Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

Reliability and capability based computation offloading strategy for vehicular ad hoc clouds

Bo Li[1], Ziyi Peng[1], Peng Hou[1], Min He[1*], Marco Anisetti[2] and Gwanggil Jeon[3,4]

Abstract

In the Internet of Vehicles (IoV), with the increasing demand for intelligent technologies such as driverless driving, more and more in-vehicle applications have been put into autonomous driving. For the computationally intensive task, the vehicle self-organizing network uses other high-performance nodes in the vehicle driving environment to hand over tasks to these nodes for execution. In this way, the computational load of the cloud alleviated. However, due to the unreliability of the communication link and the dynamic changes of the vehicle environment, lengthy task completion time may lead to the increase of task failure rate. Although the flooding algorithm can improve the success rate of task completion, the offloading expend will be large. Aiming at this problem, we design the partial flooding algorithm, which is a comprehensive evaluation method based on system reliability in the vehicle computing environment without infrastructure. Using V2V link to select some nodes with better performance for partial flooding offloading to reduce the task complete time, improve system reliability and cut down the impact of vehicle mobility on offloading. The results show that the proposed offloading strategy can not only improve the utilization of computing resources, but also promote the offloading performance of the system.

Keywords: Vehicle self-organizing cloud, Computing offloading, Task waiting time, Resource utilization, Partial flooding

Introduction

The rapid development of smart city systems has promoted the development of the IoV. Today, many in-vehicle applications are applicable to mobile robots and autonomous driving. But this type of service requires a lot of complex data processing and calculations. By acquiring the corresponding computing resources from the cloud computing center, the unmanned vehicle can realize advanced application services such as environment recognition and driving cooperation, and form a vehicle-mounted cloud computing VCC (Vehicular Cloud Computing)[1, 2]. However, a large number of computing tasks and frequent data transmission may cause the cloud device to be overloaded, so that the calculation cannot be completed in time. The Vehicular Ad Hoc Cloud combines mobile vehicles into a distributed, self-organizing cloud computing environment. These nodes communicate with each other through the VENET (Vehicular Ad Hoc Network)[3, 4]. Part of powerful computing vehicles provide computing resources for other vehicles. They can share a certain amount of tasks for cloud devices and reduce the cloud network load and resource waste.

In the vehicle environment, a computationally intensive task must to be processed at a deadline. The vehicle can communicate with other equipment by V2V (Vehicle to Vehicle)[5], V2I (Vehicle to Infrastructure)[6] or V2C (Vehicle to Cloud)[7]. Then mapping the task to a node with more resources to reduce the hardware requirements. This method is called offloading. In the case of no infrastructure, the in-vehicle task can only be mapped to other devices with richer resources through V2V. The offloading scenario is shown in Fig. 1. Within the range of node connectivity, each node directly or indirectly in the form of single or multiple hops to offload tasks to high performance nodes.

*Correspondence: hemin@ynu.edu.cn
[1]School of Information Science and Engineering, Yunnan University, Kunming 650500, Yunnan, China
Full list of author information is available at the end of the article

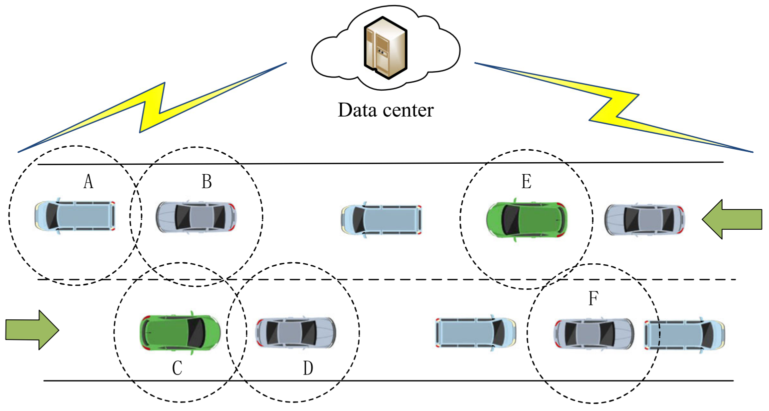


Fig. 1 Computational offloading scenario in a vehicle ad hoc network

In the dynamic environment of in-vehicle computing, the task completion time, resource utilization and system stability, is the optimization target. To optimize the above objectives, The contributions of this paper are as follows: in the network environment where computing and communication are integrated, in order to ensure the Quality of Service(QoS), we propose the partial flooding algorithm that considers node reliability and computing power. The offloading network is equivalent to a serial-and-parallel system, and the system reliability is utilized as an evaluation indicators to determine the number of offloading nodes, and after that select which nodes to execute task based on the computing power of the nodes and their link reliability. The partial flooding algorithm outperforms the existing schemes in terms of execution latency and offloading efficiency. It reduces the probability of network interruption and improves the resource utilization of the system.

The rest of this paper is organized as follows. In "Related work" section, we review the related work. "System model and problem formulation" section describes the vehicle self-organizing system and formulate the optimization problem to maximize the system computation capacity and minimize the offloading nodes number. In "The optimal solution" section, we promote the system performance by jointly optimizing the two parameters of link reliability and node computing power. Simulation results are presented in "Performance evaluation" section and the entire paper is concluded in "Conclusion" section.

Related work

Offloading in vehicle network is different from traditional offloading. Due to the mobility of various nodes in the vehicular self-organizing network, the computational offloading is largely restricted by the change of network structure. To cope with the impact of the dynamic change of vehicle position on the offloading performance, there are two main types of existing selection strategies for proxy resource. First, according to the offloading strategy, the task is offloaded to an optimal node selected by the strategy to complete the task. For example, for the scenario of path prediction, the literature[8–10] propose a path prediction algorithm based on driving habits. By collecting the position and speed of the vehicle to calculate the future driving path of the vehicle, it is determined the offloading node with the longest available link time to reduce the interruption in the course of the offloading process and decrease the failure rate of the task. Nevertheless, due to the randomness of the vehicle motion state, the prediction-based algorithm may not accurately predict the actual driving path of the vehicle node, resulting in a error between the prediction result and the actual result. Minimum hop algorithm [11–13] MH (Minimum Hop Algorithm) collects available resource information near the source node, maps the task to the target node with the smallest number of hops from the source node, and reduces the communication distance between the source node and the destination node which can cut down the completion time of the task. MH algorithm can find an optimum solution. In a dynamic environment, mobility node task completion time may lead to a communication link no longer satisfies the principle of the minimum number of hops, leading to the communication time to increase. The second is to use the flooding method[14, 15]. The task transmit to all available proxy resource nodes for simultaneous calculation, and the final result is returned from the earliest completed compute node to the source node. This method can obtain a globally optimal solution. However, it will squander quite a few many resources, and in multitasking scenarios, resource utilization is an important evaluation indicator. Therefore, how to reduce the waste of resources while preserving the advantages of the flooding algorithm is a crucial issue.

Although the above algorithm optimizes the computational offloading in the vehicular self-organizing network environment by reducing the task completion time, the impact on the stability of the performance of the offloading system is not taken into account in the previous work.

System model and problem formulation

In this section, we firstly introduce the computational offloading scenario and system models in the vehicular self-organizing network. Afterwards, we interpret the V2V offloading process. Finally, we develop an optimization problem to improve the system performance and minimize the number of nodes being offloaded.

