Reliable Mobile Edge Service Offloading Based on P2P Distributed Networks

Lujie Tang, Bing Tang, Luyu Tang, Feiyan Guo, and Jiaming Zhang

School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 412101, China; ljtang@mail.hnust.edu.cn (L.T.); fyguo@mail.hnust.edu.cn (F.G.)
School of Information Engineering, Xiangtan University, Xiangtan 411105, China; 201705551025@smail.xtu.edu.cn
School of Entrepreneurship, Wuhan University of Technology, Wuhan 430070, China
School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China
Correspondence: btang@hnust.edu.cn (B.T.); zjiaming@whut.edu.cn (J.Z.)

Received: 13 April 2020; Accepted: 12 May 2020; Published: 16 May 2020

Abstract: Intelligent vehicles and their applications increasingly demand high computing power and low task delays, which poses significant challenges for providing reliable and efficient vehicle services. Mobile edge computing (MEC) is a new model that reduces the completion time of tasks and improves vehicle service by performing computation offloading near the moving vehicles. Considering the high-speed mobility of the vehicles and the unstable connection of the wireless cellular network, symmetric and geographically distributed edge servers are regarded as peers in a peer-to-peer (P2P) network, and a P2P-based vehicle edge offloading model is proposed in this paper to determine the optimal offloading server for the vehicle and the offloading ratio of tasks to achieve the goal of minimizing execution time. Because the edge computing infrastructure is deployed at the edge of the network, the data in the edge nodes are easily damaged or lost. Therefore, a P2P-based edge node fault tolerance mechanism is proposed to improve the reliability and fault tolerance of the system. The feasibility and effectiveness of our proposed system have been verified through simulation experiments, which greatly reduces the task completion delay.

Keywords: mobile edge computing; peer-to-peer; task offloading; internet of vehicles; fault-tolerance

1. Introduction

In recent years, with the continuous growth of the number of vehicles, road carrying capacity has reached saturation in many cities, and the issues such as traffic safety, travel efficiency, and energy saving have become increasingly prominent. The intelligence and networking of vehicles are generally considered to be an important way to solve the aforementioned traffic problems. As one of the main applications of the Internet of Things (IoT), the Internet of Vehicles (IoV) realizes the information exchange between vehicles and vehicles, between people and vehicles, and also between vehicles and roadside units (RSU), which greatly improves traffic safety. With the continuous popularization of the IoV, many applications are increasingly demanding high computing power and low response delay; however, at present the requirements exceed the computation capacity of the vehicle itself [1,2].

Advances in the IoT and wireless technologies have provided vehicles with more chances and better services (such as autonomous driving), which have made vehicles more intelligent. It also brings more challenges to the resource-constrained vehicles. As a solution, vehicle cloud computing (VCC) improves resource utilization and computing performance, it has unavoidable shortcomings with the increasing quality of service (QoS) requirements for smart vehicles: (1) Cloud servers that perform task are deployed in the centralized cloud center, therefore a large number of concurrent task requests may cause network congestion; (2) Cloud servers are far from the vehicles, and the bandwidth of...
the wireless communication is limited, which provides vehicles with a high transmission delay; (3) It is difficult for a remote cloud service to obtain the vehicle’s location and surrounding environment information in real time [3,4]. These shortcomings greatly affect the service experience when making decisions on cloud servers.

Edge computing is an open platform that integrates core capabilities of computing, storage, and services in the edge of the network that near the source of sensor data. It provides intelligent services in the edge, that meet the key needs of industry digitization, real-time data streaming, application intelligence, security and privacy protection, etc. Edge computing has received great attention from academia, industry, and government. Mobile edge computing (MEC) offers cloud capabilities and service environment for IoT applications at the edge of the mobile cellular network, such as 5G. To overcome the above-mentioned VCC problems, MEC is regarded as a promising solution in which computing resources are pushed to the radio access network (RAN) and services are provided near the vehicles [5].

MEC reduces transmission load of the cloud center, and also reduces task delay; however, there are still many edge servers that use a centralized stable server as a proxy to help users query the computing workload of edge servers. With the increase of smart vehicle number and business expansion, the number of mobile edge servers is also increasing. Waiting time has become a bottleneck in this centralized model, which damages the advantages of edge computing, especially for data-intensive services (such as car navigation, search engines, etc.).

In addition, because many edge servers are not protected by secure infrastructure, they are exposed to the natural environment and easily affected by external environment factors such as weather, temperature and humidity changes, network instability, or hardware failures, and cause some edge servers unavailable. In order to reduce delays requested by users and avoid access failures and data loss due to node failures, MEC systems need to provide lightweight, low-latency data integrity verification and recovery mechanisms, as well as efficient redundant backup mechanisms to ensure quickly restore the availability of affected/corrupted data within a limited time. It also needs to provide a dynamic trusted recovery mechanism for online upgrades and maintenance of software that supports business continuity after an attack or damage. To this end, we propose a combination model of vehicle mobility, task offloading, and fault-tolerance mechanism for vehicle edge computing (VEC) based on peer-to-peer (P2P) networks. Because P2P networks have high efficiency, reliability, decentralization, and good fault tolerance, our proposed strategy ensures that the system has low latency, fault tolerance, and resilience.

The main contributions of this paper are summarized as follows:

1. We have adopted a mobile edge server distributed management solution based on P2P architecture to maintain a reliable network for the mutual cooperation between mobile edge servers and smart vehicles. The P2P network uses a backup mechanism to efficiently use the idle bandwidth, storage capacity, and processing capacity of servers on the edge of the network, which solves the problem of delay in receiving information due to the mobility of vehicles, making it easier for smart vehicles gets the information available from the closest network-connected mobile edge server.

2. We have studied the reliability and fault tolerance of edge nodes in the VEC environment, and proposed a novel VEC offload and fault tolerance scheme based on P2P network, which solves the problems of task timeout and task failure caused by edge node failure, and improves user satisfaction.

3. We propose an efficient task offloading and fault tolerance algorithm. The optimal offloading server is selected for the task and the optimal offloading ratio is determined, minimizing the task execution time and considering the failure of edge nodes.

The P2P-based edge server fault tolerance mechanism proposed in this paper effectively solves the problems of task failure and task timeout caused by edge server failure. For some time-sensitive
tasks (such as face recognition, augmented reality, etc.), if an error occurs during the task offloading and cannot be processed in a timely and effective manner, it will greatly affect user experiences and QoS. In the previous research work, in order to avoid data loss caused by failures, different data copy storage methods are used to ensure data availability and reliability [6–11], rarely using P2P networks. The reliability, decentralization, and manageability of the network are used to study the fault tolerance of edge computing systems. The method we proposed is also applicable to face recognition, augmented reality, and other application scenarios. The networks we propose saves time, but it comes at the cost of space. We use a distributed P2P network for data backup, which consumes additional storage space. The vehicles need to be equipped with specific designed equipment. In addition to vehicles, the equipment used can be mobile phones, IoT devices, etc.

