**Distributed Tasks Allocation in Vehicular Fog Computing**

**Abstract:** Data offloading is one of the most challenging tasks to perform due to unique characteristics of vehicular networks, including heterogeneous computation capabilities of servers, high mobility of vehicles and uneven distribution of workloads. Hence, we put forward a feasible solution that enables offloading for real-time traffic management in fog-based IoV systems, aiming to maximize the probability of task completion. As a complementation of cloud computing, Fog computing (FC) is an emerging paradigm to migrating computational resources to network edges, aiming at better supporting computation-intensive services with requirements of low latency or real-time processing. First, we construct a distributed city-wide traffic management system, in which vehicles with computing and storage capabilities can be utilized as fog nodes.

# 1.INTRODUCTION

With the emergence of computation intensive applications, a huge number of end-user devices are in urgent need of computational capabilities. Up-to-now, conventional cloud-based solutions are challenged by the bottleneck of network bandwidth, high delay and low quality of experience (QoE).

With the emergence of ever-growing advanced vehicular applications, the challenges to meet the demands from both communication and computation are increasingly prominent. Without powerful communication and computational support, various vehicular applications and services will still stay in the concept phase and cannot be put into practice in the daily life. Thus, solving this problem is of great importance. The existing solutions, such as cellular networks, roadside units (RSUs), andmobile cloud computing, are far from perfect because they highly depend on and bear the cost of additional infrastructure deployment. Given tremendous number of vehicles in urban areas, putting these underutilized vehicular resources into use offers great opportunity and value. Therefore, we conceive the idea of utilizing vehicles as the infrastructures for communication and computation, named vehicular fog computing (VFC), which is an architecture that uti- lizes a collaborative multitude ofend-user clients or near-user edge devices to carry out communication and computation, based on better utilization of individual communication and computational resources of each vehicle.

It is worth noting that modern vehicles can be connected to the Internet so as to serve as the computation units of Fog Computing. The network of wireless access technology enabled vehicles involving Internet and other heterogeneous networks is called the Internet-of-Vehicle (IoV). Recent researches have regarded the IoV as a promising FC platform due to the following three reasons. First, the large number and wide distribution of vehicles are convenient for users to access the computation units. Secondly, vehicles can provide abundant computation and storage resources by using their smart on-board units (OBU). Last but not least, different from other computing units, vehicles can provide enough energy for themselves, instead from acquiring from the power grid. With the FC enabled IoV (FC-IoV), users can offload their tasks to fog nodes for computation or storage. The service provision is supposed to be well designed according to the requirements of these tasks and the capability of the edge nodes. However, through the node selection, users’ quality of experience (QoE) may be greatly affected by the availability, reputation, and communication cost of the selectable nodes.



Fig.1. The hierarchical architecture （需修改）

FC is an extension of cloud computing. By pushing significant storage, control, manage- ment, and communication mechanisms onto the network edge or user equipment, FC alleviates the pressure of the core network. According- ly, we consider a novel SDFC-VeNET architec- ture in this article. Based on the SDFC-VeNET architecture, mobility management and resource allocation are discussed.

Highly dynamic mobility: The key challenge is that the topology of a vehicular network is highly dynamic due to vehicle mobility, which makes it difficult for the control plane to maintain the status of vehicles and hence increases task management overhead.

Challenge posed by vehicle mobility is mainly because the returned data packet will not be able to find the client vehicle since the location of it has changed.

Assumption:

* We consider that each vehicle is equipped with certain sensing, on-board data storage, computing facilities, and wireless communications, e.g., V2V and V2I.
* We also consider that each vehicle is allowed to transfer content data to a nearby vehicle and the infrastructure at the same time by using V2V and V2I communications.
* There are many vehicular applications generated by vehicle, such as real-time situational awareness, high-definition local maps, see-through for passing and etc. The process of an application can be decomposed into a set of tasks. We assume that different tasks of the application can be distributed to different computing nodes to be executed in a parallel and independent manner.
* In this paper, task is considered to be the basic unit to allocate. In other words, task cannot be divided into sub-task from the perspective of tasks allocation.
* Each vehicle can generate more than one task at the same time. Tasks from one vehicle can allocate in different fog nodes which in the communication range. However, each task must be allocated to one and exactly one fog node.
* The tasks may require different amount of resources for executions.
* The number of tasks that can be executed at each mobile device is not a fixed value, depending on its resource availability and the resource requirements of the tasks allocated to it.
* Fog node may have different preferences to the tasks allocated to it, because of different computation complexities, different data size of task and etc.