The model of computing offloading

When there are multiple resources available, the network topology in the vehicular ad hoc network is equivalent to the communication system shown in Fig. 2 below.

The system consists of n high-performance nodes E, v_{ij} represents the *jth* hop node in the multi-hop link between the source node i and the target node. The vehicle terminal device v_c can perform a plurality of selection policies to offloading. For example, one node may be selected as a proxy resource or the task may be offloaded to all available node. Otherwise, just some of nodes with better performance may be selected to perform the offloading. When there is a V2V link between the target node and the source node, similar to the literature[16], in this paper, the end-to-end equivalent bandwidth is calculated as follows.

$$BW = \frac{MB}{HM} \tag{1}$$

MB is the base bandwidth in the VANET, and HM is the number of hops between end-to-end communications. Due to the dynamic change of the position of the vehicle node with time, the communication link between the source node and the target node will also change, so the equivalent bandwidth at each moment is uncertain.

System model

The vehicle node model is represented by the set V_i:

$$V_i = \{R_i, F_i, RW_i\} \tag{2}$$

R_i denotes the communicable radius of the vehicle node i, and the vehicles communicate with each other through DSRC (Dedicated Short Range Communications), the coverage area is a circular area with radius R_i, F_i presents the computing power of the vehicle node, and RW_i represents the read and write capabilities of the hardware that the node is configured with. The task model can be represented by a collection:

$$Q = \{D_i, C, D_o, T_\mathrm{d}\} \tag{3}$$

D_i represents the amount of uploaded data of the task, C is the amount of calculation of the task, D_o represents the amount of downloaded data of the task, and T_d represents the deadline of the task, that is, the maximum completion time that the task can tolerate, which is proportional to the calculation amount C of the task. Provided that the task is not completed within the T_d period, the task is declared to fail, and the task is counted as a failed task.

Offloading process

In the scenario, the source node that generates the task has a certain computing power. If the task is not offloaded, the time when the task runs locally is expressed as:

$$t_{local} = \frac{C}{F_{local}} \tag{4}$$

If $t_{local} > T_d$, it means that the task cannot be completed by the local device within the deadline. In this case, we can consider offloading the task to the high-performance node to perform the operation. As to Docker uses lightweight virtualization technology for resource isolation and encapsulates various environmental dependencies into web applications, it can be easily ported and redeployed. Therefore, the Docker container-based offloading system can effectively reduce resource consumption. The system deployment steps are as follows. S1: Locally, package Java or other application tasks into Docker

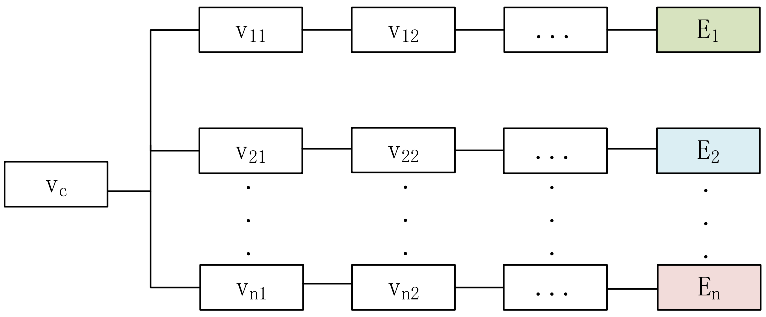


Fig. 2 Network topology in a vehicle-mounted ad hoc network

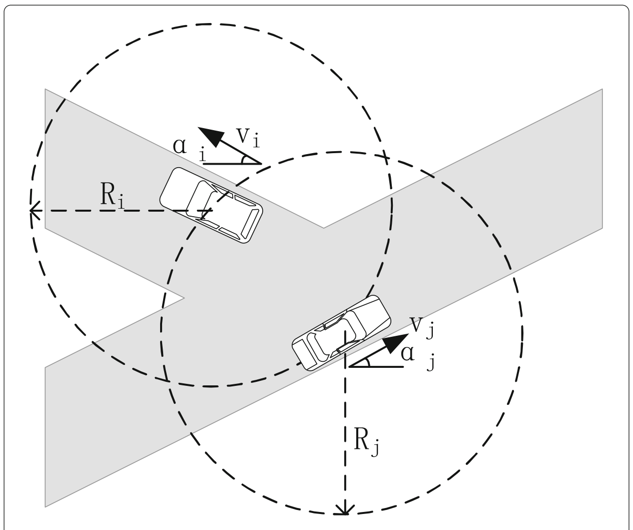


Fig. 3 Schematic diagram of adjacent vehicle nodes

image files and upload them to the Docker repository; S2: The high-performance node creates a container based on the uploaded Docker image, runs the Docker instance, and implements interactive access. According to the system deployment steps, the computing offloading can be divided into three processes of uploading, executing, and downloading. The execution process of each part is as follows.

(1) Upload Process

The time the task is packaged locally into a Docker image file is relevanted to the read/write speed of the hard disk, so the packaging time is symbolized by:

$$t_{pack} = \frac{D_i}{RW_{local}} \tag{5}$$

Assume that when the task is uploaded, the bandwidth of V2V maintain stability for a certain period of time t_x, then the amount of data transmitted during this period can be expressed as:

$$D_{u_x} = t_x \times BW_x \tag{6}$$

At a certain moment i, the hop count has changed n times, and at the time j the nth change occurs, if the following formula is satisfied:

$$\sum_{x=0}^{n-1} t_x \times BW_x + (i-j) \times BW_n = D_i \tag{7}$$

It means that at time j, the task upload is completed, and from the start of the uploading time to the time j, the entire time span t_{trans} is recorded as the transmission time of the task. The upload time of the task is recorded as:

$$t_{up} = t_{pack} + t_{trans} \tag{8}$$

(2) Execution Process

When the task is offloaded to the target node id_c, its execution time in the container is:

$$t_{ex} = \frac{C}{F_{id_c}} \tag{9}$$

We do not consider the dynamic changes in the computing power of the vehicle, assuming that the computing power provided by each vehicle is stable.

(3) Download Process

When the task calculation is completed, the target node returns the calculation result to the source node, which is similar to the upload process of the task. At a certain moment α, the hop count has changed n times, and the time at the nth change is β. If

$$\sum_{x=0}^{n-1} t_x \times BW_x + (\alpha - \beta) \times BW_n = D_o \tag{10}$$

At this point, the task download is completed, and the entire period is recorded as t_{down}. The total completion time t_{sum} and total communication time t_{com} of the task are respectively expressed as:

$$t_{sum} = t_{up} + t_{ex} + t_{down} \tag{11}$$

$$t_{commu} = t_{up} + t_{down} \tag{12}$$

Problem analysis

In order to comprehensively and objectively evaluate the importance of each link in the communication network, the concept of communication link reliability[17–20] is introduced in the partial flooding algorithm. The two factors of link reliability and node computing power, on the condition that the completion time is guaranteed, the reliability of the system is improved. Since the offloading includes two sub-processes of communication and computation, to optimize the system, the problem can be split into two sub-problems. One is to solve the reliability of the communication link of the node, and the other issue is to systematically evaluate the offloading performance according to the computing power and link reliability, and finally solve the two sub-problems comprehensively, so that the offloading strategy finally selected can improve the offloading performance and save some resource costs.

