[1] * Wenjuan Shao

A Novel SDN-Based Architecture of Task Offloading in Mobile Ad-Hoc Cloud

Journal of Electrical Systems

Abstract: - As the core function of mobile Ad-hoc cloud, task offloading has always been a research hotspot of mobile cloud computing, and the construction, offloading decision, task division and scheduling strategies of mobile Ad-hoc cloud directly affect the performance of offloading. In this paper, we propose a new mobile Ad-hoc cloud architecture based on SDN solving the performance bottleneck of opportunistic task offloading. The proposed solution constructs terminal clusters in Ad-hoc cloud to provide users with offloading service, and makes offloading decision and task allocation through SDN controller to achieve the effectiveness of task allocation and the rationality of resource scheduling. Simulation results show that compared with traditional scheduling method, our method has the advantages in offloading ratio and execution efficiency, and can be more effectively used in dynamic mobile edge network environment, to improve network performance and user service satisfaction.

Keywords: SDN-based; Task offloading; Mobile Ad-hoc cloud.

I. INTRODUCTION

Traditional centralized cloud is usually located at a remote location and far away from users. Therefore, for delay sensitive mobile applications, such as high-quality video streaming, mobile games, etc., offloading tasks to the centralized cloud may not totally meet the requirements of UEs. In order to solve this problem and support delay sensitive and resource intensive mobile applications better, researchers have proposed the cloudlet[1-3], composed of a group of computing units well connected with the Internet, which can be used for nearby mobile devices. Due to its proximity to mobile users, it provides low latency and plenty of computing power for responsive applications. However, with the limited of large-scale deployment the cloudlet is still a huge challenge for ISP. Therefore, in this paper, a novel concept of constructing mobile Ad-hoc cloud using mobile terminals is proposed.

In the meanwhile, as a new emerging technology, SDN separates data forwarding layer from network control layer, improves the data transmission efficiency, brings flexibility and programmability to the network, and introduces in new services and new features. Most SDN architectures are designed for wired infrastructure, especially for applications in data centres. The main trend of SDN application scenarios in wireless mobile networks is access networks and wireless backhaul networks[4]. In 5G, SDN is introduced into D2D technology, which makes the wireless opportunistic network and wireless Ad-hoc network more reliable, thus the application scenario of wireless communication is furtherly expanded. For example, cellular network can build Ad-hoc network (for example, MANET or VANET) through D2D, offload local traffic, expand communication area, build emergency disaster recovery communication network, etc. At the same time, with SDN technology, the resource utilization ratio of cellular network is higher and the performance is more stable.

In this paper, we propose a new mobile Ad-hoc cloud architecture based on SDN solving the performance bottleneck of opportunistic task offloading. The proposed solution constructs terminal clusters in Ad-hoc cloud to provide users with offloading service, and makes offloading decision and task allocation through SDN controller to achieve the effectiveness of task allocation and the rationality of resource scheduling. Simulation results show that compared with traditional scheduling method, our method has some advantages in offloading ratio and execution efficiency, and can be more effectively used in dynamic mobile edge network environment, to improve network performance and user's satisfaction.

II. RELATED WORKS

As the core function of mobile Ad-hoc cloud, task offloading has recently been a research hotspot of mobile cloud computing, and the construction, offloading decision, task division and scheduling strategies of mobile Ad-hoc cloud directly affect the performance of offloading. Therefore, we will summarize the current research from the following aspects:

[1] School of Computer and Software, Nanjing Vocational University of Industry Technology, 210023, China.
*Corresponding author: Wenjuan Shao

(1) Construction of mobile Ad-hoc cloud

Due to mobility, the topology of mobile Ad-hoc cloud may change with time, so it is difficult to construct and manage it. At present, there are two ways to construct mobile Ad-hoc cloud:

One is the mobile Ad-hoc cloud without the management of the controller, such as Serendipity[5], device cloud[6], Hyrax[7], OMMC[8], etc., which is characterized by local coordination between peers to form a short-term mobile Ad-hoc cloud. Due to the lack of unified control, only unstable clusters can be provided.

The other is the Ad-hoc cloud managed by the controller (or agent) [9-14]. For example, Miluzzo et al. proposed mcloud, a framework prototype of mobile device cloud. Based on the interaction between the master and slave devices, the steps of topology discovery, formation, maintenance, and member release of mobile Ad-hoc cloud are described in detail[9]. The framework did not hierarchically cluster devices, which limits the scalability and is not applicable in the network with more users. Habak et al. proposed a femtocloud system prototype for mobile device sharing, so as to provide a dynamic, self-configured Ad-hoc cloud using mobile device [15]. They implemented the system in Android, but did not make further research on how to cluster.

Asghari et al. Proposed an Ad-hoc cloud framework[16], which is composed of selfish nodes and based on the centralized management of leaders. The difference between the Ad-hoc cloud framework and other controller management methods is that to minimize the total cost of task participants, each node assumes a leader role for a period. Muthana et al. discussed a two-stage Ad-hoc cloud constructing process under SDN controller in detail. The SDN controller had a global view of all the clouds and their services, and set up the service according to the user's request[17].

In the above, the connection between controller (agent) and member is single hop. Loke et al. Proposed a task offloading mechanism based on multi-level topology[12], which is used to effectively allocate tasks between mobile devices. Agent can not only connect devices directly through single hop, but also connect more devices through two or more hops, to expand the migration The scope of dynamic network improves the performance of task execution, but the author only verifies the performance of multi-level topology device cloud, and does not elaboration and analyze how to build the device cloud.

(2) Resource awareness of mobile Ad-hoc cloud

Because device resources vary with different hardware types and their related connection availability, it is very important to locate and perceive device resources in the cloud quickly and accurately. How to collect the distributed information of mobile devices and schedule tasks to appropriate devices or clouds is the main challenge to design mobile Ad-hoc cloud.

At present, the research methods of mobile Ad-hoc cloud resource awareness can be divided into two types:

First, resource discovery is handled by each node separately; in [18], the source node needs to query the resource information of other nodes in the network (such as processing ability, energy, task information), so five types of messages are defined, which are exchanged between nodes. After receiving the reply, a receiver is selected and the task is sent to it. In [8], devices use context manager to sense and collect context data, subtask information and network information of adjacent devices.

Second, resource discovery is assisted by centralized entities such as cellular networks. For example, for the computing intensive tasks of mobile Ad-hoc cloud, Yaqoob et al. proposed a heterogeneous sensing task allocation solution. By adopting the way of system centralized sensing, the workload information can maintain real-time consistency, considering CPU speed, number of CPU cores, workload, task size and other information, select devices and allocate tasks, and minimize execution time and energy consumption[19], However, this scheme lacks consideration of other resources (such as network bandwidth, etc.) that affect the execution effect and analysis of the way of terminal resource awareness. Lu et al. Proposed offline centralized and online distributed task allocation schemes to minimize the response time of tasks in order to solve the MCC task allocation problem in heterogeneous wireless networks by comprehensively considering various delays such as communication, processing and queuing [20]. Dange et al. Proposed a local cloud framework and scheduling scheme to optimize the resource use of each function in the cloud[13]. The main device collects the function information of all devices, and performs function classification and task scheduling. This resource sensing approach can only perceive the static information of devices, and is not suitable for the scene of dynamic device information.