For the same task, allocated to different fog node may have different execution time, because of different computation capacity and etc.

# 2. RELATED WORK

There are many studies in the recent literature dealing with decision making for task migration in cloud computing, Mobile Cloud Computing (MCC), Vehicular Cloud Computing (VCC) , etc. This section, we will provide the most prominent research works related to our work.

# 3. SYSTEM MODEL

In this Section, we present the detailed model of the fog-based vehicular network architecture that is the basis for our DTA(Distributed Tasks Allocation) formulation.Then, we use an motivational example to better illustrate the presented system as well as to reveal the challenges of designing an efficient scheduling policy.

### 3.1 System Model

As show in Fig.2, there are two layers in the proposed network architecture, including Client layer and Fog Layer(consist of Mobile Fog layer and Stationary Fog layer). All the layers have computing and storage capabilities that differ from each other. In the following, we elaborate the distinct characteristics of each layer as well as the communication modes between different layers.



Fig.2. The vehicle-fog-cloud architecture(需修改)

(1) Client layer: The client layer refers to vehicles equipped with certain sensing, on-board data storage, computing facilities, and wireless communications, e.g., V2V and V2I. There are many vehicular applications generated by vehicle, such as real-time situational awareness, high-definition local maps, see-through for passing and etc. The process of an application can be decomposed into a set of tasks. For example, AR-based driving assistance consists of tasks such as video streaming and object recognition. We assume that different tasks of the application can be distributed to different computing nodes to be executed in a parallel and independent manner.

(2)Fog Layer: The fog layer is composed of lightweight network devices at the edge of network, such as vehicles, smart buildings and infrastructure which have computing, storage, and communication capabilities(eg., RSU, BS, etc.). If fog nodes are available, client vehicles can directly upload tasks to fog nodes (instead of cloud nodes) for processing to shorten the response time. In this paper, we assume that both parked and moving vehicles can be utilized to form fog nodes. We consider two types of fog node:

Mobile fog node: The mobile fog node are the mobile vehicles which drive in cities randomly (i.e. cars and taxis) or routinely (i.e. buses). They will receive computation tasks when passing through the communication range of client vehicles. We assume that vehicles are equipped with mmwave communication interfaces for task offloading. Therefore, mobile fog has a high transmission rate.

Stationary fog node: The computing nodes co-located with cellular base stations(BS), Wi-Fi access points, road side units(RSU), smart buildings, parking vehicles or any other stationary infrastructure. Generally, Stationary fog nodes have more powerful computation capability than mobile nodes.

Fig.2 illustrates the process of task allocation in model, which includes the following four procedures.

* First, fog nodes need to figure out which client vehicles are located in its communication range and which tasks need to process. So, all the client vehicles broadcast one-hop beacon message over DSRC periodically, including GPS position, acceleration, velocity, directions, data size of tasks and their requirements to fog node in the system.
* Second, fog nodes collects the information from client vehicles in the communication range. And then, they will analyse the information from client vehicles and execute the DAT strategy to decide which tasks to process and the task processing order periodically, e.g., every other 100ms.
* Third, the message containing the scheduling policy is delivered from fog nodes to client vehicles via V2I and V2V communication, respectively. Once receiving the scheduling decision, the client vehicles will select a fog node for each task based on the scheduling policy and offload the data items accordingly via either V2I or V2V communications.
* Fourth, the fog nodes are main processors in our system to handle tasks offloaded by vehicles. Once the computation task is completed, fog node will return the results directly.

In our system, our main focus is to balance tasks loads among the fog nodes to maximize the completion probability of the tasks under a series of conditions such as satisfying the response time. The advantages of our two-layer system model compared with the traditional traffic management scheme are as follows:

* Our model is based on a decentralized network structure, and the data process can be managed independently in each region;
* We use the fog nodes to offload traffic for cloud layer, which largely reduce its burdens;
* Our model can largely reduce response delay, because fog nodes are close to terminals.

### 3.2 Motivational Example

To have a better understanding of the problem we study, Table.I shows an example of the problem to illustrate the main idea of technique proposed in this paper.