Link reliability analysis

Ad hoc network is a wireless communication self-organizing network system composed of many mobile nodes. On account of the dynamic change of network topology and the unreliability of wireless communication links, the network scale expands and the routing communication overhead in the offloading system becomes more and more difficult to control. Factors such as the mobility of the nodes in the network and the uncertainty of the state may lead to network Partition, which result in the link to be broken. Route failure is a vital factor affecting the reliability of the offloading system. During the task uploading or downloading process, the link breakage will inevitably lead to data retransmission, which greatly increases the network overhead and task completion delay. Therefore, selecting a target node with a V2V link with higher reliability is also an important problem that needs to be solved.

Networked vehicles can obtain the position information of each vehicle in the network through GPS positioning system[21]. The moving speed and heading information of each vehicle can also be acquired by using sensors[22](for instance, speed sensor, acceleration sensor, etc.) equipped with the vehicle. Therefore, if the motion parameters of two adjacent nodes (such as speed, direction, available communication distance, etc.) are known we can predict the connection time between the two nodes to judge the reliability of the communication link. In the scenario shown in Fig. 3, at time t, the position of node i can be expressed as $(x_i(t), y_i(t))$, its communication radius is R_i, the velocity is $v_i(t)$, and the node motion direction is α_i. Similarly, the motion states of adjacent nodes can be obtained.

Considering the traveling speed of the vehicle in the road, the motion state information of the vehicle is refreshed at intervals of 1s, and thus it is known that the distance between the two nodes after Δt satisfied the following relationship:

$$D_{ij}(t+\Delta t) = \sqrt{(x_i(t+\Delta t) - x_j(t+\Delta t))^2 + (y_i(t+\Delta t) - y_j(t+\Delta t))^2} \tag{13}$$

When the distance between the two nodes satisfies $D_{ij}(t+\Delta t) = R_i + R_j$, and the next moment the distance meets $D_{ij} > R_i + R_j$, as shown in Fig. 4. The communicable range of node i and node j no longer intersects. The maximum communication distance between the two nodes is reached, marked as the critical communication point.

Thus, based on the motion state information of the node, the available connection time between the nodes can be calculated:

$$\Delta t = \frac{\sqrt{\left(\Delta {v_i}^2 + \Delta {v_j}^2\right)(R_i+R_j)^2 - (\Delta v_i \cdot \Delta y - \Delta v_j \cdot \Delta x)^2} - (\Delta v_i \cdot \Delta x + \Delta v_j \cdot \Delta y)}{\Delta {v_i}^2 + \Delta {v_j}^2} \tag{14}$$

$$\begin{aligned} \Delta v_i &= v_i(t)\cos\alpha_i - v_j(t)\cos\alpha_j \\ \Delta v_j &= v_i(t) sin\alpha_i - v_j(t) sin\alpha_j \\ \Delta x &= x_i(t) - x_j(t) \\ \Delta y &= y_i(t) - y_j(t) \end{aligned} \tag{15}$$

To deal with Δt, as an indicator to evaluate the link communication quality, the reliability of link between nodes is introduced:

$$P_{ij} = \frac{C_{ij}(\Delta \mathrm{t})}{C_{ij}(\Delta T)} \tag{16}$$

ΔT represents the time between the nodes i and j to ensure that the offloading task is completed, and Δt represents the time actually available during $\Delta T.C_{ij}(\Delta \mathrm{T})$ and $C_{ij}(\Delta t)$ represent the amount of data that this link can transmit in ΔT and Δt, respectively. It is defined as the product of the bandwidth BW and the corresponding link time. Link reliability can be further expressed as:

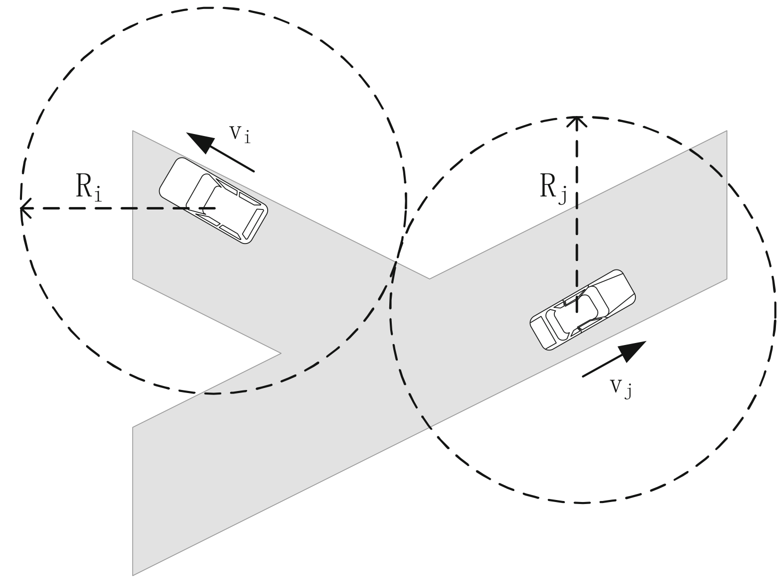


Fig. 4 Critical communication point

$$P_{ij} = \frac{C_{ij}(\Delta \mathrm{t})}{C_{ij}(\Delta T)} = \frac{BW \times \Delta \mathrm{t}}{BW \times \Delta T} = \begin{cases} \frac{\Delta t}{\Delta T}, \frac{\Delta t}{\Delta T} < 1 \\ 1, \frac{\Delta t}{\Delta T} \geq 1 \end{cases} \tag{17}$$

The value range of P_{ij} is [0, 1]. The larger the value, the better the link reliability. When the available connection time is greater than the task required, it means that the link disconnection does not occur during the task transmission, and the reliability between the two nodes is set to 1. Assuming that the source node generates the task computing demand at a certain moment, the vehicle sends a request to the nearby computing resources through the Internet of vehicles broadcast[23], and the nearby vehicles with powerful computing capability send a reply to the source node after receiving the request, including their own computing power and storage capacity. After that, the source node establishes the link reliability model for n high performance nodes in the system according to the received reply information, which is used for the complete evaluation of the reliability of the whole system.

In the vehicle self-organizing network, when the source node and the target node require communicating indirectly through the multi-hop mode, the communication link can be regarded as a serial system, as shown in Fig. 5.

Each can be viewed as a single hop link between each pair of neighboring nodes. The reliability of the tandem system with a total hop count of J_i to the target node E_i can be expressed as:

$$P_{E_i} = \prod_{j=1}^{J_i} P_{ij} \tag{18}$$

P_{ij} represents the reliability of the *jth* hop link to the edge server E_i. It can be seen from the above formula that to ensure the reliability of the system, the reliability of each subunit is higher and the number of series is less, and the system is more stable. When only one multi-hop communication link still cannot guarantee the reliability that the whole system needs to meet, the partial flooding algorithm proposed in this paper assumes that the number of all available edge servers is N, and only $K(> 0, \leq N)$ nodes are offloaded, and the system reliability is:

$$P_s = 1 - \prod_{i=1}^{\mathrm{K}} [1 - P_{E_i}] \tag{19}$$

The reliability definition of the parallel system can be derived that the better the reliability of each subunit, the better the reliability of the system. Contrary to the series system, the more parallel branches, the higher the stability that the system can achieve. In the vehicular self-organizing network environment, the communication network based on the partial flooding algorithm equivalent to a serial-and-parallel system, that is, each target node and the source node may need to go through multiple hops before proceeding indirect communication. If the system selects k nodes as the target node and J_i represents the serial cascade hop count of the *ith* branch, the reliability of the serial-to-parallel system can be expressed by the following formula:

$$P_s = 1 - \prod_{i=1}^{K} [1 - \prod_{j=1}^{J_i} P_{ij}] \tag{20}$$

Problem formulation

The current researches on the strategy focuses on offloading tasks is selecting a node for offloading. In the in-vehicle dynamic environment, the randomness of node motion may cause the communication link to be broken, affecting the completion time of the task; though the flooding strategy can greatly reduce the impact of vehicle node mobility on computing offload, for it occupies too much system resources, it will cause a certain waste of resources. Therefore, selecting the appropriate number of offloading nodes can improve the reliability of the system while reducing the completion time of the task.

