Enhancing reliability of workflow execution using task replication and spot instances. *ACM Trans. Auton. Adapt. Syst. (TAAS)* **2016**, *10*, 1–21. [CrossRef]
21. Chen, W.; Lee, Y.C.; Fekete, A.; Zomaya, A.Y. Adaptive multiple-workflow scheduling with task rearrangement. *J. Supercomput.* **2015**, *71*, 1297–1317. [CrossRef]
22. Olteanu, A.; Pop, F.; Dobre, C.; Cristea, V. A dynamic rescheduling algorithm for resource management in large scale dependable distributed systems. *Comput. Math. Appl.* **2012**, *63*, 1409–1423. [CrossRef]
23. Cao, Y.; Ro, C.W.; Yin, J.W. Scheduling analysis of failure-aware VM in cloud system. *Int. J. Control Autom.* **2014**, *7*, 243–250. [CrossRef]
24. Jing, W.; Liu, Y. Multiple DAGs reliability model and fault-tolerant scheduling algorithm in cloud computing system. *Comput. Model. New Technol.* **2014**, *18*, 22–30.
25. Jayadivya, S.K.; Nirmala, J.S.; Bhanu, M.S.S. Fault tolerant workflow scheduling based on replication and resubmission of tasks in Cloud Computing. *Int. J. Comput. Sci. Eng.* **2012**, *4*, 996.
26. Patra, P.K.; Singh, H.; Singh, R.; Das, S.; Dey, N.; Victoria, A.D.C. Replication and resubmission based adaptive decision for fault tolerance in real time cloud computing: A new approach. *Int. J. Serv. Sci. Manag. Eng. Technol. (IJSSMET)* **2016**, *7*, 46–60. [CrossRef]
27. Plachy, J.; Becvar, Z.; Mach, P. Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network. *Comput. Netw.* **2016**, *1*, 357–370. [CrossRef]
28. Machen, A.; Wang, S.Q.; Leung, K.K.; Ko, B.J.; Salonidis, T. Migrating running applications across mobile edge clouds: Poster. In Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, New York City, NY, USA, 3–7 October 2016; pp. 435–436.
29. Ha, K.; Abe, Y.; Chen, Z.; Hu, W.L.; Amos, B.; Pillai, P.; Satyanarayanan, M. *Adaptive VM handoff across cloudlets*; Technical Report CMU-CS-15-113; CMU School of Computer Science: Pittsburgh, PA, USA, 2015.
30. Zhao, F.; Zeng, X. Optimization of user and operator cost for large-scale transit network. *J. Trans. Eng.* **2007**, *133*, 240–251. [CrossRef]
31. Qiao, G.H.; Leng, S.P.; Maharjan, S.; Zhang, Y.; Ansari, N. Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks. *IEEE Internet Things J.* **2019**, *7*, 247–257. [CrossRef]
32. Wang, S.Q.; Urgaonkar, R.; Zafer, M.; He, T.; Chan, K.; Leung, K.K. Dynamic service migration in mobile edge computing based on Markov decision process. *IEEE/ACM Trans. Netw.* **2019**, *3*, 1272–1288. [CrossRef]
33. Peng, Q.L.; Xia, Y.N.; Wang, Y.; Wu, C.R.; Luo, X.; Lee, J. A decentralized reactive approach to online task offloading in mobile edge computing environments. In Proceedings of the International Conference on Service-Oriented Computing, Dubai, United Arab Emirates, 14–17 December 2020; pp. 232–247.
34. Liang, Z.Z.; Liu, Y.; Lok, T.M.; Huang, K.B. Multi-cell mobile edge computing: Joint service migration and resource allocation. *IEEE Trans. Wirel. Commun.* **2021**, *9*, 5898–5912. [CrossRef]
35. Gao, Z.P.; Jiao, Q.D.; Xiao, K.L.; Wang, Q.; Mo, Z.J.; Yang, Y. Deep reinforcement learning based service migration strategy for edge computing. In Proceedings of the 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), San Francisco, CA, USA, 4–9 April 2019; pp. 116–1165.