The rest of this paper is organized as follows: Section 2 summarizes related work about MEC, VEC, task offloading, reliability, and fault tolerance. Section 3 introduces the distribution and management scheme of the mobile edge servers. Section 4 models and analyzes the computing offloading in the P2P-based VEC environment. Section 5 studies the reliability and fault tolerance of edge nodes in the P2P-based VEC environment. Section 6 presents the simulation results and analysis. The final section summarizes the whole paper and discusses future work.

2. Related Work

In recent years, the edge computing paradigm has attracted great attention from academia and industry. It has the characteristics of fast processing speed and short response time, which could make cloud computing services closer to end users. Edge computing has provided a powerful driving force for many key technologies [12], such as 5G, IoT, augmented reality, and vehicle-to-vehicle (V2V) communication [13,14], etc. There are three common edge computing models: Cloudlets [15], fog computing [12,16], and mobile edge computing [17].

MEC, as a new architecture that moves service capabilities from the core network to the edge network, has caused extensive research by scholars [5,13]. The key research topics in mobile edge computing include the placement of edge servers [18–20], computing migration [21–23], and edge caching [24–26], etc. As the first step of mobile edge computing deployment, the placement of edge servers is critical. In [18], Li et al. studied the placement of edge servers based on energy-awareness and designed a particle-based Swarm-optimized energy-aware edge server deployment algorithms to find a more efficient low-energy placement scheme.

Computation migration and offloading research are about migration decisions and resource allocation. By migrating mobile device computing tasks/applications to servers in network for execution, it can enhance the computing capabilities of mobile devices and reduce time and energy consumption when running applications on mobile devices. A user-level online offloading framework (ULOOF) was proposed by Neto et al. [22]. The offloading framework uses a decision engine to model execution time and energy consumption based on historical data, and learns the execution time and energy consumption, and then predict the execution behavior to achieve the goal of minimizing remote execution overhead. Deng et al. provided a novel offloading strategy for workflow-type mobile services, and considered the user’s location and network connection failure in the offloading strategies, and added a mobility model and fault tolerance mechanism to design and implement an optimization method for offloading decision based on genetic algorithm (GA) [23]. To further reduce the latency and the transmission cost of the computation offloading, Zhang et al. [27], designed a task-file transmission strategy with predictive Vehicle-to-Vehicle (V2V) relay and proposed an optimal predictive combination-mode off-loading scheme, which greatly reduces the offloading cost. Considering computing tasks and load balancing among RSUs in the IoV, a heuristic algorithm is proposed by jointly considering task scheduling among MEC servers and downlink energy consumption of RSUs in [28], aiming to minimize the total energy consumption of RSUs under latency constraints.
Edge caching studies data storage and content delivery technologies. Using MEC servers as edge caching nodes can dynamically optimize content delivery services based on network status and wireless channel load. Since the MEC server is located at the edge of the network, very close to the location of the mobile device, it can combine user mobility and content access logs to optimize the experience. Cui et al. proposed a joint cache and offloading mechanism to solve the problem of minimizing the average total energy constrained by cache and deadline [24]. This mechanism involves task uploading and execution of tasks that do not cache computation results, and downloading the computation results of all tasks at the base station. It effectively uses multi-user diversity and multi-cast opportunities. Hao et al. introduced the new concept of task caching, and proposed the joint optimization of task caching and offloading on the edge cloud to meet the user’s latency requirements while minimizing the energy of mobile devices [25].

Extending edge computing capabilities to the edge of the vehicle network, VEC is emerging as a hot research topic. With the development of VEC, a new paradigm of networking has emerged, namely the vehicle edge network [29–31]. On heavy roads, the computing power limitations of the MEC server may affect the quality of offloading. To solve this problem, in [32] Zhang et al. proposed a hierarchical cloud-based VEC offloading framework, which introduced a nearby backup server to compensate for the lack of computing capacity of the MEC server. Based on this framework, an optimal multi-stage offloading scheme was designed using the Stackelberg game theory method, so as to ensure the delay constraints of computing tasks while maximizing its benefits. Regarding the problem of offloading the vehicle-connected computing tasks in a high-speed mobile environment, Huang et al. designed an MEC architecture based on software-defined networks (SDN) control to improve the expandability and flexibility of the vehicle-connected system, and use software-defined controllers to simplify the task offloading complexity [33]. Zhang et al. considered the completeness of the task data and the similarity of the geographical district to which the task belongs, and used the coordinated transmission of adjacent multiple vehicles to implement hierarchical data access to improve the data transmission rate of the task offloading [26]. In order to promote the development of VEC, there is another problem that must be solved, and that is security. In order to ensure security and improve network efficiency in VEC, Huang et al. proposed a distributed reputation management system (DREAMS) [34], which uses a VEC server to perform local reputation management tasks of the vehicle. It proposed the concept of reputation-assisted optimization and an optimized resource allocation algorithm was designed for optimizing resource allocation based on reputation in computing offload schemes, and the vehicle with higher reputation value can obtain the better response and improve user satisfaction.

In addition, due to the dynamics, distribution, and heterogeneity of edge systems, node failures (such as hardware and software failures, power failures, and network failures) often occur, resulting in data being unavailable. In order to avoid data loss caused by access failures and node failures, data replicas are usually placed on different nodes to ensure the availability and reliability [9–11]. In order to improve the availability of data in the edge computing environment and respond to the fluctuation of user access, Li et al. proposed a dynamic replica creation algorithm (DRC-GM) [9], which dynamically adjusts the number of replicas using the relationship between the access frequency of data blocks and the number of replicas to ensure data availability requirements. In order to find the best location for replica distribution, the replica placement algorithm (RPFNSG) was proposed. If we allocate some spare nodes for each job so that the process that terminated on the failed node can be restarted, considering the magnitude of the expected failure rate, the number of spare nodes allocated for each job will be high, resulting in a lot of resources are wasted. In [10], a dynamic node replacement algorithm was proposed by Prabhakaran et al., which can replace faulty nodes in time with other nodes in good working condition. The algorithm is deployed on the central scheduler and will be triggered when a node fails, which can result in a more efficient usage of resources. With the increasing deployment of mobile edge servers, their centralized management would cause certain performance issues. In order to address these challenges, Tang et al. [21] proposed a P2P-enabled decentralized
mobile edge server management scheme for the edge computing environment and a deadline-aware and cost-effective offloading approach, which aims to improve the offloading efficiency among mobile edge servers and allows more tasks to meet their deadlines. This scheme can enable all mobile edge servers to participate in updating information in a decentralized manner; however, it does not take advantage of the reliability and fault tolerance of distributed P2P networks to study the reliability and fault tolerance of edge nodes in the VEC environment, which is considered in this paper.

3. Distribution and Management of Mobile Edge Servers

For certain mobile edge servers, the proportion of offloading tasks remains to be determined. In fact, the offloading scheme should take into account the load status of the candidate edge server. These issues are illustrated in Figure 1. In order to facilitate understanding of system models and algorithms, we list main notations in Table 1.