According to the above definition, the smaller the series number J_i is, the more reliable the system is. The smaller the parallel count k is, the less reliable the system is. The more the number of offloading nodes selected, the more reliable the system and the better the performance, but the problem of excessive resource usage will be brought about. However, if the number of target nodes is too small, the reliability of the system is reduced, which may result in the task being impossibly completed within the specified deadline. To balance the performance of the system and the resource performance, each of the offloading can get a certain gain, first, it is necessary to ensure the minimum number of the series, and each parallel branch of the communication link can be solved by the minimum hop algorithm [24]. Furthermore, based on the above analysis, we attach importance to the selection of the number of offloaded nodes and how to choose, in order to improve the success rate of task completion, and reduce the system resource consumption, then the problem can be expressed as:

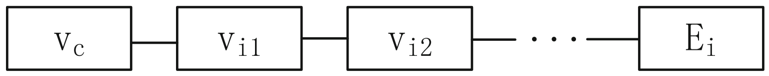


Fig. 5 Series system logic diagram

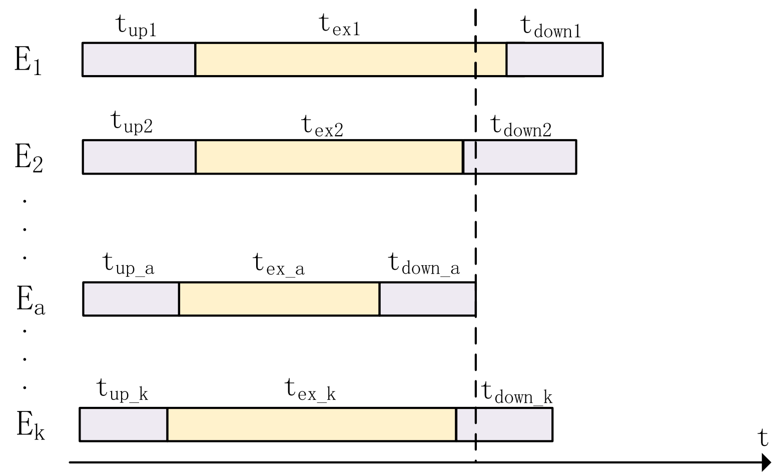


Fig. 6 Multi-resource execution computing offloading

$$
\begin{aligned}
&P1: arg \min_{k}(P_s - P_T) \\
&\quad s.t. P_s \geq P_T \qquad (a) \\
&\quad T_{sum} \leq T_d \qquad (b)
\end{aligned}
\tag{21}
$$

In the formula, arg min represents the value of the variable when the objective function takes the minimum value, that is, the value of the selected number of nodes k can minimize the difference between the system reliability P_s and the set certain threshold P_T. The smaller the number of offloading nodes selected, the smaller the system overhead. The constraint condition (a) guarantees that the reliability of the offloading system must be greater than the set system reliability threshold, that is to say, the influence of the reliability of the system on the offloading cannot be neglected for the minimum value obtained; the constraint condition (b) Considering the performance of each resource, required that the time T_{sum} for the selected k nodes to complete the task is enough to meet the deadline, to reduce the failure rate of the task. Where T_{sum} includes the communication time of the task, execution time and the waiting time of the task when the communications link is broken, namely:

$$T_{sum} = t_{sum} + t_{wait} \tag{22}$$

The optimal solution

To satisfy constraint conditions (a) and (b), the influence of node computing power and communication link reliability on offloading should be considered. System instability can weaken the performance of the entire system to some extent. If the selected node has strong computing power, but the communication link reliability of the node is poverty, the constraint condition (a) cannot be met. At this time, the increase of the communication time may cause the performance of the offloading system to fail to achieve the expected offloading effect. Selecting a node with a stable communication link performs the offloading, but the computing power of the node is small, and the task may not be completed within the deadline, and the constraint condition (b) cannot be met, causing the task failed. Partial flooding algorithm not only needs to consider the computing power of the node, but also needs the reliability of the communication link as another parameter indicator that affects the offloading, and makes a comprehensive evaluation of the system. Since the value of the communications link reliability, P_{E_i} ranges from 0 to 1, when $P_{\mathrm{E}_i} = 1$, the source node can transmit the task to the target node within the available communication time, and the communication quality will not affect the offloading. The entire offloading process is equivalent to static offloading; when $0 < P_{\mathrm{E}_i} < 1$, the communication link is unstable, and the offloading performance will change with the link reliability. Therefore, the impact of the reliability of the communication link on the offloading performance is Expressed as:

$$C_s = C_i \times P_{\mathrm{E}_i}, 0 < P_{\mathrm{E}_i} \leq 1 \tag{23}$$

In the above formula, C_i is equal to the computing power of node i; C_s is the node performance size after equivalent processing of link reliability and node computing power. In order to solve the minimum value of k satisfying the condition, the C_s of each node are sorted in descending order, and the node with the higher ranking indicates that the comprehensive performance is better. Therefore, the nodes are selected in the order of comprehensive performance, and P_s and T_{sum} are updated according to equations (19), (20), and (22). The minimum k value can be found to make the constraint conditional and reduce the waste of system resources. With the increase in the number of selected nodes, the values of P_s and T_{sum} will be prone to converge. At this time, the number of offloading nodes has little effect on the system performance, on the contrary, the effect of several offloading nodes on system resource utilization has been increased. It is obvious that the number of nodes selected based on the partial flooding algorithm can meet the requirements of performance and reduce the waste of resources of the entire system.

If all the R times are combined and the above constraints are still not met, task selection executes locally; if k nodes are selected, based on the flooding concept, the task is uploaded to the selected node for computing. Finally, the result is returned from the fastest completed node to the source node. At this point, the node that has not been completed discards the task cleans up the memory space and prepares for the next task. The algorithm flow is shown in Algorithm 1.

Algorithm 1 The offloading algorithm based on partial flooding.

```
Input: The set of available resources,$E_n$; The set of node
    reliability,$P_{E_i}$; The set of node computing power, $C_i$;
Output: The optimal number of offloading nodes,$k^*$;
    The system reliability,$P_s$; The task completion
    time,$T_{sum}$;
 1: for each $E_i \in E_n$ do
 2:     $C_{s(i)} = C_i \times P_{E_i}$;
 3: end for
 4: Sort nodes according to the obtained $C_{s(i)}$ in descend-
    ing order;
 5: Select $k$ by sorting results;
 6: initial $k = 1$ and $P_s = C_{s(1)}$;
 7: while $P_s < P_T, T_{sum} > T_d$ do
 8:     $k \leftarrow k + 1$.
 9:     update $P_s$ and $T_{sum}$;
10: end while
11: $k^* = k$.
12: return $k^*$.
```

The computational complexity of the partial flooding algorithm is $O(n^2)$. Compared with the flooding algorithm, the partial flooding algorithm reduces the scope of flooding and thus decrease the system overhead.