36. Chen, W.; Chen, Y.H.; Wu, J.X.; Tang, Z.B. A multi-user service migration scheme based on deep reinforcement learning and SDN in mobile edge computing. *Phys. Commun.* **2017**, *1*, 101397. [CrossRef]
37. Chen, Y.P.; He, S.S.; Jin, X.M.; Wang, Z.M.; Wang, F.W.; Chen, L. Resource utilization and cost optimization oriented container placement for edge computing in industrial internet. *J. Supercomput.* **2022**, 1–29. doi: 10.1007/s11227-022-04801-z. [CrossRef]

38. Li, C.L.; Zhang, Y.; Gao, X.; Luo, Y.L. Energy-latency tradeoffs for edge caching and dynamic service migration based on DQN in mobile edge computing. *J. Parallel Distrib. Comput.* **2022**, *1*, 15–31. [CrossRef]
39. Yuan, Q.; Li, J.L.; Zhou, H.B.; Lin, T.; Luo, G.Y.; Shen, X.M. A joint service migration and mobility optimization approach for vehicular edge computing. *IEEE Trans. Veh. Technol.* **2020**, *8*, 9041–9052. [CrossRef]
40. Long, T.Y.; Chen, P.; Xia, Y.N.; Jiang, N.; Wang, X.; Long, M. A novel fault-tolerant approach to web service composition upon the Edge Computing Environment. In Proceedings of the International Conference on Web Services, Chicago, IL, USA, 5–10 September 2021; pp. 15–31.
41. Wang, R.F.; Chen, N.J.; Yao, X.Y.; Hu, L.Q. Fasdq: Fault-tolerant adaptive scheduling with dynamic qos-awareness in edge containers for delay-sensitive tasks. *Sensors* **2021**, *9*, 2973. [CrossRef]
42. Tuli, S.; Casale, G.; Jennings, N.R. PreGAN: Preemptive Migration Prediction Network for Proactive Fault-Tolerant Edge Computing. In Proceedings of the IEEE INFOCOM 2022-IEEE Conference on Computer Communications, London, UK, 2–5 May 2022; pp. 670–679.
43. Xu, J.L.; Ma, X.; Zhou, A.; Duan, Q.; Wang, S.G. Path selection for seamless service migration in vehicular edge computing. *IEEE Internet Things J.* **2020**, *7*, 9040–9049. [CrossRef]
44. Wang, C.; Gill, C.; Lu, C.Y. Adaptive data replication in real-time reliable edge computing for Internet of Things. In Proceedings of the 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI), Sydney, Australia, 21–24 April 2020; pp. 128–134.
45. Jhawar, R.; Piuri, V.; Santambrogio, M. Fault tolerance management in cloud computing: A system-level perspective. *IEEE Syst. J.* **2012**, *7*, 288–297. [CrossRef]
46. Plankensteiner, K.; Prodan, R. Meeting soft deadlines in scientific workflows using resubmission impact. *IEEE Trans. Parallel Distrib. Syst.* **2011**, *23*, 890–901. [CrossRef]
47. Yao, G.; Ding, Y.; Hao, K. Using imbalance characteristic for fault-tolerant workflow scheduling in cloud systems. *IEEE Trans. Parallel Distrib. Syst.* **2017**, *28*, 3671–3683. [CrossRef]
48. Li, Y.Z.; Zhou, A.; Ma, X.; Wang, S.G. Profit-aware edge server placement. *IEEE Internet Things J.* **2021**, *9*, 55–67. [CrossRef]
49. Guo, Y.; Wang, S.G.; Zhou, A.; Xu, J.L.; Yuan, J.; Hsu, C.H. User allocation-aware edge cloud placement in mobile edge computing. *Softw. Pract. Exp.* **2020**, *50*, 489–502. [CrossRef]
50. Wang, S.G.; Guo, Y.; Zhang, N.; Yang, P.; Zhou, A.; Shen, X.M. Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach. *IEEE Trans. Mob. Comput.* **2019**, *20*, 939–951. [CrossRef]
51. Liu, S.Y.; Liu, Y.H.; Ni, L.M.; Fan, J.P.; Li, M.L. Towards mobility-based clustering. In Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 24–28 July 2010; pp. 919–928.