(3) Due to the influence of device mobility and heterogeneous wireless network environment, there are many choices in task decision-making in MCC. Many scholars are committed to the research of mixed offload decision-making. Zhou B proposed a general offload decision-making framework for heterogeneous Ad-hoc cloud. It uses multiple criterions such as energy consumption, execution time reduction, resource availability,

network conditions and user preferences, and uses mobile devices, nearby small clouds and public clouds to improve the performance and availability of mobile Ad-hoc cloud services. However, the proposed framework focuses on generality, but does not particularly consider the characteristics of mobile Ad-hoc cloud during task allocation [21].

Based on different wireless network technologies, Fahim et al. compared the potential gain of time and energy of the above three offloading methods, and provided a real mobile device cloud platform. The test results show that using mobile device cloud can save 50% of execution time and 26% of energy consumption [22].

In order to reduce energy consumption and computing cost, Guo et al. proposed data offloading and task allocation scheme of mobile device cloud based on small cloud [10]. Using game theory, under the joint constraints of multiple QoS, the scheme can optimize the resource utilization of Ad-hoc cloud and small cloud at the same time.

Li YJ et al. studied the size, lifetime and reachable time of small cloud, and further deduced the upper and lower limits of computing power and computing speed of small cloud. Task initiators can use these two boundaries to decide whether to forward their tasks to a remote cloud or a small cloud for better mobile application services [23]. Chen et al. put forward a new hybrid task offloading framework[24], the task initiator can flexibly select a variety of task execution modes, and further developed a new minimum weight matching algorithm of three-layer graph, which is used for efficient hybrid task offloading between devices. Chi et al. proposed a three-layer mobile game offload architecture based on 'mobile Ad-hoc cloud, cloudlet, cloud'[25]. Compared with the fixed cloud offloading scheme, this scheme has a lower system resource utilization ratio.

(4) Task scheduling in mobile Ad-hoc cloud

Researchers have proposed a variety of mobile ad-hoc cloud task scheduling methods to make offloading decision more beneficial. The common scheduling objectives in Ad-hoc cloud computing environment are to reduce completion time and energy consumption , and to optimize the combination of multiple objectives [18, 21, 28, 29]. As a whole, task scheduling of Ad-hoc cloud can be divided into two types: centralized scheduling[7, 13, 19, 20] [22] [26] and distributed scheduling[18, 27].

Centralized scheduling is more suitable for dynamic environment because of its low communication cost and delay. However, there are still some bottlenecks in mobile network, such as single point failure and high deployment cost. In distributed scheduling, each node has its own scheduling decision-maker, which has scalability and flexibility.

In summary, the above researches need to be further improved in the following aspects:

(1) The above researches assume that the nodes in the cloud are stable and do not change their positions. They are divided into several device clusters by static way through manual or system configuration. There is no relevant research on how to build terminal clusters with dynamic network topology, so the management configuration lacks flexibility. At the same time, the main function of cluster head is to collect and broadcast the device information in the cluster, and the global controller is still responsible for the member management. In order to reduce the flow of global control signaling and enhance the real-time scheduling, the member management and local scheduling function of cluster head can be enhanced.

(2) The above researches on task decision-making and scheduling show that the task decision-making and scheduling in mobile Ad-hoc cloud mainly focuses on the optimal task partition and optimal device scheduling, and the node still provides computing function as a separation service unit; On the other hand, due to the distance of mobile Ad-hoc cloud, the different wireless network environment and the different hardware of the devices, the performance of different clouds (such as transmission delay, communication ability and processing ability) varies greatly. Since the performance of task execution mainly depends on the performance of cloud resources, the research on task scheduling among different self-organizing clouds is relatively less. Further research can be done to analyze the overall service performance provided by self-organizing clouds, and effectively match and schedule the resource requirements and self-organizing cloud performance from the global level, to improve the resource utilization efficiency of self-organizing clouds. In addition, due to the distributed characteristics of mobile users and dynamic task arrival, centralized scheduling can't meet the timeliness and flexibility of scheduling strategy, so it is necessary to study a more flexible and real-time scheduling mechanism.

(3) In the meanwhile, the SDN-based architecture on has been widely used in cellular network in recent years. The role of SDN in D2D communication of 5G network is gradually extending from flow control and routing management to Ad-hoc cloud management. Using SDN can effectively solve the complexity of Ad-hoc cloud and mobile network joint management, reduce the complexity of task scheduling, and increase the fairness of scheduling. Liao et al. applied SDN to remote cloud, mobile cloud and small cloud respectively, discussed their

potential efficiencies and problems, and finally proposed a small cloud architecture based on SDN[28]. Ku et al. studied a self-organizing cloud architecture based on SDN and deployed SDN components in MANET, implemented SDN routing as a cloud service, and discussed various application scenarios of traffic control (such as medical or emergency service traffic) that need to be prioritized based on the wireless self-organizing cloud framework[4]. Authors respectively proposed the hierarchical architecture of self-organizing cloud based on SDN, which divided the terminal equipment group into several self-organizing clouds, and deployed SDN controllers globally to communicate with cluster headers in the cloud[17, 29]. The framework reduced the LTE wireless links of the original terminal, reduced the energy loss of the network, and enhanced the scalability of the wireless network. However, these two papers only considered the decision of communication and energy consumption, and did not consider the scheduling of computing resources. Cui et al. considered the similarity of tasks in the co-location scenario, proposed a software defined collaborative task offloading model in the self-organizing cloud. Based on the sharing of computing results between devices, the model can achieve higher performance and lower energy consumption, reduce the duplication of computing and network traffic, and schedule tasks in a centralized way, effectively solving the problem of large-scale task requests [30]. Baktir et al. use the coordination function provided by SDN to propose a service-oriented and service-centered task offloading Scheme[31]. This scheme uses the functions provided by SDN and OpenFlow, and uses port number in TCP and DSCP field in IP header to achieve service centrality without modifying the existing protocol stack, so as to coordinate edge services transparently for end users.

Based on the above ideas, considering the mobility of devices, the difference of terminal resources, and the diversity of tasks, a framework of Ad-hoc cloud decision scheduling and resource allocation is proposed, to realize the flexibility, stability and matching of task scheduling, and improve the service quality of Ad-hoc cloud services.

Thus, aiming at the maximum execution efficiency, a self-organizing cloud task decision allocation scheme based on dynamic programming is proposed.

III. SYSTEM MODEL

A. Application Scenario

The results and discussion may be presented separately, or in one combined section, and may optionally be divided into headed subsections. In some temporary co-location groups caused by dense areas with spatial characteristics, such as scenic spots, music venues, etc., users need to execute some intensive computing applications, such as “pdf2text” applications. Due to the limitation of terminal resources, they can’t run successfully locally. Users put forward higher and more reliable QoS requirements for the completion of tasks, in this case, the traditional opportunistic task offloading can’t meet the corresponding QoS requirements. Therefore, users turn to Ad-hoc cloud server and are willing to pay a certain execution cost in exchange for better execution performance and better service experience. As shown in Figure 1, the cloud server searches for and schedules to the appropriate terminal cluster for execution in the Ad-hoc cloud. After the terminal in the cluster completes the task, it directly delivers the result to the user through the path in Ad-hoc cloud.