When there are multiple resources available, in the partial flooding algorithm, as one of the reference data for evaluating the number of offloading nodes, the resource validity u_e is defined as:

$$u_e = \frac{u_{ac}}{u_{sum}} \tag{24}$$

u_{ac} indicates the size of the resource consumed by the proxy resource whose execution result is adopted, and u_{sum} indicates the total resource size consumed in the system during the entire offloading process. Resource overhead can be divided into the communication overhead and the computational overhead. The communication overhead of node i is the size of the network bandwidth occupied during the offloading process, indicating the amount of data actually passing through a certain network during the communication time t_{com}:

$$u_{com} = BW_i \times t_{com\_i} \tag{25}$$

The node i computational overhead is the computational resource size consumed during the computation time t_{ex}:

$$u_{ex} = F_i \times t_{ex\_i} \tag{26}$$

Generally, the ratio of the communication time of the task to the calculation (execution) time of the task is defined as the Communication Computation Rate (CCR), which can be expressed as:

$$CCR = \frac{(D/BW)}{(C/F_i)} = \frac{t_{com}}{t_{ex}} \tag{27}$$

According to the CCR definition, when the calculation amount C of the task is determined, the data amount of the task can be derived by the above formula. Then we can get the weight of the communication process and the execution process when the task is offloaded:

$$\omega_c = \frac{t_{com}}{t_{com} + t_{ex}} \tag{28}$$

$$\omega_e = \frac{t_{ex}}{t_{com} + t_{ex}} \tag{29}$$

For different nodes, the offloading overhead is as shown in the figure below. t_{up} and t_{down} are the task upload time and download time of the node respectively, $t_{com} = t_{up} + t_{down}$.

It is assumed that k nodes are selected as proxy resources to perform computational offloading since the execution result is downloaded from node a that first completed the calculation to the source node. When the result is successfully downloaded, the offloading is complete. Therefore, the actual useful resource consumption in the whole process is the computational overhead of node a and the communication overhead of task uploading and downloading, and the resource utilization of the system can be obtained by the following formula:

$$u_e = \omega_c \times \frac{BW_a \times t_{com\_a}}{\sum_{i=1}^{k} BW_i \times t_{up\_i} + \sum_{i=1}^{k} BW_i \times t_{down\_i}} + \omega_e \times \frac{F_a \times t_{ex\_a}}{\sum_{i=1}^{k} F_i \times t_{ex\_i}} \tag{30}$$

ω_c and ω_e are the weighting values of the communication process and the execution process, respectively, obtained from the CCR. Fig. 6 shows an example of calculating the CCR, when the task on the node a is completed, the node 2 and the node k have not yet completed the task download. Then, the download time of this type of the node is from the task download time of the node to the task completion time of the node a. For nodes like node 1, since the node a task is completed, they have not started the task download process, and the download time $t_{down} = 0$. Similarly, for the execution time, when the task is completed at node a, node 1 has not completed the calculation. Thereafter, the execution time of the node is the time from the start of the calculation of the node to the task completion time of the node a, and if the node has not started the calculation, the execution time $t_{com} = 0$. The larger u_e is, the less resources is wasted during the offloading process and the higher the resource utilization. Compared to the flooding offloading algorithm, the partial flooding algorithm reduces the number of offloading nodes and improves resource utilization under the condition of guaranteeing constraints.

Performance evaluation

In this section, we will present simulation results to the performance enhancement of the proposed partial flooding algorithm based on link reliability and computing power. Commonly used node movement models can be divided into geographically restricted and geographically unrestricted. To accurately simulate the simulation scenario under the unmanned scene, this paper uses the Manhattan network model as the moving model of the vehicle node, as shown in Fig. 7.

In the Manhattan model, the movement of the vehicle is limited by surrounding roads or else traffic lights at buildings and intersections. When the vehicle arrives at the intersection, the probability that the vehicle keeps on moving in the same direction can be recorded as p_1, while the probability of traveling left and right are p_2 and p_3, respectively. In this paper, the experimental simulation environment simulates the environment of urban roads. The simulation area is $2500m \times 2500m$, and one intersection is set every 500m in the x-axis and y-axis directions. The intersection traffic light length is generated according to a certain proportion, and there are 30 vehicle nodes in the setting scene. In the predicted scene, the initial turning probability of the vehicle is: straight line $p_1 = 0.5$, left turn $p_2 = 0.25$, right turn $p_3 = 0.25$. Vehicles can communicate by multi-hop. The parameter settings are shown in Table 1.

The experimental data of this experiment are the system reliability P_s, the task offloading time t_{sum}(including the task communication time and the task computation time), the time t_{wait} of the node waiting for the link when the link is disconnected, the task completion time t_{finish}, and the resource utilization rate u_e. among them,