Figure 1. Example of mobile edge computing and moving vehicles.
### Table 1. Notations.

| Symbol | Definition |
|--------|------------|
| $R$    | The set of road side units |
| $E$    | The set of mobile edge servers |
| $T_j$  | The task $j$ |
| $C_i^M$ | The computing power of the vehicle $i$ in performing tasks |
| $C_k^M$ | The computing power of the edge server $k$ |
| $d_{ij}$ | The amount of data uploaded by task $j$ |
| $d_{ej}$ | The amount of data downloaded by task $j$ |
| $c_j$ | The computation instructions for task $j$ |
| $x_{i,k}^j$ | Whether the task $j$ of the vehicle $i$ is offloaded to the edge server $k$ (=1) or not (=0). |
| $t_{d_{i,k}}^j$ | The time required for the vehicle $i$ to reach edge server $k$ |
| $t_{s_{i,j,k}}^j$ | The time required for the task $j$ to upload task data to the edge server $k$ |
| $t_{c_{i,j,k}}^j$ | The computation time for the task $j$ executed on the edge server $k$ |
| $t_{r_{j,z}}^i$ | The time required to transfer task result data from the edge server $z$ to the vehicle $i$ |
| $T_{x_i}$ | The transmission rate of data uploading |
| $T_{x_0}$ | The transmission rate of data downloading |
| $a_{i,k}^j$ | The offloading ratio of the task $j$ |
| $t_{f_{i,j,k}}^z$ | The query time for the edge server $z$ to find the computation result on the edge server $k$ |
| $t_{loc}^j$ | The time spent on performing the part of the task locally |
| $t_{act}^j$ | The actual execution time of the task $j$ without considering the occurrence of edge node errors |

### 3.1. Distribution of Mobile Edge Servers

We divide a one-way straight road into $m$ road side units (RSU), and the ID set of all RSU is expressed as $R = \{r_1, r_2, r_3, \ldots, r_m\}$. Many mobile edge servers are distributed along the road, with a mobile edge server at the center of each road unit, and ID number are expressed by $E = \{1, 2, 3, \ldots, m\}$. Due to the different radio power of each RSU and the change of the wireless environment, the size of the wireless coverage area of these RSU may be different, that is to say, the service range of each edge server is different. The distribution of RSU and mobile edge server, and their service ranges are shown in Figure 2.
3.2. Management of Mobile Edge Servers

In the edge computing environment, the actual VEC computing offloading usually involves two entities, namely the edge servers and the smart vehicles. Before providing computing offloading for smart vehicles, edge servers must be efficiently managed. It is assumed that each mobile edge server processes task offloading in a first-in-first-out (FIFO) order.

3.3. Client/Server (C/S) Architecture for Management of Mobile Edge Servers

The C/S architecture is a typical two-tier architecture, that is, client/server architecture. The centralized management scheme for mobile edge servers based on the C/S architecture is shown in Figure 3. There is a central server in the cloud layer, which is connected to all mobile edge servers through a wired network. Mobile users can establish network communications with edge servers over cellular networks. The central server acts as a proxy to help mobile users query the computing workload of the mobile edge server.

With this management scheme, any user’s query request must be proxied through the central server, and it will only be answered after the central server receives query responses from all edge servers. If the central server is under heavy load or some edge servers fail to return a query response in time, the user cannot get an immediate response.

With the increase of smart vehicles and business expansion, the number of mobile edge servers is increasing, and the efficiency and quality of tasks completion in mobile edge servers based on C/S architecture are gradually decreasing. As all data queries need to go through a central server, this makes single-point failures, security management, and authentication issues increasingly prominent.

In contrast to the C/S network architecture, peer-to-peer communication can be performed directly between nodes in a P2P network architecture. Each node simultaneously has the functions of receiving, storing, processing, sending media content, searching, and being searched for media metadata. Nodes can directly cooperate and share various resources. To this end, we explored the ways in which P2P networks can be used in an MEC environment.

![Figure 2. Road side units (RSU) and mobile edge server distribution and their service ranges.](image-url)
3.4. Peer-to-Peer Architecture for Management of Mobile Edge Servers

P2P is a distributed network. Participants in the network share a part of their software and hardware resources (processing power, storage capacity, network connection capacity, content resources, etc.). These shared resources provide services through the network and can be directly accessed by other peer nodes without the involvement of intermediate nodes. The participants in this network are both resource providers (server) and resource acquirers (client) [35,36].

The characteristics of P2P architecture are listed as follows:

1. Compared with the central server, the nodes in the system have obvious autonomy.
2. The system makes full use of resources at the edge of the network, such as storage/computing capabilities and content resources.
3. Dynamic changes of resources are applicable to the edge of the network (new resources are added and existing resources disappear).

With the rapid development of edge computing, there will be a large number of geographically distributed mobile edge servers throughout urban areas. The distributed management scheme of mobile edge servers based on the P2P architecture refers to the fact that the status of each edge server in the network is equal, and each node acts as a server to provide services to other nodes, while also consuming the services provided by other nodes. This distributed management solution can greatly alleviate the pressure on the central server side and single point of failure in a centralized architecture, and can make full use of the spare resources located at the edge of the network.

The main advantages of the distributed management solution for mobile edge servers based on the P2P architecture are summarized as follows:

1. **Efficiency**: Make full use of the idle bandwidth, storage capacity, and processing capacity of many mobile edge servers on the edge of the network.
2. **Reliability**: There are a large number of data replicas in the network, and the holders of these replicas are also independently distributed geographically, ensuring the reliability of resources. In addition, because the business is conducted directly between the nodes, the damage to some nodes or networks has little impact on other nodes. On the contrary, there will be no service interruption caused by a single point of failure in the C/S architecture.
3. **Decentralization**: Information transmission and service implementation in the network are performed directly between nodes, without the intervention of intermediate servers.
In addition, due to the system’s error tolerance and load balancing mechanism, it also makes management easier.

We believe that P2P networks enable distributed nodes to collaborate with each other and maintain a reliable network. When a high “churn” rate occurs, that is, when a large number of peer nodes frequently join and leave the network, the P2P network is highly robust. So, it is almost impossible for the entire system to crash. Therefore, P2P networks are a promising solution for decentralized management of mobile edge servers to offload task in an MEC environment, and some efforts have also been made using P2P model in the MEC environment [37–39]. The distributed management scheme of mobile edge server based on P2P architecture is shown in Figure 4. The system can theoretically accommodate tens of thousands of nodes. We simulated dozens of nodes through simulation experiments.