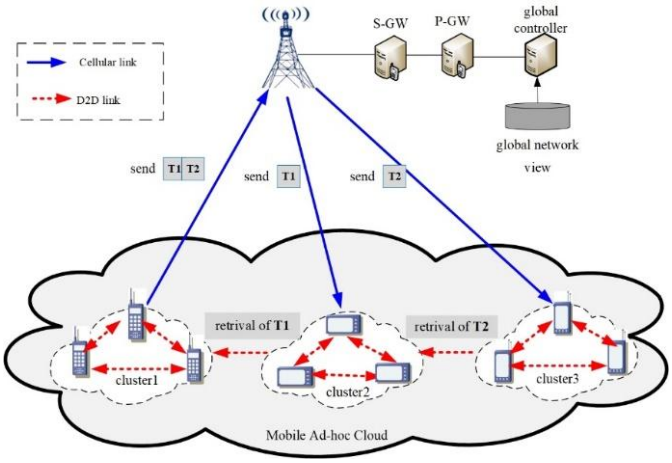


Figure 1. Task offloading scenario of mobile Ad-hoc cloud based on SDN

Due to the lack of flexibility of current IP network, the integration of mobile network and Internet is very prominent. In the existing mobile network, if the self-organizing cloud server is deployed separately, it not only

needs the deployment cost, but also changes the existing mobile network architecture and routing, which will lead to certain integration complexity. Therefore, as shown in Figure 1, in order to simplify the system model, we propose a method to integration the self-organizing cloud service scheduling function in the global controller based on the SDN framework without changing the existing mobile network elements, Because the self-organizing cloud server needs real-time interaction information with the SDN controller (for example, the cloud server requests global network information from the controller as the basis for decision-making and scheduling), the unified integration can reduce the delay of mutual information transmission and improve the real-time performance of scheduling; Secondly, as an open framework with management capability, SDN can implement the scheduler with task offloading function through software based controller and programmable northbound application program interface.

B. Problem Formulation

The scenario studied is aimed at the terminal task requirements generated by users in the same LTE base station, assuming that there are N users in the temporary group under the current base station, and the user terminal wireless interface supports two communication modes: cellular communication and D2D communication. The heterogeneous terminal resources owned by each user, such as computing resources, communication resources, storage resources, are provided as a whole service for task scheduling. Suppose that terminal resource of i is expressed as a three-dimensional tuple: $\mathbf{R}_i = \{r_i^{\text{com}}, r_i^{\text{band}}, r_i^{\text{ram}}\}$, where r_i^{com} denotes the computing processing capacity (in MIPs), r_i^{band} denotes cellular network bandwidth (in Mbps) and r_i^{ram} denotes the memory resource (in MB).

Since some local resources do not meet the requirements of tasks, i submits several task requests to the cloud service controller. To ensure the QoS requirements of users, users have specifically described the requirements of task execution resources and completion time, which consist of the following two aspects:

(1) Completion time limit (τk) and the number of instructions required to complete the task (Ik);

(2) Requirements of tasks for each terminal resource dimension vary according to actual applications, and are expressed as a triple: $\mathrm{D}_k = \{t_k^{\text{com}}, t_k^{\text{band}}, t_k^{\text{ram}}\}$, in which t_k^{com} denotes the requirements of computing resources (in MIPS), t_k^{band} denotes the requirements of cellular bandwidth (in Mbps), and t_k^{ram} denotes the requirements of memory (in MB).

Next, the specific composition of the execution efficiency function is analysed.

Definition 1: The offloading efficiency of the task is defined as the resource efficiency saved by the terminal locally after the task is executed by the cloud, which is measured according to the CPU, memory and bandwidth resources. Therefore, the offloading efficiency of Tk allocated to the self-organizing cloud is defined as:

$$\mathrm{b}_i(\mathrm{T}_k) = \mathrm{I}_k \mu_{\mathrm{I}} + r_i^{\text{band}} t_i^{\mathrm{l}} \mu_{\mathrm{b}} + r_i^{\text{ram}} t_i^{\mathrm{L}} \mu_{\mathrm{r}} \tag{1}$$

Where μI is the unit instruction price, μb is the unit bandwidth price, μr is the unit memory price, and t_i^{L} is the resource occupation time for the task to be executed locally.

There may be resource waiting when tasks are executed locally, so the longest resource waiting time is used to estimate t_i^{L}:

$$t_i^{\mathrm{L}} = \frac{\mathrm{I}_k}{r_i^{\text{com}}} \max(t_i^{\text{band}} / r_i^{\text{band}}, t_i^{\text{ram}} / r_i^{\text{ram}}) \tag{2}$$

In the above equation, $\frac{\mathrm{I}_k}{r_i^{\text{com}}}$ indicates the local execution time of the task, and $\max(t_i^{\text{band}} / r_i^{\text{band}}, t_i^{\text{ram}} / r_i^{\text{ram}})$ indicates the maximum execution time occupying bandwidth and memory.

Assuming that the terminal resources of i are (1000MIPS, 20Mbps, 4G), the number of execution instructions of Tk is 5000MI, the required resources are (2000MIPS, 30Mbps, 4G), μI = 1 centers/KMI, μb = 0.5 centers/Mbps, μr = 0.5 cents/GB/s, then the local execution time is $t_i^{\mathrm{L}} = \frac{5000}{1000} \max(30/20, 4/4) = 7.5$ seconds, and the offloading efficiency generated by offloading to the self-organizing cloud is 95 cents.

Definition 2: Task execution cost reflects the occupation of resources by tasks. Assuming that the service of mobile self-organized cloud adopts the service model of pricing per unit, task execution cost is defined as the

sum of cloud resources obtained by tasks and execution occupation time:

$$p^{C_j}(T_k)=(r_{C_j}^{com}+r_{C_j}^{band}+r_{C_j}^{ram})\mu_C t_i^{C_j}(T_k) \quad (3)$$

In the above equation, the first item indicates the unit comprehensive resources of the self-organized cloud. Since bandwidth, instructions and memory belong to different dimensions, each resource dimension needs to be preprocessed separately, and the data is uniformly expressed so that it is uniformly measured and expressed as $r_{C_j}^{com}$, $r_{C_j}^{band}$, $r_{C_j}^{ram}$, μC represents the unit price of cloud service comprehensive resources, and $t_i^{C_j}$ indicates the execution elapsed time of Tk on Cj.

Assuming that μC is set at 10 centers/unit/s, the execution cost of Tk in each cluster is seen in TABLE 1.

Definition 3: Tk is assigned to Cj for execution, and the generated efficiency function is defined as:

$$u^{C_j}(T_k)=b_i(T_k)-p^{C_j}(T_k)|\ T_k \in T_i \quad (4)$$

Accordingly, the execution efficiencies corresponding to Tk executed on C1 to C4 are 77, 84, 82 and 81 cents respectively, and it can be seen that the execution efficiency obtained by allocating Tk to C2 is the highest.

In view of the dynamicity and heterogeneity of resources, based on the performance terminal clustering, the optimization questions are raised: If the user submits the task request information to the global controller one after another, denoted as $T=\{T_1,T_2,...T_m\}$. At t0, the task set received by the global controller is expressed as follows: Its scheduling goal is to find the best Cj to complete each task in the formed K clusters within the user's residence time and under the requirements of the task resource and execution time, so as to obtain the maximum execution efficiency:

$$\max \sum_{T_k \in T} \sum_{C_j \in £} u^{C_j}(T_k) \quad | t \in [t_0, t_0+T] \quad (5)$$

s.t.

$$\sum_{T_k \in T} a_k^j t_k^{com} \le r_{C_j}^{com} \quad (6)$$

$$\sum_{T_k \in T} a_k^j t_k^{band} \le r_{C_j}^{band} \quad (7)$$

$$\sum_{T_k \in T} a_k^j t_k^{ram} \le r_{C_j}^{ram} \quad (8)$$

$$e^{C_j}(T_k) \le \tau_k \quad (9)$$

$$\sum_{C_j \in £} a_k^j \le 1 | \forall T_k \in T \quad (10)$$

$$a_k^j \in \{0,1\} \quad (11)$$

Equation (5) indicates that the objective function is to achieve the greatest execution efficiency. (6)-(8) represent the resource limitation of cluster services respectively, and the total resources occupied by tasks should be less than the resources provided by terminal clusters; (9) is the constraint condition of completion time, which requires that the completion time must meet the required time limit for task execution. (10) indicates that for each Tk, at most one cluster can be responsible for executing the task. In (11), when $a_k^j = 1$,Tk is scheduled to Cj, otherwise it is 0.