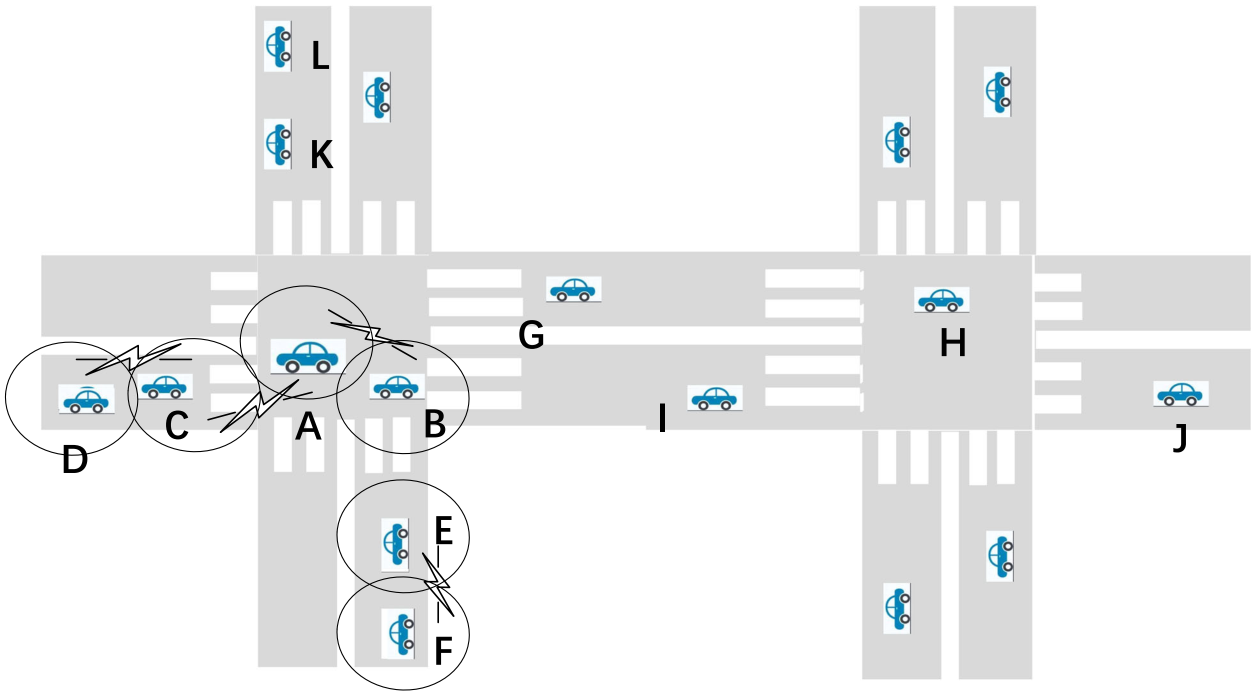


Fig. 7 Vehicle-based self-organizing network based on Manhattan motion model

Table 1 Simulation parameter table

Parameter	Value/unit	Parameter	Value/unit
Number of simulations	1000(time)	Simulation area	$2500 \times 2500(m^2)$
Number of vehicles	30	Junction width	15(m)
High performance node ratio	20%	end-to-end delay	[100 500] (ms)
V2V single hop bandwidth	11(Mbps)	communicable distance	[200 300](m)
Task calculation	$[8\ 10] \times 10^6$(MI)	Vehicle speed range	[15 30] (m/s)
Task data volume	[50 100](M)	Vehicle computing power	82332×[1 2] (MIPS)
Result data volume	[50 100](M)	High-performance vehicle computing power	82332×[5 6] (MIPS)

$t_{finish} = t_{sum} + t_{wait}$. To evaluate the efficiency of the partial flooding algorithm, we compared it to the following scenarios.

Minimum hop algorithm: At the time of task generation, the hop count of the current source node to the target node is calculated, and the node with the smallest hop count is selected as the target node for offloading.

Path prediction algorithm: By predicting the motion trajectory of the node in the future time, the node with the shortest task completion time is judged, and it is selected as the target node.

As shown in Fig. 8, compared with the traditional offloading algorithm based on the MH algorithm and the prediction based offloading algorithm, the partial flooding algorithm selects a group of nodes with better offloading performance as target nodes. According to the content of the system reliability mentioned above, it can be seen that the partial flooding algorithm is effectively improved in link reliability. The selected set of offloading nodes constitutes a string-and-parallel system that is more reliable than the tandem system that selects only one node. For the flooding algorithm, since the task is offloaded to all nodes to perform computation, the increase in parallel branches increases the reliability of the system. However, the flooding algorithm will lead to excessive waste of resources. For the comparison of system resource utilization, a detailed analysis will be made below.

Figure 9 shows that the reliability of the communication system increases as the number of offloading nodes. And the task completion time decreases as the number of nodes. It can be seen from the figure that the number of nodes is selected from 2 to 4, the rate of system reliability is gradually reduced, and the rate of task completion time is gradually reduced. Although with the number of nodes selected for offloading increasing, the system is more reliable and the task completion time is smaller, the system performance improvement effect is not obvious, and the system resource occupancy rate is proportionally increased. Therefore, the number of offloading nodes is controlled to be about two to four depending on the set threshold of system reliability. Compared with the traditional algorithm that only offloads tasks to one node, the proposed algorithm not only improves the system reliability but also reduces the task completion time.

As shown in Fig. 10, for the three items task offloading time, waiting time and completion time, the proposed

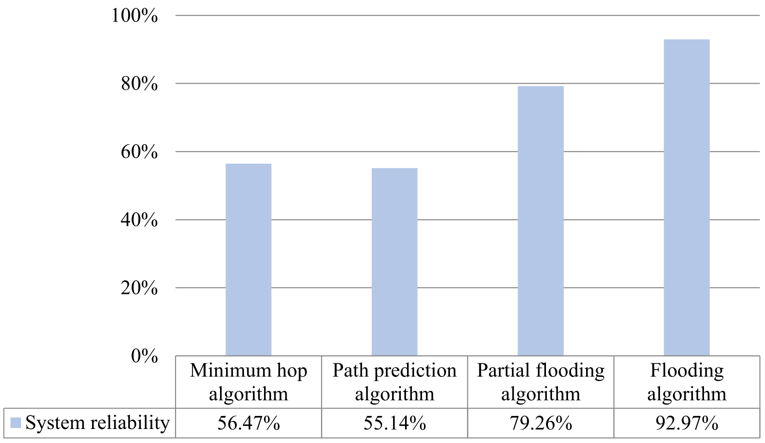


Fig. 8 Comparison of system reliability of different offloading strategies

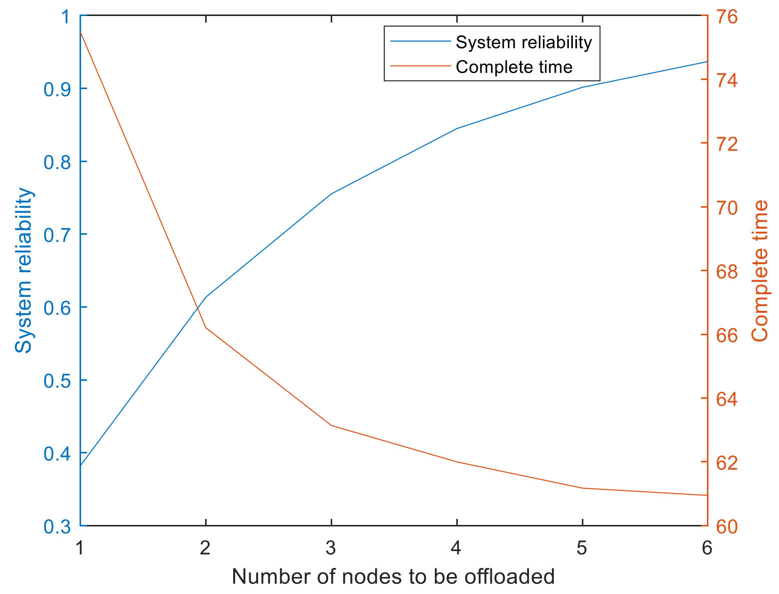
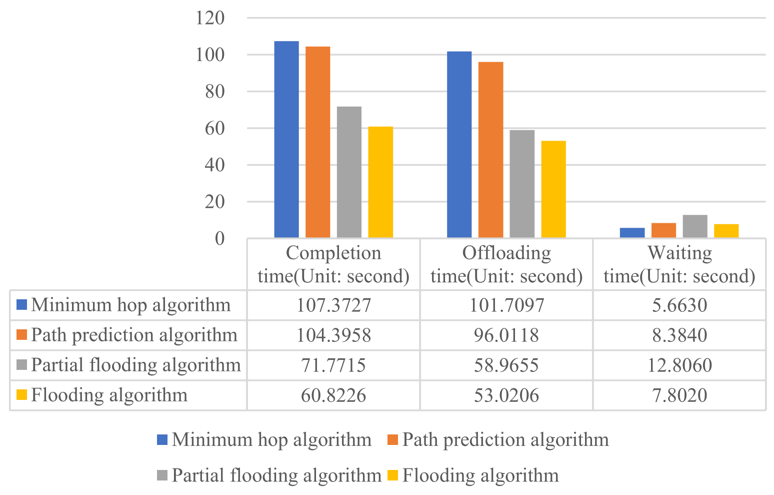


Fig. 9 Resource utilization and complete time comparison

partial offloading algorithm based on the traditional algorithm has different degrees of improvement compared with the traditional computing offloading algorithm.