**Figure 4.** The distributed management of mobile edge server based on peer-to-peer (P2P) architecture.

4. Task Offloading Model in Vehicle Edge Computing Based on P2P

In our model, each vehicle $i$ usually performs only one computing task. A triple term can be used to describe the task $j$ performed by vehicle $i$, $T_j = \{di_j, do_j, c_j\}$, where $di_j$ represents the amount of data uploaded by the task, $do_j$ represents the amount of data downloaded, and the computation instructions for task $j$ is $C_j$ (in millions of instructions per second, MIPS). In fact, a vehicle cruising on the road may be connected with its nearest edge server geographically. Whenever it continues to move, its geographic location may move away from the previously connected edge server and closer to the next edge server with a strong wireless signal, and the vehicle will disconnect from the previous connection and establish a wireless connection to the new edge server. Because the transportation department has all the information on the highway, each smart vehicle can rely on the service range and computing power of each mobile edge server predetermined by the transportation department, and these edge servers are distributed along the traffic route.

We assume that each vehicle can transfer its current task to a mobile edge server only once, when driving on the road, and we define $x_{ijk} = \{0, 1\}$, when $x_{ijk} = 1$, it means that task $j$ of vehicle $i$ is partially or completely offloaded to edge server $k$, otherwise $x_{ijk} = 0$, so we get $\sum_{k=1}^{w} x_{ijk} \leq 1$.

4.1. Time Cost for Task Offloading without Considering Errors

If a user in a mobile vehicle wants to transfer their computation task to an edge server near them, the time cost consists of five parts: (1) the time it takes for the vehicle traveling smoothly to the vicinity of the edge server that the task needs to be offloaded; (2) the time required to establish a wireless
connection between the smart vehicle and the mobile edge serve, and upload the required data for the task to be processed; (3) the time that the smart vehicle waits for the edge server to complete task computation; (4) the time required to find the corresponding result data from the edge server; (5) the time the result data is transmitted back to the mobile vehicle.

Let \( l_{ik} \) denote the distance between vehicle \( i \) and edge server \( k \), and \( v_i \) denote the driving speed of vehicle \( i \), then the time required for vehicle \( i \) to reach edge server \( k \) is denoted by:

\[
  t_{dijk} = \frac{l_{ik}}{v_i} \tag{1}
\]

When the vehicle \( i \) arrives near the edge server \( k \), task \( j \) of vehicle \( i \) starts uploading task data to the edge server. The time required is expressed by:

\[
  t_{sijk} = \frac{d_{ij}}{T_{x_i}} \tag{2}
\]

The computation time for the specific task \( j \) of the vehicle \( i \) executed on the mobile edge server \( k \) is:

\[
  t_{cijk} = \frac{\alpha_{ijk} c_{ij}}{C_k} \tag{3}
\]

When the mobile edge server \( k \) finishes the offloading task, the geographical location of the vehicle may have moved away from the previously connected edge server \( k \) and closer to the next edge server \( z \) with a strong wireless signal. Time required to transfer task result data from edge server \( z \) to the vehicle is denoted by:

\[
  t_{r_{ijz}} = \frac{d_{oj}}{T_{x_0}} \tag{4}
\]

where \( T_{x_i} \) represents the transmission rate of data upload, \( T_{x_0} \) represents the transmission rate at which the task computation result is transmitted back to the smart vehicle, and \( \alpha_{ijk} \in [0, 1] \) represents the offloading ratio of task \( j \).

Then, the time from the start of task \( j \) to uploading data to edge server \( k \), and until the smart vehicle receives the result returned by edge server \( z \) is:

\[
  t_{i, offj, kz} = t_{sijk} + t_{cijk} + t_{f_{ijkz}} + t_{r_{ijz}} \tag{5}
\]

where \( t_{f_{ijkz}} \) is the query time for the mobile edge server \( z \) to find the computation result on the edge server \( k \). When \( k \) is equal to \( z \), we have \( t_{f_{ijkz}} = 0 \).

For task \( j \), the proportion represented by \( \alpha_{ijk} \) is executed on the remote mobile edge server, so the other proportion of the computing task should be executed on the vehicle itself, the time spent is:

\[
  t_{i, locj} = \frac{\left(1 - \alpha_{ijk}\right) c_{ij}}{C^M_i} \tag{6}
\]

where \( C^M_i \) indicates the computing power of vehicle \( i \) in performing tasks.

We assume that \( tw_k \) denotes the waiting time available for the \( k \)-th edge server. Without considering the occurrence of edge node errors, the actual execution time of task \( j \) is:

\[
  t_{j, act} = \max \left\{ t_{i, offj, kz} + \max \left\{ t_{dijk}, tw_k \right\}, t_{i, locj} \right\} \tag{7}
\]
4.2. Optimization Function

In this paper, the offloading scheme was designed to pick out the best offloading strategy to minimize time cost. The functions that need to be optimized are as follows:

\[
\min_{\{k, z \in E\}} F(a_{i,j,k}^t) = \sum_{k=1}^m x_{i,j,k}^t \cdot t_{j}^{\text{act}}
\]

\[
\begin{align*}
\{ x_{i,j,k}^t & = \{0, 1\} \cr
\sum_{k=1}^m x_{i,j,k}^t & \leq 1 \cr
a_{i,j,k}^t & \in [0, 1] \cr
\end{align*}
\]

s.t. \[ \begin{align*}
t_{i,k}^{\text{off}} + r_k/\nu_i & < t_w + t_{i,k}^t \cr
i_{k,z}^{\text{off}} + \max \{ t_{i,k}^{\text{off}}, t_w \} & \leq t_{i,z} + r_z/\nu_i \quad (r_z \in R) \cr
\end{align*} \]

where

\[
F(a_{i,j,k}^t) = \max \{ t_{i,k}^{\text{off}}, t_w \} \geq t_{i}^{\text{loc}}
\]

\[
= \begin{cases} 
t_{i,k}^{\text{off}} + \max \{ t_{i,k}^{\text{off}}, t_w \} & (t_{i,k}^{\text{off}} + \max \{ t_{i,k}^{\text{off}}, t_w \}) \geq t_{i}^{\text{loc}} \\
t_{i}^{\text{loc}} & (t_{i,k}^{\text{off}} + \max \{ t_{i,k}^{\text{off}}, t_w \}) < t_{i}^{\text{loc}} 
\end{cases}
\] (9)

If \( (t_{i,k}^{\text{off}} + \max \{ t_{i,k}^{\text{off}}, t_w \}) \geq t_{i}^{\text{loc}} \), we have

\[
F(a_{i,j,k}^t) = t_{i}^{\text{loc}} + t_{i}^{\text{off}} + t_{i,z} + t_{i,k}^{\text{off}} + \max \{ t_{i,k}^{\text{off}}, t_w \} + \frac{a_{i,j,k}^t c_i^j}{C_k}
\] (10)

If \( (t_{i,k}^{\text{off}} + \max \{ t_{i,k}^{\text{off}}, t_w \}) < t_{i}^{\text{loc}} \), we have

\[
F(a_{i,j,k}^t) = \frac{(1 - a_{i,j,k}^t) c_i^j}{C_i} + \frac{a_{i,j,k}^t c_i^j}{C_k}
\] (11)