Table I: Attributes of Tasks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Vehicles | Tasks | Size  (size unit) | Start time | Slack Time | required number of CPU cycles |
|  |  | 1 | 2 | 13 | 5 |
|  | 2 | 1 | 8 | 2 |
|  |  | 2 | 0 | 16 | 4 |
|  | 1 | 0 | 12 | 7 |
|  | 1 | 2 | 5 | 1 |
|  |  | 2 | 1 | 11 | 2 |
|  |  | 1 | 0 | 20 | 1 |
|  | 2 | 2 | 8 | 1 |

Table II: Attributes of Fog Node

|  |  |  |  |
| --- | --- | --- | --- |
| Fog Node |  |  |  |
| Memory Capacity (size unit) | 4 | 2 | 10 |
| Computing capacity  (time unit/CPU cycle) | 1 | 3 | 2 |
| Wait Delay (time unit) | 0 | 1 | 2 |
| Penalty Delay (time unit) | 4 | | |

Table III: Data Rate Matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | 1 | 0 | 2 |
|  | 2 | 1 | 0 |
|  | 0 | 1 | 0 |
|  | 0 | 0 | 1 |

Table IV: Network Availability Matrix

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | | |  | | | |  | | | |
| 0 | 5 | 10 | 15 | 0 | 5 | 10 | 15 | 0 | 5 | 10 | 15 |
|  | 100% | 70% | 60% | 50% | 0% | 0% | 0% | 0% | 100% | 70% | 60% | 50% |
|  | 100% | 70% | 60% | 50% | 100% | 70% | 60% | 50% | 0% | 0% | 0% | 0% |
|  | 0% | 0% | 0% | 0% | 100% | 70% | 60% | 50% | 0% | 0% | 0% | 0% |
|  | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 100% | 70% | 60% | 50% |

Table V: Scheduling Solution Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Probability of task Completed |
| ILP | ，, |  | , | 75% |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |

# 4. PROBLEM FORMULATION

In this section, we will present the formulation of the task allocation problem, named as Distributed Tasks Allocation (DTA). In order to understand the problem more clearly, we first define some notations in TableⅠused throughout this paper.

TABLE **Ⅰ** Notation and definitions

|  |  |
| --- | --- |
| Notation | Description |
|  | The set of client vehicles. |
|  | The -th client vehicle in . |
|  |  |
| *A* | The set of fog nodes that are willing to participate in task execution. |
|  | The -th fog node in . |
|  | The memory capacity of fog node , that is, the maximum tolerable data size. |
|  | The computing capacity of fog node (i.e.,unit times per CPU cycle). |
|  | The transmission rate of fog node . |
|  | The communication range of fog node . |
|  |  |
|  | The set of mobile tasks generated by client vehicles. |
|  | The -th mobile task in . |
|  | The data size of task . |
|  | The generation time of task . |
|  | The deadline of task . |
|  | the total number of CPU cycles required to complete task . |
|  |  |
|  | The set of tasks can be processed at fog node . |
|  | The set of fog candidates available for task . |
|  | The set of tasks generated by client vehicle . |
|  |  |
|  | The distance between client vehicle and fog node . |
|  | The last dwell time of task on the fog node . |
|  | A binary variable, indicate whether task is assigned to execute at the position of fog node . |
|  |  |
|  | The round trip time between a client vehicle and a fog node. |
|  | The Initial waitting time of anyone task allocated in fog node . |
|  | The process time of fog node for the task . |
|  | The transmission time of task allocated in fog node . |
|  | The waitting time of task allocated in fog node . |
|  | The total response delay of a task allocated in fog node . |
|  |  |
|  | The duration of a time slot, which is sufficiently small. |
|  | The maximum number of time slots allowed tasks to execute in fog node. |
|  | The penalty delay of sending the computation result to the end user. |
|  |  |
|  |  |
|  |  |

***4.1 Related Terms***

We define as the set of client vehicles. Consider a set of mobile tasks generated by the client vehicles, denoted by , to be allocated to fog nodes for executions. Each task is associated with a four-tuple ***(, )***.

(1) The attribute represents the data size of task .

(2) The attribute represents the generation time of task .(任务生成时间可不需要)

(3) The attribute represents the deadline of task . Therefore, the maximum service interval of task is [**,** ].

(4) The attribute denotes the required number of CPU cycles for completing task .