Specifically, as for the completion time of the task, since the accuracy based on the prediction algorithm is affected by the random motion of the vehicle, the prediction result may be inaccurate with the actual situation. This will result in the selected node not being the actual optimal node. Therefore, the partial flooding algorithm is improved by 35.04% compared with the path prediction algorithm, and the minimum hopping algorithm is the shortest distance of the current time. In a dynamic environment, the distance between nodes can vary greatly. And the performance of the selected node may not be globally optimal. Therefore, the performance comparison and minimum hopping algorithm proposed in this paper improve the performance by 35.82%. For the offloading time of the task, a group of nodes with strong communication stability is selected as the candidate offloading node based on the partial flooding algorithm. It reduces the risk of the communication link disconnected when the task is transmitted. Compared with the algorithm such as MH and path prediction, the performance of the partial flooding algorithm is improved by 46.52% and 44.06%, respectively, which ensures the stability of the V2V communication link. The time at



	Completion time(Unit: second)	Offloading time(Unit: second)	Waiting time(Unit: second)
🟦 Minimum hop algorithm	107.3727	101.7097	5.6630
🟧 Path prediction algorithm	104.3958	96.0118	8.3840
⬜ Partial flooding algorithm	71.7715	58.9655	12.8060
🟨 Flooding algorithm	60.8226	53.0206	7.8020

Fig. 10 Comparison of task completion performance of different offloading strategies

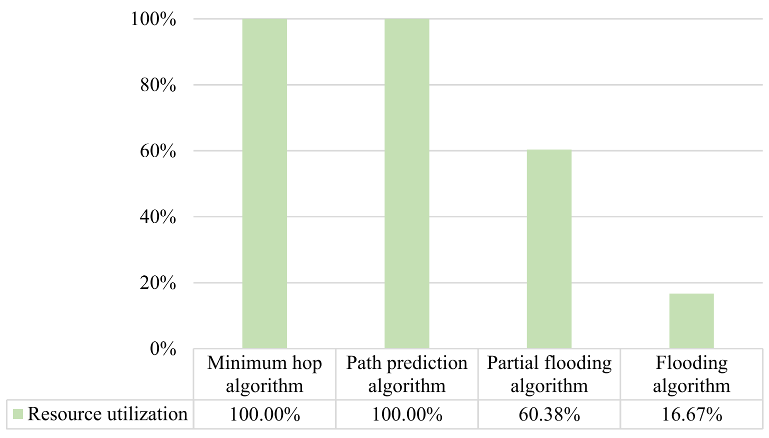


Fig. 11 Resource utilization comparison

which the task waits for the upload and download to start depends on the task generation time and the task execution time where the node at the completion time is calculated on the proxy resource. There is not much difference in the algorithm.

By comparing these three performance indicators, it prove that the partial offloading algorithm can effectively improve the offloading performance. At the same time, the experimental results illustrate the performance difference between the partial offloading algorithm and the flooding algorithm is not large, and the advantage of the completion time of the flooding offloading algorithm is retained.

In multi-user multi-tasking systems, resource utilization is one of the important factors affecting system offload performance. Compared with the flooding offloading strategy, we only select some better-performing nodes as the target nodes for offloading, which improves the system reliability, further, optimizes the effectiveness of resource utilization.

As shown in Fig. 11, it is verified by multiple experiments that the resource utilization rate of the partial flooding algorithm corresponds to the case where the number of selected offloading points in Fig. 12 is 2 or 3. In comparison with the flooding offloading strategy, while ensuring the completion time of the task, the number of selected nodes is reduced, which saves the computing resources and communication resources required in the offloading process. The algorithm proposed in this paper improves the resource availability of the system, and the offloading nodes that execute the task will have a greater probability of being the final adopted node. Since

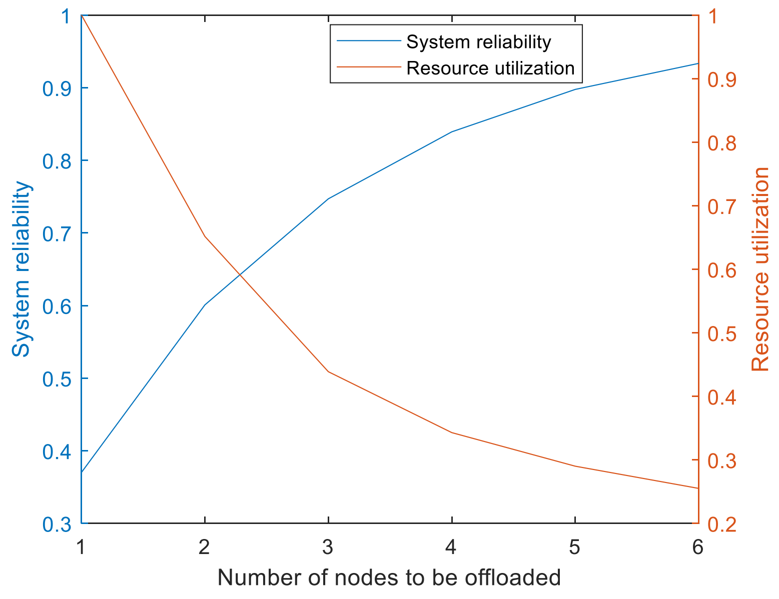


Fig. 12 Resource utilization and system reliability comparison

the traditional algorithm only occupies one computing resource, although its resource utilization rate is higher than the partial flooding algorithm, the system reliability is much lower than the partial flooding algorithm, as shown in the figure below.

From the analysis in Fig. 12. Although the partial flooding algorithm reduces the reliability of the system, its resource utilization rate is 43.71% higher than that of the flooding algorithm. For the traditional algorithm (MH algorithm and path prediction algorithm) that only offloads to one node, the partial flooding algorithm establishes a parallel system, which has greatly improved the system reliability. That is to say, the partial flooding strategy based on link reliability and node computing power balances the two performance parameters of resource utilization and reliability of the system. At this point, the offloading performance reaches an equalization, which not only improves the reliability of the system, but also ensures that the task is completed within the deadline, and reduces the resource utilization of the system. From the user's point of view, the user terminal can receive the result of the application execution more quickly, and improve the QoS. In addition, from the perspective of the proxy resource, the waste of resources is reduced, and the execution cost is greatly reduced.

Conclusion

This paper studies the computational offloading strategy in vehicle self-organizing network and optimizes the traditional computing offloading strategy. Compared with the algorithm only offloading to one node, the partial flooding algorithm proposed in this paper improves the reliability of the system. At the same time, the influence of the high-speed motion of the vehicle node on the execution and offloading is reduced. It also reduces task completion time and enhances the user experience. Different from the flooding algorithm, since the partial flooding algorithm considers the performance parameters such as system reliability and node computing ability, only the appropriate number of target nodes are selected for execution and offloading. Therefore, the partial flooding algorithm consumes less computational and communication resources than the flooding algorithm during the offloading process, which greatly reduces the waste of system resources. The results show that the partial flooding algorithm proposed in this paper can guarantee the delay-sensitive task to complete in the deadline, improve resource utilization, balance the system performance and resource availability, and effectively improve the offloading performance of the whole system.

In future research, we will consider multi-target offloading strategies with infrastructure to reduce the execution energy consumption of task execution, maximize system offloading performance, and minimize computation completion time to optimize overall offloading overhead.

Acknowledgements
There is no acknowledgement.