As shown in the equations above, we need to determine two values to minimize our objective function, one is the offloaded edge server \( k \), and the other is the offloading proportion \( a_{i,j,k}^t \). From the above analysis, we know that if the \( k \) is determined, the optimization objective function becomes a unary linear function about \( a_{i,j,k}^t \). Because it is monotonic, we can easily find the optimal offloading ratio \( a_{i,j,k}^t \). Because the number of mobile edge servers on the road side is limited, we can use the traversal method to solve the optimal offloading proportion of tasks performed on each mobile edge server and obtain the task completion time on them. By comparing the execution time of different mobile edge servers, we can find the best offloading mobile edge server \( k^{bt} \) and the best offloading ratio \( a_{i,j,k}^{bt} \). Figure 5 presents the flow chart of the proposed two-stage offloading algorithm based on P2P architecture. The description of the first stage about selecting the optimal mobile edge server and determining the offloading ratio is shown in Algorithm 1, and the second stage about fault-tolerant offloading algorithm is introduced in the next section.
Figure 5. Flow chart of the two algorithms proposed in this paper.

Algorithm 1: Optimal mobile edge server selection and offloading ratio determination.

Input: The vehicle with computation task $T_{ij}$; vehicle speed $v_{ij}$; mobile edge servers set $E$ next to the road and their respective service ranges $R$ and computing power $C^c$.

Output: Optimal mobile edge server $k^{bt}$ and corresponding parameter $\alpha_{ij,k^{bt}}$.

1. $k^{bt} \leftarrow \phi$, $\alpha_{ij,k^{bt}}^{l} \leftarrow \phi$
2. foreach mobile edge server $k$ in $R$ do
   3. query its waiting available time $tw_{ij}^{l}$;
   4. Minimize $F(\alpha_{ij,k}^{l}, k)$ for $k$ with corresponding optimal $\alpha_{ij,k}^{l}$;
   5. Compare the time that each mobile edge server runs at the optimal $\alpha_{ij,k}$ and find the server $k^{bt}$ with the least time spent on all servers and corresponding optimal $\alpha_{ij,k^{bt}}^{l}$;
   6. $k^{bt} \leftarrow k$, $\alpha_{ij,k^{bt}}^{l} \leftarrow \alpha_{ij,k}^{l}$;
3. end
4. return $k^{bt}, \alpha_{ij,k^{bt}}^{l}$.
5. Fault-Tolerant Task Offloading in P2P Architecture

Unlike cloud computing, which has stable infrastructure protection, many edge nodes are exposed to the natural environment; therefore, it is very important to ensure the reliability of edge nodes. For example, computer vision-based public safety solutions need to rely on smart cameras for storage and computing. Because the edge computing infrastructure is located at the edge of the network, there is a lack of effective data backup and recovery measures. If a fault occurs, this will cause the task to time out and seriously reduce QoS and user satisfaction. In the edge computing scenario, once user data are lost or damaged on the edge node/server, and there is no backup of the user data in the cloud, the edge node does not provide an effective mechanism to recover the data. In order to minimize the consequences of this kind of unreliability of edge nodes, we need to design reasonable fault tolerance mechanisms to ensure the reliability of edge nodes and improve the fault tolerance of edge systems.

One of the key reasons why P2P networks are considered as an excellent platform for large-scale distributed systems is that P2P systems have better robustness and survivability than traditional C/S systems, and they have higher performance in elasticity for node failure. This resilience depends on the accurate and timely detection of node failures; therefore, we established the distributed management of mobile edge servers based on the P2P architecture. This management scheme can not only reduce the waiting delay for users to request, but also greatly improve the reliability of edge nodes and the fault tolerance of the system.

The network of P2P architecture enables distributed edge nodes to cooperate with each other and maintain a reliable network, which can quickly return to normal in a limited time when equipment fails or data is damaged or lost; therefore, using P2P networks for decentralized management of mobile edge servers is an effective solution to ensure data security and improve system reliability and fault tolerance.

For non-offload tasks, there is only one state: non-offload execution (SNE). For the offload task, there are three states: upload input data (SU), offloading execution (SOE), and download output data (SD). As mentioned before, because the edge server may suffer from node failure, network congestion or other faults at any time due to poor natural conditions or malicious attacks, the vehicles on the road may malfunction during the SU, SD, and SOE states.

When the mobile edge servers fails for various reasons, the fault tolerant process at different error stages is listed as follows:

1. **Data uploading stage**: select a new best edge server from other edge servers and upload the task again.
2. **Execution stage**: find and re-calculate the tasks on the edge server where the replica data is stored.
3. **Result output stage**: find the replica of the result and re-output it at the adjacent edge server in the geographic location.

We assume that after the intelligent vehicle selects the initial optimal offloading server $k^h_i$ and the optimal offloading ratio $\alpha^1_{i,k^h}$, if an error occurs during the upload of the task, the task computation instructions performed by the task migration to another mobile edge server is the same as before, we just need to select the optimal edge server to offload, and the offloading ratio remains the same; therefore, the execution time of the task offloading with fault tolerance becomes:

$$
FT_{i,off,j,k^h,z} = \begin{cases} 
\{ t_{i,off,j,k^h,z}^i + ts_{i,j,k^h}^i + tr_{i,j,k^h,z}^i + t_{i,j,k^h,z}^i + G + tr_{i,j,k^h,z}^f \} & \quad P_j^i \geq t_{i,off,j,k^h,z}^i \\
\{ ts_{i,j,k^h}^i + tr_{i,j,k^h,z}^i + G + tr_{i,j,k^h,z}^f \} & \quad P_j^i < t_{i,off,j,k^h,z}^i 
\end{cases}
$$

(12)

where $P_j^i$ represents the time from the start of the execution of task $j$ to the occurrence of the error. If $P_j^i \geq t_{i,off,j,k^h,z}^i$, there was no error in the offloaded task, the execution time of the task is still $t_{i,off,j,k^h,z}^i$. 


If $P_j^i < t_{k_{j, k}^{lh}}^{o f f}$, it means a failure occurred before the offload service completed. $z^*$ indicates the new mobile edge server that downloads the computation results after the fault occurs, and $G$ indicates the time traveling to the edge server adjacent to $z$ after the fault. $t_{k_{j, k}^{lh}}$ indicates the duration from the time when the data were uploaded to the time when the error occurred (if the error occurred during the data upload stage). $t_{k_{j, k}^{lh}}$ indicates the duration from the time when the task started to execute on the edge server to the time when the error occurred (if the error occurred during the task computation stage). $t_{k_{j, k}^{lh}}$ indicates the duration from the beginning of downloading result to the time when the error occurred (if the error occurred during the result download stage).