Table 1. Example of Execution Costs on Clusters

Cluster number	Processing capacity (MIPS)	Processing capacity (normalized)	Bandwidth (Mbps)	Bandwidth Capability (Normalized)	Memory (GB)	Memory Capacity (Normalized)	Comprehensive resources (unit)	Execution elapsed time (sec)	Task execution costs (Cent)
1	2000	0.18	50	0.19	8	0.33	0.71	2.50	18
2	4000	0.36	100	0.38	4	0.17	0.91	2.00	11
3	3000	0.27	50	0.19	8	0.33	0.80	1.67	13
4	2000	0.18	60	0.23	4	0.17	0.58	2.50	14

Table 2. Symbols

Symbol	Notation
$X_{u,v}$	Encounter duration between (u,v)
$I_{u,v}$	Encounter interval between (u,v)
P_j	Probability of l providing offloading service for C_j

$ST_l(x)$	The longest service time for l to serve C_j within x
$\Psi_{CH,l}$	Task distribution path set from cluster head to l
$\varphi_{l,i}$	Task result recycling path set from member l to task requester i
$d_{\Psi_{CH,l}}$	Average transmission delay of task distribution path
$d_{\varphi_{l,i}}$	Average Transmission delay of result retrieval path
$\Psi_{l,i}^k$	The k-th task distribution path of $\Psi_{l,i}$
$\varphi_{l,i}^k$	The k-th task distribution path of $\varphi_{l,i}$
El	Average executable time of l
$\mathbf{R}_{C_j}$	The resource vector of C_j
$\mathop{E}_{l \in C_j}(\mathbf{R}_l)$	The expectation of $\mathbf{R}_{C_j}$
$Q(C_j)$	Current execution queue length of C_j
Queue (l)	Current execution queue length of l
$t^{C_j}(T_k)$	Transmission time for T_k from global controller to C_j
$c^{C_j}(T_k)$	The execution time of T_k in C_j

IV. MOBILE AD-HOC CLOUD TASK SCHEDULING STRATEGY BASED ON DYNAMIC PROGRAMMING

A. *Problem Analysis*

Based on the cluster construction based on terminal classification, we propose a task decision model based on mobile Ad-hoc cloud. As shown in Figure 2, the global controller makes a unified decision on the similarity between the task resource requirements and the performance of the terminal cluster based on the global network view, and the corresponding cluster completes the tasks, such computing mobile applications as image recognition and processing, augmented reality, etc., which requires high CPU speed and can be executed by the cluster, the task of downloading file requiring high bandwidth, which can be executed by cluster with high cellular bandwidth.

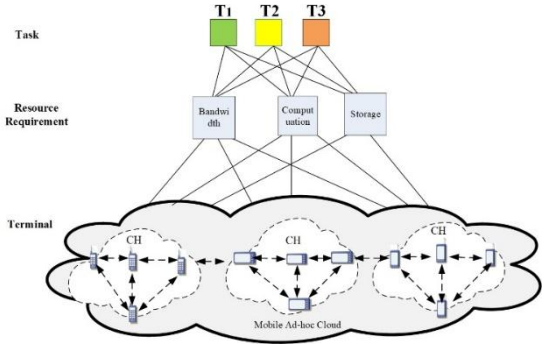


Figure 2. Task scheduling Model Based on Mobile Ad-hoc Cloud

To solve the problem of multi-task and multi-cluster allocation, how to select the cluster with the best performance from the formed K clusters and improve the execution efficiency. Based on this, we propose a decision allocation scheme of mobile Ad-hoc cloud based on dynamic programming, aiming at maximizing execution efficiency.

The symbols used in this paper are shown in Table1.

The resources of Cj come from the resources provided by member l in the reachable path within a given time limit. Therefore, the definition of the longest service duration is introduced below.

Definition 5: S l(x) is defined as the longest service time for l to serve Cj within x, that is, the time between the initial

Assuming that the request node is represented as i, the total offloading time (τ) consists of the average transmission delay of task dispatch path, the average transmission delay of result retrieval path and the average executable time of l denoted as el, shown as follows:

$$e_l + d_{\Psi_{CH,l}} + d_{\varphi_{l,i}} = \tau \tag{12}$$

According to the number of hops on the path, taking task dispatching path as an example, the reachable path from the cluster head to the execution node (l) can be divided into: 1) single hop offloading path: the cluster head

can reach l directly within the D2D communication range of the cluster head; 2) multi hop offloading path: l is not in the D2D communication range of the cluster head, but it can reach l through D2D relay nodes within multi hops.

To simplify the analysis, the single hop offloading path is regarded as a special multi hop path without any relay node. $\Psi_{l,i}^{k}$ denotes the k-th distribution path of $\Psi_{l,i}$, expressed as $\Psi_{l,i}^{k}=\{\mathrm{CH},...,u,v,...,l\}$. Since the transmission delay of multi-hop path is generated in the process of opportunistic encounter between each relay node, which is composed of multi-hop D2D transmission, the average transmission delay of $\Psi_{l,i}^{k}$ is:

$$d_{\Psi_{\mathrm{CH},l}^{k}}=\mathrm{E}\left(\sum_{u,v\in\Psi_{\mathrm{CH},l}^{k}}\left(\mathrm{I}_{u,v}+\mathrm{X}_{u,v}\right)\right) \tag{13}$$

Because the variables are independent of each other, so:

$$d_{\Psi_{\mathrm{CH},l}^{k}}=\sum_{u,v\in\Psi_{\mathrm{CH},l}^{k}}\left(\mathrm{E}\left(\mathrm{I}_{u,v}\right)+\mathrm{E}\left(\mathrm{X}_{u,v}\right)\right) \tag{14}$$

Assuming that the probability of each selected path is identical, the transmission delay of the distribution path is expected to be:

$$\begin{aligned} d_{\Psi_{\mathrm{CH},l}}&=\left|\Psi_{\mathrm{CH},l}\right|^{-1}\sum_{k=1}^{\left|\Psi_{\mathrm{CH},l}\right|}d_{\Psi_{\mathrm{CH},l}^{k}}\\ &=\left|\Psi_{\mathrm{CH},l}\right|^{-1}\sum_{k=1}^{\left|\Psi_{\mathrm{CH},l}\right|}\sum_{u,v\in\Psi_{\mathrm{CH},l}^{k}}\left(\mathrm{E}\left(\mathrm{I}_{u,v}\right)+\mathrm{E}\left(\mathrm{X}_{u,v}\right)\right)\end{aligned} \tag{15}$$

where $\left|\Psi_{\mathrm{CH},l}\right|$ represents the number of paths in $\Psi_{\mathrm{CH},l}$.