Suppose that is the set of resource-rich fog nodes in system that are willing to participate in task execution for client vehicles. As each fog node is also with limited resources, which we use a three-tuple (**, ,** ) to represent. Specifically, denotes the memory capacity of , attribute denotes the computing capacity in CPU of (unit time per cycle), and denotes the transmission rate of fog node .

It is noted that, the completion time of a task executed in different fog nodes may not be the same. This is reasonable, as different fog node may have different computation capacity.

For simplification, we assume that client vehicle and fog node can support the delivery of tasks when there is a communication link between the two nodes. Due to the mobility of the vehicles and the communication distance limitation of the DSRC, it is impossible to maintain communication all the time. We denote the distance between client vehicle and fog node as . When the distance is no more than the communication range(denoted by ), i.e., there is a communication link between the two nodes and they can deliver tasks and information with each other while the link exists.

To ease presentation, we assume that , where denotes the set of mobile tasks generated by client vehicle . The set of client vehicle

Therefore, we define as the set of tasks that the fog node receives from the client vehicles within its communication range. Each have a attribute indicating the last dwell time on the fog node .

For each fog node, there is a list of tasks to be processed at , denoted as . Particularly, the First-Come-First-Served (FCFS) model is adopted to simulate the computation procedure of the fog node. That is, the task with earlier generation time has higher priority to be executed by the FC server. The queuing tasks have to tolerate pending delay until all the tasks ahead are completed.

***4.2 Response delay***

Before giving the definition of the response delay, we first introduce three related concepts as follows.

***Transmission time:*** Given limited bandwidth, the transmission time depends on the size of data to be transmitted. For each task , we define as the corresponding size of data to be transmitted. The transmission data rate of the link between the client vehicle and the selected fog node is denoted as . We calculate the transmission delay as follows:

(1)

***Process time:*** We assume that the process time of each task is only related to the number of CPU cycles and the computation capacity of fog nodes, defined as:

, (2)

***Waiting time:*** When is assigned to the fog node , the waiting delay equals the time cost by completing all the waiting tasks which has higher priority than . Therefore, the waiting time is formulated as follows:

(3)

***Response delay:*** The objective of our work is to reduce the response time for the traffic management system. The total response delay can be obtained by

, (4)

where is the travel time between a fog node and a client vehicle.

***Allocated profit:*** we propose to maximize the total profits gained from allocating tasks to the fog nodes under the constraints of computation and storage capacity, where profit denoted as:



where .

***4.3 Constraints***

Without loss of generality, we define a binary variable to indicate whether task is assigned to fog node .

The allocated problem must follow the following rules:

* Each task is supposed to be the basic unit for task allocation. So, it can only allocate to no more than one fog node within the communication range.

**(1)**

**(2)**

* For each position of a fog node, only can be assigned one task.

**(3)**

* The total size of tasks assigned to one fog node cannot exceed its memory capacity. According to Table 1, defined as the capacity of fog node , and is the size of task . We formulate the memory capacity constraint as below.

**(4)**

* The maximum tolerable response time is determined by the type of application. According to the table I, we know the last service time of is . To guarantee that the task would be completed in time, a service constraint is given as follows:

**(5)**

The objective of DTA is to search the optimal task assignment X= () in a distributed way, to maximize the probability of task completion, which is expressed as follows:

**Max:**



# Algorithm

To obtain the optimal solution, we modify a heuristic greedy algorithm which is intended for 0-1 knapsack problem.

Firstly, we will introduce the detailed process of each allocation round, which includes making assignment scheme at each fog node, evaluating the profit decision with the collaboration of client vehicles, and determining the final allocation plan by client vehicles. Then, we use a fog node as an example to illustrate how the proposed mechanism works.

The allocation decision is made locally at each fog node, so we will use a fog node as an example to introduce the detail of task allocation scheduling algorithm. There are some tasks generated by the client vehicles within the communication range of fog node that need to be assigned for executing, indicated by .

To ease presentation, we define a binary variable . If , task is allocated to fog node , otherwise, . Initially, for . According to Eq.(4) and Eq.(5), can be assigned to fog node , only if it satisfies the following two conditions:

(1) The remaining storage capacity of the fog node is not less than the size of the task .

(2) The final complete time is no more than the minimum staying time of task , i.e., min{}.