When $P_j^i < t_{k_{j, k}^{lh}}$, it means that the task fails during the SU state, then the new best mobile edge server $k^*$ will be chosen for offloading execution, where $td_{k_{j, k}^{lh}} = l^z_i/v_i - \left(\max\{td_{k_{j, k}^{lh}}, tw_{k_{j, k}^{lh}}\} + t_{k_{j, k}^{lh}}\right)$ represents the time it takes for the vehicle to travel from the original offloaded edge server $k_{j, k}^{lh}$ to the new optimal offloaded edge server $k^*$, and $tw_{k_{j, k}^{lh}}$ represents the waiting time for performing the offloaded task on the new edge server $k^*$. $t_{k_{j, k}^{lh}}^{o f f}$ indicates the duration from the time when the vehicle uploads the task data to the new edge server $k^*$ to the time when it receives the return result. If $t_{k_{j, k}^{lh}} \leq P_j^i < t_{k_{j, k}^{lh}} + t_{k_{j, k}^{lh}}$, the task failure occurs during the SOE state, the mobile edge server that stores the replica data will be found and the offloaded task will be re-calculated on it. $F$ indicates the search time of edge server with backup file based on P2P network after edge server failure. If $t_{k_{j, k}^{lh}} + t_{k_{j, k}^{lh}} + t_{k_{j, k}^{lh}}^{o f f} \leq P_j^i < t_{k_{j, k}^{lh}}^{o f f}$, it indicates that the task failure occurred during the SD state, after the failure occurs, the edge server with a copy of the calculated results will be found and the results will be transmitted by the next mobile edge server adjacent to $z$ in the geographical location.

The total time to complete task $j$ can be expressed as:

$$FT_{j}^{a c t} = \max\{FT_{j}^{o f f} + \max\{td_{k_{j, k}^{lh}}, tw_{k_{j, k}^{lh}}\}, t_{k_{j, k}^{lh}}^{o f f}\}$$

(13)

Objective function for offloading strategy can be expressed as:

$$\min_{\{k, z \in E\}} FT_F(x_{j, k}^i) = \sum_{k=1}^{m} x_{j, k}^i FT_{j}^{a c t} (14)$$

The description of fault tolerant offloading algorithm for mobile edge server based on P2P architecture is shown in Algorithm 2.

6. Simulation Results and Analysis

The simulation experiment in this paper was implemented by a self-designed edge scheduler written in Java language. By writing a program to simulate the entire process of the vehicle from entering the service range of the mobile edge server to task offloading, and finally receiving the returned results, choosing the best offloading server and offloading ratio according to our proposed offloading strategy, we could take into account the possible failures and errors during the execution of the task, by configuring a certain task error rate and fault-tolerance strategy. The simulation results mainly include the impact of different factors such as task error rate, intelligent vehicle speed, and different traffic congestion conditions on the average task completion time. The average task completion time of the edge node fault tolerance algorithm based on P2P network proposed in this paper was compared with the average task completion time without fault tolerance. Simulation scenario is described as follows: 10 mobile edge servers were randomly located on a road of 1000 m, and the service range of each mobile edge server was distributed in [20, 100] m. In our simulation, the average network upload speed was 10 Mb/s and the download speed was 30 Mb/s. We organized mobile edge servers along the road to form an unstructured P2P network. The simulation experiment parameters are shown in Table 2.
Algorithm 2: Fault-tolerant offloading of mobile edge servers based on P2P architecture.

**Input:** $P^i_j$, get parameter $t_{j,k}^{i,off}$, $t_{j,k}^{j,loc}$, $t_{j,k}^{f,off}$ by running Algorithm 1.

**Output:** $FT_{j,act}^i$.

1. if no error occurred then
   2. $FT_{j,k}^{i,off} \leftarrow t_{j,k}^{i,off}$;
   3. end
4. if the error occurred during the data upload stage then
   5. $/*$ select new server to re-upload $*/$
   6. $FT_{j,k}^{i,off} \leftarrow t_{j,k}^{i,off} + \max\{td_{j,k}^{i,off}, tw_{k}^*\} + t_{j,k}^{i,off}$;
   7. end
5. if the error occurred during the data calculation stage then
   6. $/*$ find the backed server and re-calculate $*/$
   7. $FT_{j,k}^{i,off} \leftarrow t_{j,k}^{i,off} + tc_{j,k}^i + F + t_{j,k}^{f,off} + tr_{j,k}^i + G + F + tr_{j,k}^i$;
   8. end
9. if the error occurred during the data return stage then
   10. $/*$ re-transfer on the adjacent server $*/$
   11. $FT_{j,k}^{i,off} \leftarrow t_{j,k}^{i,off} + tc_{j,k}^i + t_{j,k}^{f,off} + tr_{j,k}^i$;
   12. end
13. $FT_{j,act}^i \leftarrow \max\{FT_{j,k}^{i,off} + \max\{td_{j,k}^{i,off}, tw_{k}^*\}, t_{j}^{i,loc}\}$;
14. return $FT_{j,act}^i$.

### Table 2. Parameters and their values in the simulation.

| Parameter                              | Value               |
|----------------------------------------|---------------------|
| The length of the road (m)             | 1000                |
| The server range of mobile edge server (m) | [20, 100]          |
| The number of mobile edge server       | 10                  |
| Smart vehicle speed (km/h)             | [80, 140]           |
| Wireless upload speed (Mb/s)           | 10                  |
| Wireless download speed (Mb/s)         | 30                  |
| Amount of data uploaded for a task (MB) | [2, 5]              |
| The size of result data for a task (MB) | [0.5, 1]           |
| Computing instructions for a task (millions) | [20, 80]        |
| Computing Capacity for edge server (MIPS) | [3, 10]            |
| Computing capacity for smart vehicle (MIPS) | [1, 3]             |
| Task error rate                        | [0, 1]              |

The result of the task completion time of the vehicle at speed of 120 km/h and 25% task error rate under different traffic congestion situations is shown in Figure 6. As shown in this figure, we can see that when the offloading ratio is 0%, that is, the tasks were performed locally and when the offloading ratio was fixed at 25%, the execution time of the vehicle offloading task under different traffic congestion situations was similar. This indicates that at this time, the task completion time is mainly determined by the part of the task executed locally. As the proportion of offloading continues to increase, the average task completion time is declining. When the task offloading ratio reaches 75% or all are completed on the edge server, we found that the time cost of task completion is increasing, indicating that all available mobile edge servers have a high computing workload, and a large amount of task offloading will consume more time because of long waiting times.
The effect of traffic congestion on average task completion time on different error rates is presented in Figure 7. In the situation of relatively heavy traffic (0.5 s/car), the average task completion times for different error rates are 8.93, 9.71, 10.48, and 11.38 s, respectively. When the traffic flow is light (2 s/car), the average task completion time for different error rates are 6.76, 7.42, 7.92, and 8.21 s respectively, compared with previous heavy traffic, the task completion time is reduced by more than 2 s. This is because with the reduction of traffic pressure, the time required for vehicle task offloading queue is also continuously reduced, so that in the case of various error rates, the task completion time is also continuously reduced.