Let $\varphi_{l,i}^{k}$ denote the k-th retrieval path, expressed as $\varphi_{l,i}^{k}=\{l,\mathrm{s},...,u,v,...,i\}$. Similarly, the transmission delay of retrieval path is expected to be:

$$d_{\varphi_{l,i}}=\left|\varphi_{l,i}\right|^{-1}\sum_{k=1}^{\left|\varphi_{l,i}\right|}\sum_{u,v\in\varphi_{l,i}^{k}}\left(\mathrm{E}\left(\mathrm{I}_{u,v}\right)+\mathrm{E}\left(\mathrm{X}_{u,v}\right)\right) \tag{16}$$

Assuming l execute and return the task results in "carry-execute" mode, and Z_{ls}^{i} represents the time variable of the i-th opportunistic calculation between (l, s) during x. Since $\{\ \mathrm{Z}_{ls}^{i}$,i=1,2,…,n $\mid n=\lambda_{ls}x\ \}$ is a series of iid random variables, $\sum_{i=1}^{n}Z_{ls}^{i}$ represents the total time from the first encounter to the nth encounter between (l, s) during x. Let el denote the longest executable time of l, so $\sum_{i=1}^{n}\mathrm{Z}_{l,s}^{i}\le e_l$.

According to the definition of $ST_l(x)$, we can get:

$$ST_l(x)=\sum_{i=1}^{n}Z_{ls}^{i}\mid n=\lambda_{ls}(x-d_{\Psi_{CH,l}}-d_{\varphi_{l,i}}) \tag{17}$$

Let the PDF of Z_{ls}^{i} denoted as $f_{Z_{ls}}(x)$, then PDF of $ST_l(x)$ can be deduced as follows:

$$f_{ST_l}(x)=f_{Z_{ls}^{1}}(x)\mathrm{L}\otimes f_{Z_{ls}^{l}}(x)\mathrm{L}\otimes f_{Z_{ls}^{n}}(x)\mid n=\lambda_{ls}x \tag{18}$$

When τ, $d_{\Psi_{\mathrm{CH},l}}$ and $d_{\varphi_{l,i}}$ is specified, el can be obtained. In Ad-hoc cloud, l's actual service duration of in τ is equal to the expectation of the longest service duration during el:

$$\begin{aligned}\mathrm{E}\left(ST_l(\tau)\right)&=\int_0^{e_l}xf_{ST_l}(x)\mathrm{d}x\\&=\int_0^{\tau-d_{\Psi_{\mathrm{CH},l}}-d_{\varphi_{l,i}}}xf_{ST_l}(x)\mathrm{d}x\end{aligned} \tag{19}$$

Within τ, the probability of l providing offloading service for Cj is:

$$P_{l}=\mathrm{E}\left(ST_l(\tau)\right)/\sum_{l\in C_j}\mathrm{E}\left(ST_l(\tau)\right) \tag{20}$$

The resource vector of Cj is expressed as:

$$\begin{aligned}\mathbf{R}_{C_j} &= \mathop{\mathrm{E}}_{l\in \mathrm{C}_j}(\mathbf{R}_l)\\ &=(r_{C_j}^{\mathrm{com}}, r_{C_j}^{\mathrm{band}}, r_{C_j}^{\mathrm{ram}})\\ &=(\sum_{l\in \mathrm{C}_j} r_l^{\mathrm{com}} P_l, \sum_{l\in \mathrm{C}_j} r_l^{\mathrm{band}} P_l, \sum_{l\in \mathrm{C}_j} r_l^{\mathrm{ram}} P_l)\end{aligned} \tag{21}$$

The current execution queue length of Cj is:

$$\mathrm{Q}(\mathrm{C}_j) = \sum_{l\in \mathrm{C}_j} \mathrm{queue}(l) P_l \tag{22}$$

where queue(l) represents the current execution queue length of member l.

After receiving the request, the global controller predicts the completion time of each task in each terminal cluster. The completion time of Tk in Cj is the sum of the execution time, task transmission time and queue time of Tk in Cj:

$$\mathrm{e}^{\mathrm{C}_j}(\mathrm{T}_k) = \mathrm{c}^{\mathrm{C}_j}(\mathrm{T}_k) + \mathrm{t}^{\mathrm{C}_j}(\mathrm{T}_k) + \mathrm{Q}(\mathrm{C}_j) \tag{23}$$

The execution time of Tk in Cj is the calculation load of the task divided by the calculation processing capacity of Cj :

$$\mathrm{c}^{\mathrm{C}_j}(\mathrm{T}_k) = \mathrm{I}_k / r_{C_j}^{\mathrm{com}} \tag{24}$$

In order to simplify the analysis, it is assumed that the task transmission time from the global controller to Cj is predicted by RTT (ms) = 0.03 * distance (km) + 5 when the transmission ratio of cellular network and D2D network remains stable[32]. According to $\mathrm{t}^{\mathrm{C}_j}(\mathrm{T}_k)$, the completion time matrix is formed as $\mathbf{E}$:

$$\mathbf{E} = \begin{bmatrix} \mathrm{e}^1(\mathrm{T}_1) & \mathrm{L} & \mathrm{e}^1(\mathrm{T}_m) \\ \mathrm{M} & \mathrm{e}^j(\mathrm{T}_l) & \mathrm{M} \\ \mathrm{e}^{\mathrm{K}}(\mathrm{T}_1) & \mathrm{L} & \mathrm{e}^{\mathrm{K}}(\mathrm{T}_k) \end{bmatrix} \tag{25}$$

Then, according to the completion time matrix, the global controller calculates the execution efficiency of Tk in each cluster. The larger the execution efficiency is, the higher the execution efficiency of the cluster is, and the more reasonable the resource utilization is, then Tk is scheduled to the optimal cluster.

B. ALGORITHM DESIGN

1. Construction self-organizing cloud terminal cluster based on non-uniform granularity

Based on the analysis in the previous section, in this section we will propose an algorithm framework for building an Ad-hoc cloud cluster based on non-uniform granularity. It is assumed that the global controller maintains the encounter information graph($\mathrm{G}(\mathrm{U},\mathrm{E})$)and the device resource table(R).

First, according to the characteristics of terminal resources in the mobile network, we can get a terminal granularity system p= (T, R, S), where P represents the terminal resource pool, T represents the list of all terminal resources in the network, R represents the description set of terminal resources, and S represents the similarity relationship between terminal resources (Euclidean distance is used as the similarity of terminal resources in this model).

Then, according to the similarity, the terminal equipment resources are clustered and analyzed, and the clustering pedigree is obtained. Using the classification cutting value to cut it, we can get different branches.

Then, repeat cutting with finer granularity (cutting value) until the stop condition is met, and finally get the classification result of terminal clusters.

To provide stable intra cluster task scheduling, when selecting the cluster head, the sustainable D2D communication time between the cluster head and members is as long as possible. Therefore, the global controller selects the node with the maximum centrality in Cj as the cluster header:

$$\mathrm{CH}_j = \arg\max(\mathrm{Centrality}_l) \mid \forall l \in \mathrm{C}_j \tag{26}$$

The pseudo code of the algorithm is shown as Algorithm 1, the input is the granularity system (P), the number of terminal clusters (K), and the output is the classification terminal cluster (C) and the number of clusters (n).