Since the task complete time is related to the execution order, we need to sort the tasks in according to increasing order of their urgency, i.e., . To ease understanding, we use to denote the profit, if is selected as one of the excuting candidates of .

The detail of the Task Allocation Decision-making (TAD) Algorithm is presented in Algorithm 1, and the main objective is to selecte an allocation scheme that satisfies the above two conditions and maximizes the total profit. All the fog node in follow the same procedure as .

|  |  |
| --- | --- |
| Algorithm 1 Task Allocation Decision-making (TAD)Algorithm in fog node | |
| Input:  Output:  Step:  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | The attribute of fog node : ;  The attribute of each task : ;  : Assignable task candidate set of fog node ;  : The maximum tolerable delay of the task executed by the fog node ;  : The profit of the fog node processing tasks .  : the begin time of Schedule.  Allocation scheme and the completion time ’  //Step 1 initialize  Sort the set in ascending order according to the urgency of each task (i.e., the end time of task);  Set ;  Set ;  //Step 2:  For do  For do  For do  If then  ;  ;  End if  Eles  Calculate the transmission time  Calculate the processing time ;  ;  ;  If |=0 then  If )and () then  If then  ;  ;  End if  End if  ElseIf )and () then  If then  ;  ;  End if  End if  End else  End for  End for  End for  //Step 3:  The maximal profit is and the allocation is stored in .  Set ;  For do  Calculate the processing time ;  If is the first task in then  Calculate the transmission time  ;  End if  Set ;; *;*  End for |



With the above Algorithm, each fog node selects some appropriate tasks as the candidates to executing. However, it is noted that a task may be assigned to multiple fog nodes, which cannot satisfy the rule defined in Eq. (1). Hence, for each task, it is still necessary to determine the final fog node locally.

For each task , according to {}, there are three case:

Case (1): represents was not selected by any fog node. In this case, we set .

Case (2): represents was only selected by one fog node. As such, we have .

Case (3): represents was selected by multiple fog nodes. Due to each task can be executed only at a single fog node, we need to choose the final fog node to execute . Among all the possible fog nodes in {}, we select the one with the largest completion time as the final fog node, e.g., , where

.

That is, task will execute by fog node , i.e.,.

After those two stages, we can obtain the selection strategy , which is the optimized result.

# Experiment

In order to evaluate the performance of the proposed algorithm, we simulate the model based on the architecture and implement the heuristic method using MATLAB. In this section, we will present all the adopted scenarios and the simulation parameters in Section 6.1. Secondly, a discussion of the numerical results will be presented in Section 6.2.

***6.1 Experimental Setup***

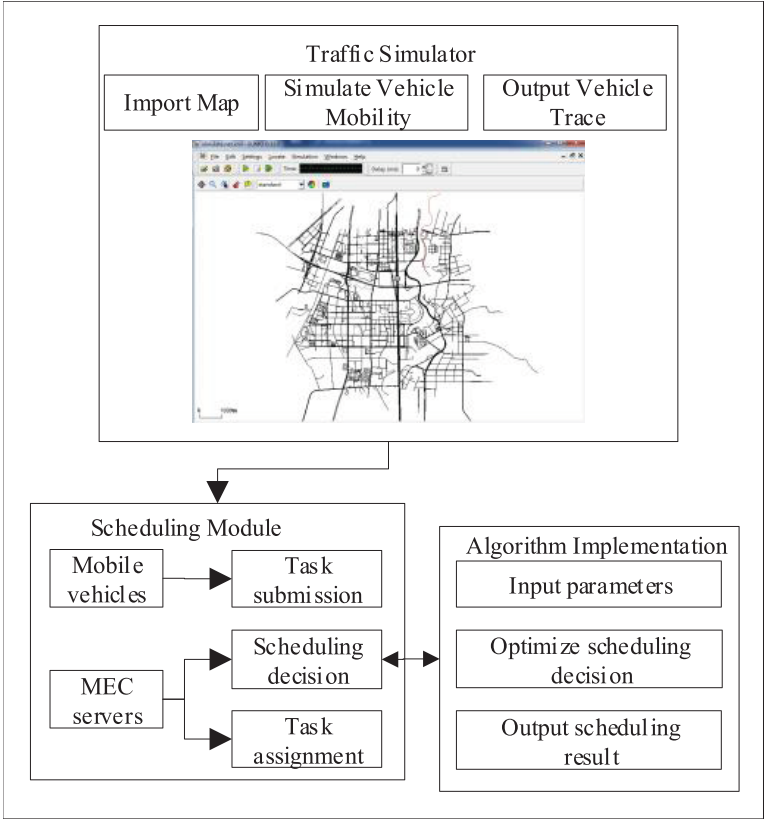
The simulation model integrates a traffic simulator with real-world map, a scheduling module and an optimization module. Specifically, a real-world map, extracted from the core area of High-tech Zone in Chengdu, China, is imported to establish the traffic scenario in SUMO. SUMO is adopted to simulate the vehicle mobility and generate the vehicular traces. Then a scheduling module is implemented at each FC server for managing the submissions of tasks, optimizing the scheduling decision of task assignment and making the task assignment among MEC/Cloud servers. Particularly, the scheduling decision is determined by the optimization module, which runs the algorithm based on the parameter input from scheduling module and sends back the scheduling decisions. The relationship among these three modules is illustrated in Fig.3.