Figure 8 shows the effect of vehicle speed on the average task completion time at different error rates. We can see in this figure that as the vehicle speed increases from 60 km/h to 140 km/h, the average task completion time at each error rate also decreases. The average task completion time is reduced by 26.7% when no errors occur, and reduced by 23.6% when the error rate is 25%, and reduced by 20.6% when the error rate is 75%, respectively. This is because the faster the speed, the more mobile edge servers are available to offload, resulting in a reduction in task completion time.

In order to evaluate the impact of fault-tolerant mechanism in the process of vehicle task computation and offloading, we compared the task completion time of our proposed P2P-based fault-tolerant offloading strategy with the non-fault-tolerant offloading strategy in various situations, as shown in Figures 9–11. Figure 9 presents the average task completion time of fault-tolerant (FT) and non-fault-tolerant (without FT) offloading under different error rates measured at the vehicle speed of
120 km/h and a traffic congestion situation of 1.5 s/car. Through this figure, we found that as the error rate increases, the effect of our proposed fault tolerance strategy on reducing task completion time becomes more and more significant. Even at an error rate of 75%, our proposed fault tolerance strategy can save 10.6% of the time. Both the data in Figures 10 and 11 are measured under an error rate of 50%. From Figure 10 we found that when the traffic is relatively congested (0.5 s/car), our fault tolerance strategy obtains good results and can save about 18% of the time. At different speeds and different traffic congestion situations, our proposed fault-tolerant offloading strategy can save about 0.5 s of the time compared to the non-fault-tolerant strategy.

**Figure 8.** The impact of different vehicle speed on the average task completion time in different traffic congestion situations.

**Figure 9.** The impact of different error rate on average task completion time in different fault tolerance strategy.

**Figure 10.** The impact of vehicle traffic congestion on average task completion time in different fault tolerance strategy.
Figure 11. The impact of vehicle speed on average task completion time in different fault tolerance strategy.

Through previous analysis, we found that traffic congestion and vehicle speed are important factors affecting task completion time. In Figure 12, we compared the impact of vehicle speed and traffic congestion on task completion time (setting the task error rate to 25%). From the figure we can see that when the traffic speed is low and the traffic is congested, the completion time of the task is very high. As the speed of the vehicle increases and the traffic pressure decreases, the completion time of our task also decreases.

Figure 12. The impact of vehicle traffic congestion and vehicle speed on average task completion time (error rate is set to 25%).

7. Conclusions

In this paper, we made full use of the advantages of distributed P2P networks, and adopted a distributed management solution based on P2P architecture for mobile edge servers to provide efficient offload services for smart vehicles in vehicle edge computing environments. At the same time, in order to solve the problem of task timeout and failure caused by task failures or node failures, a novel task offloading and fault-tolerant strategy for vehicle edge computing based on P2P network was proposed. Simulation experiments have proved that our proposed task offloading and fault tolerance strategy can greatly reduce the average time for task completion and improve user satisfaction. Simulation results indicate that vehicle speed and traffic congestion condition have an impact on the task completion delay of vehicle offloading tasks. The faster the vehicle speed, the less time it takes to complete the task, and the better the traffic conditions, the less time it takes to complete the task.

In future work, we will consider more complex road and vehicle movements, establish a more accurate simulation model to evaluate the service range of the mobile edge server, and enable our proposed offloading fault tolerance strategy to adapt to more complex and dynamic road environments.
**Author Contributions:** Conceptualization, L.T. and B.T.; data curation, L.T.; formal analysis, J.Z. and B.T.; funding acquisition, B.T.; investigation, L.T.; methodology, L.T.; project administration, F.G.; software, L.T.; supervision, B.T. and J.Z.; validation, F.G. and J.Z.; visualization, F.G.; writing—original draft, L.T.; writing—review and editing, B.T. and J.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research is supported by the National Natural Science Foundation of China under grant No. 61602169, and the Natural Science Foundation of Hunan Province under grant No. 2018JJ2135, as well as the Scientific Research Fund of Hunan Provincial Education Department under grant No. 18A186.

**Acknowledgments:** The authors would like to thank all anonymous reviewers for their invaluable comments.

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

**Abbreviations**

The following abbreviations are used in this manuscript:

- **MEC** Mobile Edge Computing
- **P2P** Peer-to-Peer
- **IoT** Internet of Things
- **IoV** Internet of Vehicles
- **VEC** Vehicle Edge Computing
- **VCC** Vehicle Cloud Computing
- **RAN** Radio Access Network
- **V2V** Vehicle-to-Vehicle
- **QoS** Quality of Service
- **RSU** Roadside Units
- **C/S** Client/Server
- **MIPS** Millions of Instructions Per Second

**References**

1. Yang, F.; Li, J.; Lei, T.; Wang, S. Architecture and key technologies for Internet of Vehicles: A survey. *J. Comm. Inform. Netw.* 2017, 2, 1–17.
2. Ni, Y.; He, J.; Cai, L.; Pan, J.; Bo, Y. Joint Roadside Unit Deployment and Service Task Assignment for Internet of Vehicles (IoV). *IEEE Internet Things J.* 2019, 6, 3271–3283.
3. Boukerche, A.; De Grande, R.E. Vehicular cloud computing: Architectures, applications, and mobility. *Comput. Netw.* 2018, 135, 171–189.
4. Sun, F.; Cheng, N.; Zhang, S.; Zhou, H.; Gui, L.; Shen, X. Reinforcement Learning Based Computation Migration for Vehicular Cloud Computing. In Proceedings of the IEEE Global Communications Conference (GLOBECOM 2018), Abu Dhabi, UAE, 9–13 December 2018; pp. 1–6.
5. Mach, P.; Becvar, Z. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Commun. Surv. Tutor.* 2017, 19, 1628–1656.
6. Tang, B.; He, H.; Fedak, G. HybridMR: A new approach for hybrid MapReduce combining desktop grid and cloud infrastructures. *Concurrency Comput. Pract. Exp.* 2015, 27, 4140–4155.
7. Tang, B.; Tang, M.; Fedak, G.; He, H. Availability/Network-aware MapReduce over the Internet. *Inf. Sci.* 2017, 379, 94–111.
8. Tang, B.; Fedak, G. WukaStore: Scalable, Configurable and Reliable Data Storage on Hybrid Volunteered Cloud and Desktop Systems. *IEEE Trans. Big Data* 2017, doi:10.1109/TBDATA.2017.2758791.
9. Li, C.; Wang, Y.; Tang, H.; Zhang, Y.; Xin, Y.; Luo, Y. Flexible replica placement for enhancing the availability in edge computing environment. *Comput. Commun.* 2019, 146, 1–14.
10. Prabhakaran, S.; Neumann, M.; Wolf, F. Efficient fault tolerance through dynamic node replacement. In Proceedings of the 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Washington, DC, USA, 1–4 May 2018; IEEE Press: 2018; pp. 163–172.
11. Mohiuddin, M.; Saab, W.; Bliduz, S.; Le Boudec, J.-Y. A xo: Detection and Recovery for Delay and Crash Faults in Real-Time Control Systems. *IEEE Trans. Ind. Inform.* 2017, 14, 3065–3075.
12. Meng, Y.; Ali Naem, M.; Almagrabi, A.O.; Ali, R.; Kim, H.S. Advancing the State of the Fog Computing to Enable 5G Network Technologies. *Sensors* 2020, 20, 1754.
13. Khan, W.Z.; Ahmed, E.; Hakak, S.; Yaqoob, I.; Ahmed, A. Edge computing: A survey. Future Gener. Comput. Syst. 2019, 97, 219–235.