Algorithm 1 Mobile Ad-hoc Cloud Cluster Formation

Input: P, K;

1: **Initialization:**

2: V=ϕ; S=ϕ; Z=ϕ; $x=0; n=0;$

3: Perform hierarchical clustering on P to obtain clustering pedigree V;

4: $\mathrm{d}_{\mathrm{cut}}$ =random(maxDistance(V), averageDistance(V));

5: τ=minDistance(V);
6: **while** $n \neq K$ **and** $d_{cut} \geq \tau$ **do**
7: Cut V;
8: calculate *r*;
9: **if** $r > f(x)$ **then**
10: $d_{cut}=C_{max}-\lambda$;
11: **else**
12: $d_{cut}=C_{max}+\lambda$;
13: **endif;**
14: **if** aggregation result set (Z)exists in leaf node **then**
15: V=V\Z;
16: S=S∪ Z;
17: updateΔB;
18: **endif;**
19: calculate *x* and λ;
20: get *n*;
21: **end while;**
22: C=V ∪ S ;
23: *n*=N(V)+N(S) ;
24: **for** *j*=1 **to** k **do**
25: C[*j*].ch= argmax(degree(C[*j*]));
26: **end for;**
27: **return** C, *n;*

The cut value (dcut) represents the size of granularity. The selection of appropriate classification threshold can reflect the difference of sample resources. In the initial state, try to choose coarse granularity. The initial value of dcut is randomly selected between the average sample distance and the maximum sample distance of S. Then, the algorithm executes the hierarchical clustering algorithm on P to obtain the hierarchical cluster set V, and then performs repeated cutting operations on V under the condition that the number of terminal clusters and the cutting threshold are met (lines 6-21). For each cut, compare the cluster ratio with the reference function value, and dynamically adjust dcut (lines 8-13) according to the comparison results. Since leaf nodes belonging to only one class can be obtained in the cutting process, no further cutting is required for these branches. When there are leaf node aggregation result sets, we use a transition set-S to store these aggregation results and update the cutting boundary area at the same time Δ B (lines 14-17). Next, calculate the cutting ratio (X) and update the cutting step, and continue to perform the next segmentation. When the number of self-organizing clouds is equal to K or less than the cutting threshold, it means that the expected self-organizing cloud classification results are achieved and the cycle body is skipped. According to (26), in each formed terminal cluster, select the node with the largest centrality as the cluster head (line 25), and finally return the classified terminal clusters and the number of terminal clusters.

When the scale of terminals is medium, this algorithm can dynamically and adaptively adjust the cutting granularity according to the granularity of sample instances, and carefully reflect the differences between performance categories. It can quickly build a locally optimal and stable terminal cluster set in the network environment with large terminal resource differences and medium sample size. If each cutting produces a branch, the number of iterations of non-uniform cutting is n, the time complexity of classification cutting process is O(N), while the time complexity of cluster head selection process is O(KN), and the time complexity of constructing clustering pedigree is O(N3). The time complexity of this algorithm is O(N3), which can obtain better classification performance within the time complexity of polynomials, produce high-quality classification results.

B. 4.2. Mobile Ad-hoc cloud scheduling algorithm based on dynamic programming

The above optimization problem is to solve a nonlinear optimization problem composed of m*K decision matrix under certain constraints. It can be regarded as solving the matching relationship between m tasks and K clusters. The scheduling goal is to minimize the execution efficiency function of the overall task.

For the optimal scheduling problem of the multitask multiprocessor, since the execution efficiency of the k-th task scheduling is the execution efficiency of the k-1 item plus the execution efficiency obtained after the k-th task scheduling, according to the above recursive conditions, we can use the dynamic programming method to solve the task scheduling problem:

(1) Phase variable: k

The scheduling order of m tasks is taken as the stage variable, k=1, 2, …, m, and sorted according to the descending order of task scheduling utility value as the allocation scheduling order.

(2) State variable: xkj

Represents the selection state of the cluster in the k-th stage, because each terminal cluster can be scheduled for selection in each stage, so the state set Xj= {1, 2, …, K}.

(3) decision variable: ak

When the k-th task is scheduled to the j-th cluster in the xkj state at the k-stage, the decision variable ak(xkj)=j and the decision variable allows the set Ak to be {1, 2, …, K}.

(4) The stage goal is the execution efficiency of the current stage:

$$\mathrm{v}_k(x_k, a_k(x_{kj})) = \mathrm{u}^j(\mathrm{T}_k) \tag{27}$$

The optimal function is to obtain the overall maximum execution efficiency, which can be expressed as:

$$f_k(x_{kj}) = \max[\mathrm{v}_k(x_k, a_k(x_{kj})) + f_{k\text{-}1}(x_{(k\text{-}1)j})] \tag{28}$$

(5) State transition equation

When the state variable x(k-1)j and the decision variable ak-1 in the k-1 stage are determined, the remaining resources and task queues on each cluster are also determined accordingly, then the execution efficiency of the kth task on each terminal cluster in the kth stage state can also be determined, and the value of xkj is also determined correspondingly according to the optimal scheduling target. The state transition equation can be expressed as:

$$x_k = \mathrm{T}_{k\text{-}1}(x_{(k\text{-}1)j}, a_{k\text{-}1}(x_{(k\text{-}1)j})) \tag{29}$$

(6) The starting conditions are:

$$f_0(x_0) = 0 \mid \forall i \in \{1, \mathrm{L}, \mathrm{K}\} \tag{30}$$

According to recursion conditions, the pseudo code of the self-organizing cloud task decision algorithm is shown in Algorithm 2. The input of the algorithm is task set and terminal cluster set, and the output result is the scheduling policy:

The execution idea of the algorithm is to predict the execution efficiency of each task on different clusters under different scheduling strategies in each state, and select the scheduling strategy with the greatest execution efficiency, specifically: initialize the execution queue duration and remaining service resources of each cluster (lines 4-5), and the execution efficiency of each cluster is 0 (lines 6-8) in the initial state; Calculate the execution time of each task on a different cluster in turn (line 11), According to the execution queue duration of the previous stage and the remaining service resources, calculate the completion time (line 13), When the completion time constraint is met, It is judged whether the resource requirements of the task are met. When the above constraints are met at the same time, the maximum overall execution efficiency of the current stage (line 21) is obtained from the total execution efficiency of the previous stage and the execution efficiency of the current stage according to the recursive condition. Then, the optimal policy path (line 22) is saved, and the execution queue duration and remaining service resources of the j are updated. Finally, the optimal overall execution efficiency of the m-th stage (lines 27 to 29) is obtained. According to the strategy corresponding to the optimal efficiency, the optimal scheduling matrix (lines 30 to 34) is obtained by recursion from back to front in the saved scheduling path. In the worst case, the time complexity of the algorithm is O (mK2) and the space complexity is O (Km). From the above analysis, it can be seen that this algorithm obtains the global optimal scheduling strategy at the expense of large space overhead. In this algorithm, the use of multiple constraints can exclude most infeasible states, so it can greatly reduce the amount of space storage.

Algorithm 2: Mobile Ad-hoc Cloud Task Decision

Input: T,£

1: **Initialization:**
2: The number of tasks of T →m;
3: Number of terminal clusters→ K;
4: Obtaining current queue length of each cluster;
5: Obtaining the remaining service resources of each cluster;
6: **for** *j* = 1 to K **do**

7: $f[j] = 0$;
8: **end for;**
9: **for** k=1 to m **do**
10: **for** j = 1 to K **do**
11: Calculate $c^j(T_k)$;
12: **for l = 1 to K do**
13: Calculate $e^j(T_k)$;
14: **if (** $e^j(T_k) \le \tau_k$**) then**
15: **if** (j's remaining computing resources > and j's remaining bandwidth resources > and j's remaining memory resources >) **then**
16: Calculate $u^j(T_k)$;
17: $f[k][j]$=max($u^j(T_k)$ +$f^*[k\text{-}1][l]$);
18: **endif;**
19: **endif;**
20: **end for;**
21: l^*=argmax ($f[k][j]$);
22: path. add ((k, j), l*);
23: Update j's remaining service resources;
24: Update j's queue length;
25: **end for;**
26: **end for;**
27: **for** j = 1 to K **do**
28: curNode=argmax(f[m][j]);
29: **end for;**
30: **for** (k=m, k=1, k--) **do**
31: X[k][curNode] = 1;
32: PreNode=path. get ((k, curNode));
33: CurNode=preNode;
34: **end for;**
35: **return X;**

V. NUMERICAL RESULTS

To further study the performance of this algorithm, two types of offloading algorithms are compared in this simulation. The task decision-making strategies of all offloading algorithms are different, but the scheduling strategy of "minimum load, priority scheduling" is adopted in the cluster scheduling.