In the default setting, there are 10 SFC servers, which are evenly distributed along the road. To differentiate the computing and storage capabilities, the processing rate of each SFC server is randomly generated from the interval [32, 38] CPU cycle per time unit and the . The arrival pattern of vehicles in the service range of each Stationary FC server follows the Poisson process and the arrival rate is denoted by , which is randomly generated from the interval [1500, 2500] veh/h. To differentiate the computing capability, the processing rate pi of each MEC server is randomly generated from the interval [32, 38] resource unit per time unit. For each vehicle, the required computation resource of tasks is generated from the interval [55, 65] resource units. Particularly, the penalty delay is set to 60 time units. Unless stated otherwise, the simulation is conducted under the default setting.

The communication rate from RSU to cloud is set to 15 Mbps [3], and 1.5 Gbps to mobile fog [5], and 80 Mbps to fixed fog. The computation capability of fog nodes is 20×10 cycles/s [10]. We consider an area with 10 vehicles and 2 fixed fog nodes. Then, the maximum computation capacity of mobile fog layer is 200 × 108 cycles/s. Consider that the fixed fog node such as base station has much more computation resources than mobile vehicles, we assume the available computation resources of a fixed fog node is 10 times than that of a mobile vehicle. Thus, the maximum computation capacity of fixed fog layer is 400×108 cycles/s. The computation capacity of cloud is unlimited, but it usually can assign the speed of 100×108 cycles/s [11] for each user or regular task. Assume that |A| is not larger than 40. Then the maximum computation capacity can be set as 4000×108. For each task, the intended transmission rate for different layers are uniformly generated from the range [1, 50] Mbps, [1, 20] Mbps and [1, 10] Mbps, respectively. The demanding computation capabilities for different layers are uniformly distributed in the range of [1, 15]×108 cycles/s, [1, 20]×108 cycles/s and [1, 200] × 108 cycles/s, respectively. The unit price p0 of computation resource is set to 0.1 [12]. The size of computation task is uniformly distributed in [1, 50] Gcycles. We run each sample for 50 times to compute the average value of all samples.

Table II. Default Setting

|  |  |  |
| --- | --- | --- |
| Notation | value | Description |
| |SA| | 4 | The number of Stationary Fog Computing(SFC) servers. |
| |MA| | 10 | The number of Mobile Fog Computing(MFC) servers. |
| | V| | 100 | The number of client vehicle. |
| || | [1..5] | The number of tasks generated by each client vehicle. |
| size | [10..25] | The range of data size of each task. |
| r | [10..25] | The range of number of CPU cycles requested by each task. |
|  | [1..300]s | The range of start time of each task. |
| dt | [50..150]s | The range of task vaild period. |
| Ss | [50..100] | The range of storage capacity of SFC servers. |
| Sm | [80..150] | The range of storage capacity of MFC servers. |
| Rs | s/cycle | The range of computation capability of SFC servers. |
| Rm | s/cycle | The range of computation capability of MFC servers. |
| Ls | 1000m | The communication range of V2I. |
| Lm | 500m | The communication range of V2V. |
| Cs | 20 unit size/s | The transmission rate of V2I. |
| Cm | 10 unit size/s | The transmission rate of V2V. |
|  | [1500, 2500] veh/h | Vehicle arrival rate. |
|  |  |  |
|  |  |  |
|  |  |  |



(参考)