14. Gedeon, J.; Brandherm, F.; Egert, R.; Grube, T.; Mühlhäuser M. What the FogEdge Computing Revisited: Promises, Applications and Future Challenges. IEEE Access 2019, 7, 152847–152878.

15. Shaukat, U.; Ahmed, E.; Anwar, Z.; Xia, F. Cloudlet deployment in local wireless networks: Motivation, architectures, applications, and open challenges. J. Netw. Comput. Appl. 2016, 62, 18–40.

16. Stojmenovic, I.; Wen, S. The fog computing paradigm: Scenarios and security issues. In Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, Warsaw, Poland, 7–10 September 2014; IEEE: 2014; pp. 1–8.

17. Ahmed, E.; Rehmani, M.H. Mobile Edge Computing: Opportunities, solutions, and challenges. Future Gener. Comput. Syst. 2017, 70, 59–63.

18. Li, Y.; Wang, S. An energy-aware edge server placement algorithm in mobile edge computing. In Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; IEEE: 2018; pp. 66–73.

19. Wang, S.; Zhao, Y.; Xu, J.; Yuan, J.; Hsu, C.-H. Edge server placement in mobile edge computing. J. Parallel Distrib. Comput. 2019, 127, 160–168.

20. Xu, X.; Yue, Y.; Qi, L.; Zhang, X.; Wan, S.; Dou, W.; Chang, V. Load-Aware Edge Server Placement for Mobile Edge Computing in 5G Networks. In Proceedings of the 17th International Conference on Service-Oriented Computing (ICSOC 2019), Toulouse, France, 28–31 October 2019; pp. 494–507.

21. Tang, W.; Zhao, X.; Rafique, W.; Li, C.; Dou, W.; Ni, Q. An offloading method using decentralized P2P-enabled mobile edge servers in edge computing. J. Syst. Archit. 2019, 94, 1–13.

22. Neto, J.L.D.; Yu, S.-Y.; Macedo, D.F.; Nogueira, J.M.S.; Langar, R.; Secci, S. ULOOF: A user level online offloading framework for mobile edge computing. IEEE Trans. Mob. Comput. 2018, 17, 2660–2674.

23. Deng, S.; Huang, L.; Taheri, J.; Zomaya, A.Y. Computation offloading for service workflow in mobile cloud computing. IEEE Trans. Parallel Distrib. Syst. 2015, 26, 3317–3329.

24. Cui, Y.; He, W.; Ni, C.; Guo, C.; Liu, Z. Energy-efficient resource allocation for cache-assisted mobile edge computing. In Proceedings of the 2017 IEEE 42nd Conference on Local Computer Networks (LCN), Singapore, 9–12 October 2017; IEEE: 2017; pp. 640–648.

25. Hao, Y.; Chen, M.; Hu, L.; Hossain, M.S.; Ghoneim, A. Energy efficient task caching and offloading for mobile edge computing. IEEE Access 2018, 6, 11365–11373.

26. Zhang, S.; He, P.; Suto, K.; Yang, P.; Zhao, L.; Shen, X. Cooperative edge caching in user-centric clustered mobile networks. IEEE Trans. Mob. Comput. 2017, 17, 1791–1805.

27. Zhang, K.; Mao, Y.; Leng, S.; He, Y.; Zhang, Y. Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading. IEEE Veh. Technol. Mag. 2017, 12, 36–44.

28. Ning, Z.; Huang, J.; Wang, X.; Rodrigues, J.J.P.C.; Guo, L. Mobile edge computing-enabled Internet of Vehicles: Toward energy-efficient scheduling. IEEE Netw. 2019, 33, 198–205.

29. Liu, Y.; Yu, H.; Xie, S.; Zhang, Y. Deep Reinforcement Learning for Offloading and Resource Allocation in Vehicle Edge Computing and Networks. IEEE Trans. Veh. Technol. 2019, 68, 11158–11168.

30. Liu, J.; Wang, S.; Wang, J.; Liu, C.; Yan, Y. A Task Oriented Computation Offloading Algorithm for Intelligent Vehicle Network With Mobile Edge Computing. IEEE Access 2019, 7, 180491–180502.

31. Zhang, J.; Letaief, K.B. Mobile Edge Intelligence and Computing for the Internet of Vehicles. Proc. IEEE 2020, 108, 246–261.

32. Zhang, K.; Mao, Y.; Leng, S.; Maharjan, S.; Zhang, Y. Optimal delay constrained offloading for vehicular edge computing networks. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; IEEE: 2017; pp. 1–6.

33. Huang, C.-M.; Chiang, M.-S.; Dao, D.-T.; Su, W.-L.; Xu, S.; Zhou, H. V2V data offloading for cellular network based on the software defined network (SDN) inside mobile edge computing (MEC) architecture. IEEE Access 2018, 6, 17741–17755.

34. Huang, X.; Yu, R.; Kang, J.; Zhang, Y. Distributed reputation management for secure and efficient vehicular edge computing and networks. IEEE Access 2017, 5, 25408–25420.

35. Antony, I.T. Rowstron, A.I.T.; Druschel, P.Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), Heidelberg, Germany, 12–16 November 2001; pp. 329–350.
36. Seok, S.-J.; Song, W.-C.; Choi, D. Implementation of Pastry-based P2P system to share sensor data. *Int. J. Sens. Netw.* 2010, 8, 125–135.

37. Karagiannis, V.; Venito, A.; Coelho, R.; Borkowski, M.; Fohler, G. Edge computing with peer to peer interactions: Use cases and impact. In Proceedings of the Workshop on Fog Computing and the IoT (IoT-Fog@IoTDI 2019), Montreal, QC, Canada, 15 April 2019; pp. 46–50.

38. Ramachandran, U.; Gupta, H.; Hall, A.; Saurez, E.; Xu, Z. Elevating the Edge to Be a Peer of the Cloud. In Proceedings of the 12th IEEE International Conference on Cloud Computing (CLOUD 2019), Milan, Italy, 8–13 July 2019; pp. 17–24.

39. Yadgar, G.; Kolosov, O.; Aktas, M.F.; Soljanin, E. Modeling The Edge: Peer-to-Peer Reincarnated. In Proceedings of the 2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 2019), Renton, WA, USA, 9 July 2019; USENIX Association: Berkeley, CA, USA.