One is opportunistic task offloading algorithm:

(1) Min-Min Scheduling Algorithm (MMSA), in which the task initiator assigns the task to the terminal with the shortest processing time among the neighbouring meeting users, so as to ensure the shortest completion time.

(2) Socially Perceived Task Unloading Mechanism (SATO), in which the possibility of task completion is predicted, and the node with the largest unloading probability is selected.

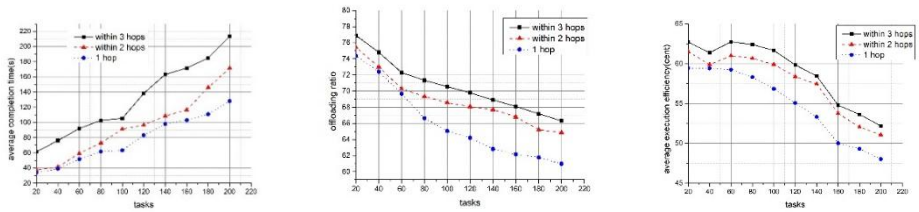
The other is decision scheduling algorithm based on terminal cluster construction:

(1) SDN-based Ad-hoc cloud decision and scheduling (SACDS)

(2) K-means-based Ad-hoc cloud decision and scheduling (KACDS) based on K-means has the same task decision and intra cluster scheduling method as SACDS, but uses K-means clustering algorithm to construct terminal cluster. The specific method is: firstly, select K terminals randomly as the initial cluster centre, then calculate the distance between the other terminals and each cluster canter, and assign the terminals to the nearest cluster center until the cluster centre does not change.

To ensure the accuracy of the experimental results, 50 experiments are repeated in the same simulation environment, and the simulation results are averaged to obtain the simulation results, to evaluate the performance of different task scheduling algorithms.

In the simulation, the number of clusters is set as K= 4, and the hops are set as single hop, within 2 hops and within 3 hops respectively. The influence of different hops on the offloading performance of NCCDS is analyzed. It can be seen from Figure 3 that the overall offloading ratio and average execution efficiency decrease with the increase of the number of tasks, and the average completion time increases with the increase of the number of tasks.



(a)offloading ratio (b)average completion time (c)average execution efficiency

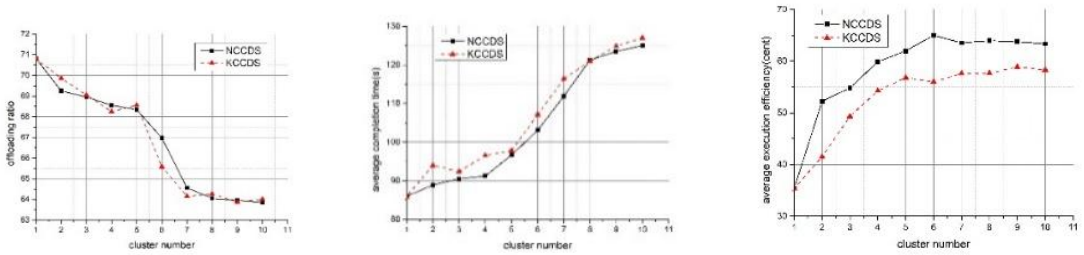
Figure 3. Influence of hops

For example, the offloading ratio of 2 hops is 4.42%, higher than that of single hop, and the execution efficiency is 4.86% higher than that of single hop, but the average completion time increases by 21.84%. The improvement of the offloading ratio and execution efficiency is at the cost of the rapid increase of the task completion delay of the multi hop path. At the same time, as shown in Figure3 (b), when the number of path hops increases from less than 2 hops to less than 3 hops, the average completion time will increase rapidly (the increase proportion is 31.93%), while the increase of offloading ratio and execution efficiency is relatively limited (the increase proportion is 2.41% and 2.59%, respectively). This is because in the wireless network environment, when the number of hops is greater than 2, the average completion time will increase rapidly, multi-hop relay will cause long data transmission delay. Therefore, in the following simulation, we focus on the more practical path within 2 hops (including 2 hops).

(2) Influence of cluster number

To verify the effect of terminal classification method on task scheduling, the performance of NCCDS and KCCDS based on terminal cluster classification is compared.

When the number of tasks is 100, as shown in Figure 4(a), the overall offloading ratio decreases with the increase of K, and the overall offloading ratio of NCCDS algorithm is higher than that of KCCDS. When K is small ($K \leq 4$), the offloading ratio of NCCDS remains stable, while the KCCDS will change slightly with the classification result due to the selection of initial aggregation point.



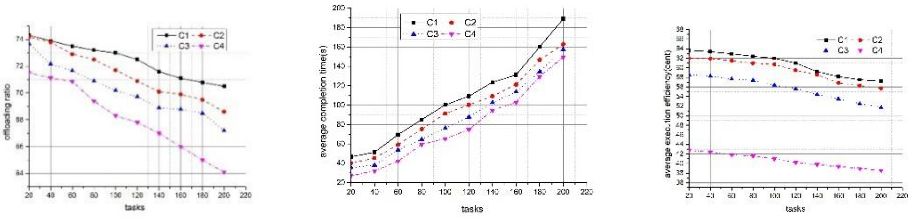
(a)offloading ratio (b)average completion time (c)average execution efficiency

Figure 4. Influence of cluster number

When K is larger ($K \geq 5$), the success ratio of scheduling decreases sharply. This is because with the decrease of device number in the cluster, the chance of meeting each other decreases, resulting in the weakening of task offloading ability, the increase of completion time, and the corresponding decrease of offloading ratio. Especially, when K = 7, the success ratio of scheduling appears inflection point, which decreases to 64.56% and 64.15% respectively. After that, when K is further increased, the offloading ratio decreases very slowly, and the offloading effect cannot be enhanced.

From Figure4 (b) and (c) the average completion time and execution efficiency of the two algorithms increase with the increase of K, and the offloading ratio reaches the best when K = 3 or 4. This is because with the increase of K, due to the decrease of the number of devices in the cluster, the average encounter chance between nodes decreases, and the corresponding completion time also increases. But at the same time, with the increase of the number of terminal cluster classification, the greater the similarity between tasks and clusters, the higher the execution efficiency. NCCDS is more accurate, so its average execution efficiency is greatly affected by the cluster number.

(3) Performance comparison of clusters



(a)offloading ratio (b)average completion time (c)average execution efficiency

Figure 5. Influence of tasks

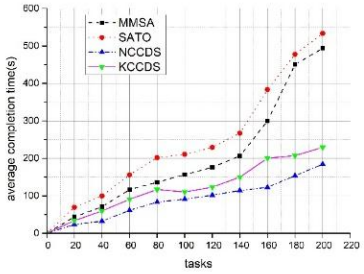
Set the number of terminal clusters K = 4. After non-uniform classification, four terminal clusters are obtained, which are sorted in descending order according to the comprehensive performance, and are represented as C1, C2, C3 and C4. The number of members in the corresponding cluster is 25, 31, 21 and 20. The resource vectors of each cluster are (3804.1, 5.5, 50.7), (2800.5, 4.1, 53.7), (1810.2, 4.3, 45.1) and (833.3, 4.8, 49.8) respectively, Run the mobile Ad-hoc cloud decision and scheduling algorithm to obtain the offloading performance of each terminal cluster, as shown in the figure below.