Authors' contributions
BL, ZP, and PH are the main researcher, MH, MA and GJ are coauthor. All authors read and approved the final manuscript.

Authors' information
Bo Li is a Professor at Yunnan University, received his Ph.D. degree in Information and Communication Engineering from Huazhong University of Science and Technology, China, in 2005.
Ziyi Peng and Peng Hou are master students under the advising of Bo Li.
Min He is an Associate Professor at Yunnan University, received her Ph.D. degree in Computer Science from University of Electronics Science and Technology of China, in 2006.
Marco Anisetti is an Associate Professor at the Università degli Studi di Milano. He received his Ph.D. degree from Ottawa University in Computer Science in 2006.
Gwanggil Jeon is an Assistant Professor with the Department of Embedded Systems Engineering, Incheon National University, Incheon, Korea. He received his Ph.D. degree from the Department of Electronics and Computer Engineering, Hanyang University, Seoul, Korea, in 2008.

Funding
This work has been institutionally supported by the National Natural Science Foundation of China with grant number 61562092 and the Science and Technology Plan of Yunnan Province with grant number 2014AB016.

Availability of data and materials
There is no supporting data available.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]School of Information Science and Engineering, Yunnan University, Kunming 650500, Yunnan, China. [2]Dipartimento di Informatica (DI), Università degli Studi di Milano, Via Celoria 18, Milano 20133, (MI), Italy. [3]School of Electronic Engineering, Xidian University, Xi'an 710071, China. [4]Department of Embedded Systems Engineering, Incheon National University, Incheon 22012, Korea.

Received: 25 September 2019 Accepted: 28 November 2019
Published online: 16 December 2019

References

1. Ahmed B, Malik AW, Hafeez T, Ahmed N. Services and simulation frameworks for vehicular cloud computing: a contemporary survey. EURASIP J Wirel Commun Netw. 2019;2019(1):4.
2. Aliyu A, Abdullah AH, Kaiwartya O, Cao Y, Usman MJ, Kumar S, Lobiyal DK, Raw RS. Cloud computing in vanets: Architecture, taxonomy, and challenges. Iete Tech Rev. 2018;5(5):523–547.
3. Wu H-T, Horng G-J. Establishing an intelligent transportation system with a network security mechanism in an internet of vehicle environment. IEEE Access. 2017;5:19239–19247.
4. Arif M, Wang G, Bhuiyan MZA, Wang T, Chen J. A survey on security attacks in vanets: Communication, applications and challenges. Veh Commun. 2019100179. https://doi.org/10.1016/j.vehcom.2019.100179.
5. Abbas F, Fan P, Khan Z. A novel low-latency v2v resource allocation scheme based on cellular v2x communications. IEEE Trans Intell Transp Syst. 2018;20(6):2185–2197.
6. Huang C-M, Wu Z-Y, Lin S-Y. The mobile edge computing (mec)-based vanet data offloading using the staying-time-oriented k-hop away offloading agent. In: 2019 International Conference on Information Networking (ICOIN). IEEE; 2019. p. 357–62. https://doi.org/10.1109/icoin.2019.8718188.

7. Lin X, Li J, Yang W, Wu J, Zong Z, Wang X. Vehicle-to-cloudlet: Game-based computation demand response for mobile edge computing through vehicles. In: 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring). IEEE; 2019. p. 1–6. https://doi.org/10.1109/vtcspring.2019.8746335.
8. Naresh M, Raje A, Varsha K. Link prediction algorithm for efficient routing in vanets. IEEE; 2019. p. 1156–1161. https://doi.org/10.1109/iccmc.2019.8819723.
9. Kim W, Kang CM, Son YS, Lee S-H, Chung CC. Vehicle path prediction using yaw acceleration for adaptive cruise control. IEEE Trans Intell Transp Syst. 2018;19(12):3818–3829.
10. Ye M, Guan L, Quddus M. Mpbrp-mobility prediction based routing protocol in vanets. IEEE; 2019. p. 1–7. https://doi.org/10.1109/commnet.2019.8742389.
11. Lei T, Wang S, Li J, Yang F. A cooperative route choice approach via virtual vehicle in internet of vehicles. Springer; 2016. p. 194–205. https://doi.org/10.1007/978-3-319-51969-2_16.
12. He X, Zhao H, Yu BJ, Zhu J. Design and implementation of minimum hop routing algorithm with reliability assurance for wsn. In: Advanced Materials Research, vol. 268. Trans Tech Publ; 2011. p. 975–980. https://doi.org/10.4028/www.scientific.net/amr.268-270.975.
13. Fan SP, Bao-Ying MA, Gao CG, Yao NM, University MN. Minimum hop routing algorithm for clustering wireless sensor networks. J Chin Comput Syst. 2014;35(8):1775–1779.
14. Le HQ, Al-Shatri H, Klein A. Efficient resource allocation in mobile-edge computation offloading: Completion time minimization. IEEE; 2017. p. 2513–2517. https://doi.org/10.1109/isit.2017.8006982.
15. Toh C-K, Bunchua S. Performance evaluation of flooding-based and associativity-based ad hoc mobile multicast routing protocols. IEEE; 2000. p. 1274–1279. https://doi.org/10.1109/wcnc.2000.904815.
16. Li J, Blake C, De Couto DSJ, et al. Capacity of ad hoc wireless networks. In: Proceedings of the 7th annual international on mobile computing and networking. New York: ACM; 2001. p. 61–69.
17. Eiza MH, Ni Q. An evolving graph-based reliable routing scheme for vanets. IEEE Trans Veh Technol. 2013;62(4):1493–1504.
18. Chaturvedi SK. Network reliability: measures and evaluation. New York: John Wiley & Sons; 2016.
19. Khanna G, Chaturvedi S, Soh S. Reliability evaluation of mobile ad hoc networks by considering link expiration time and border time. Int J Syst Assur Eng Manag. 2019;10(3):399–415.
20. Ünlü B, Özceylan B, Baykal B. Ipbm: an energy efficient reliable interference-aware periodic broadcast messaging protocol for manets. Wirel Netw. 2019;25(5):2769–2787.
21. Ylianttila M, Mäkelä J, Pahlavan K. Analysis of handoff in a location-aware vertical multi-access network. Comput Netw. 2005;47(2):185–201.
22. Su W, Lee S-J, Gerla M. Mobility prediction and routing in ad hoc wireless networks. Int J Netw Manag. 2001;11(1):3–30.
23. Zhang Y, Zhao J, Cao G. Roadcast: a popularity aware content sharing scheme in vanets. ACM SIGMOBILE Mob Comput Commun Rev. 2010;13(4):1–14.
24. Peng S, Chai R, Chen Q, Qin Y. Minimum end-to-end transmission delay based routing algorithm for vanets. IEEE; 2017. p. 176–181. https://doi.org/10.1109/icait.2017.8388910.
25. Hussain S, Keung J, Khan AA, Ahmad A, Cuomo S, Piccialli F, Jeon G, Akhunzada A. Implications of deep learning for the automation of design patterns organization. J Parallel Distrib Comput. 2018;117:256–266.
26. Chianese A, Marulli F, Piccialli F. Cultural heritage and social pulse: A semantic approach for ch sensitivity discovery in social media data. IEEE; 2016. p. 459–464. https://doi.org/10.1109/icsc.2016.50.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.