When the number of tasks increases, the offloading ratio and execution efficiency generally decrease, and the average completion time also increases. The stronger the comprehensive performance is, the higher the offloading ratio and execution efficiency are, and the longer the average completion time is. As shown in Figure 5 (b), when the number of tasks increases from 20 to 200, the average completion time of each cluster increases by 142.42 seconds, 122.60 seconds, 121.78 seconds and 122.44 seconds, respectively. Among them, C1 has the best comprehensive terminal resource performance and the highest execution efficiency. At the same time, the decline of offloading ratio is also affected by the number of members. As shown in Figure 5 (a), the decline of offloading ratio of each cluster are 9.4%, 8.8%, 9.8% and 10.4%. From the analysis, the decline of offloading ratio of C2 is less affected by the number of tasks because of its large number of members.

(4) Comparison of different algorithm performance

Setting the number of terminal clusters K=4, as can be seen from Figure 6, under the condition of the same number of tasks, NCCDS has the highest offload ratio, mainly because when scheduling requests, there are multiple similar performance nodes in the cluster that meet the constraints and can be selected for scheduling. From the overall trend of the scheduling success ratio, with the gradual increase of the number of tasks, the scheduling success ratios of the four algorithms all show a downward trend, which are: 47.3% (MMSA), 45.3% (SATO), 17.7% (KCCDS), 14.0% (NCCDS). Among them, NCCDS has the smallest decline, which shows that its service stability and reliability are the highest.

As shown in Figure 6, in comparison, NCCDS has the shortest average completion time, which is 45.0%, 36.8% and 73.4% of MMSA, SATO and KCCDS respectively. When the number of tasks increases, the nearby scheduled terminal resources are limited by the number and capability, thus resulting in a large completion delay, and the processing advantage of opportunistic offload algorithm is correspondingly weakened. However, the algorithm based on self-organizing cloud terminal cluster can effectively narrow the search range of resource matching after using terminal resource classification, and can realize the matching between tasks and resources in a shorter time, thus obtaining a smaller completion time.



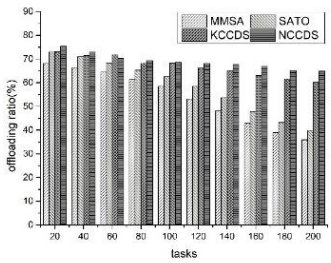


Figure 6. Comparison of offloading ratio **Figure 7.** Comparison of average completion time

We compare the execution efficiency of two algorithms based on terminal cluster classification, as can be seen from Figure 7, with the increase of the number of tasks, the execution efficiency shows an overall downward trend, especially when the number of tasks reaches more than 140, there is a rapid decline. The reason

is that the continuous increase of the number of tasks causes the execution queue duration of the terminal to correspondingly increase, the optimal matching ability between tasks and resources to weaken, and the execution efficiency decreases. However, under the same number of tasks, the average execution efficiency generated by NCCDS is higher, which is mainly because NCCDS is more stable and accurate than KCCDS algorithm in classification, can better realize the matching of resources and tasks in time limit, effectively schedule tasks in clusters, reduce redundant resource costs, and make more effective use of terminal resources.

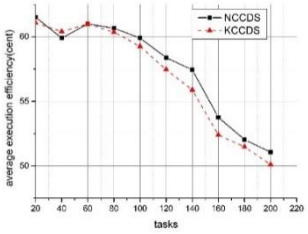
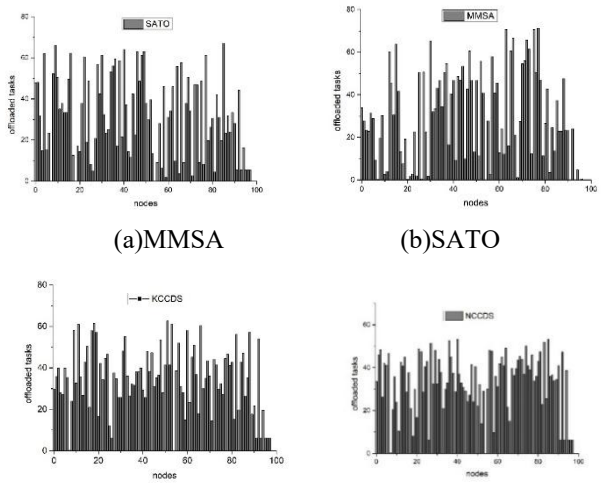


Figure 8. Comparison of average execution efficiency

Under the action of different allocation methods, the distribution of offloaded tasks of nodes is also different. With the change of the number of tasks, the number of nodes with more than 40 tasks offloaded by MMSA, SATO, NCCDS and KCCDS nodes is 39, 35, 29 and 32, respectively, while the number of nodes with less than 10 tasks is 24, 21, 9 and 10, respectively. Furthermore, by analysing the standard deviation of the number of tasks, the corresponding standard deviation is 21.3, 19.8, 11.9 and 13.9. The difference in the number of tasks offloaded by nodes using opportunistic offloading method is obvious, while the self-organizing cloud terminal cluster offloading method can effectively balance the load of nodes, and the load balancing performance of NCCDS is better than KCCDS.



(a)MMSA (b)SATO

(c)NCCDS (d)KCCDS

Figure 9. Comparison of offloaded tasks of nodes

Based on the above simulation results, we can draw the following conclusions:

(1) Increasing the number of hops can improve the offloading ratio and execution efficiency, but the improvement of the offloading ratio and execution efficiency of multi-hop paths comes at the expense of the rapid increase of task completion delay. Because when the number of path hops is greater than 2, multi-hop relay will produce a long data transmission delay, in the simulation of this algorithm, we focus on the practical path within 2 hops (including 2 hops).

(2) The number of clusters plays an important role in offloading success ratio and average execution efficiency. Reasonable selection of the number of terminal clusters can make the proposed method achieve the best performance. The optimal K can be determined by analyzing the performance in massive experiments, observing the slope change of offloading index and paying attention to the existence of "inflection point".

(3) With the gradual increase of task number, the completion time of the four algorithms also increases correspondingly, and the offload success ratio and the average execution efficiency all show a downward trend. Among them, the offload success ratio of NCCDS algorithm decreases the least. Its service has the highest

stability and reliability. However, under the condition of the same number of tasks, NCCDS algorithm also has the highest offloading ratio and average execution efficiency, which is mainly because it considers execution efficiency, effectively schedules tasks in classified self-organizing cloud, realizes the matching of resources and tasks, reduces redundant resource cost, and can make more effective use of terminal resources.

(4) When the scale of terminals in the network is medium and the performance of terminal resources varies greatly, the performance of KCCDS is affected by the selection of initial aggregation points, and the offloading performance will fluctuate. Because NCCDS algorithm is based on non-uniform clustering classification and dynamically adjusts the cutting value according to the similarity distribution of terminal resources, it can provide more stable and efficient cloud service performance than KCCDS algorithm. However, when the number of terminal resources increases extremely fast, this will bring more complicated processing work to the clustering pedigree construction in the preprocessing stage. To meet the system efficiency constraints, the simple KCCDS algorithm is more efficient.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we study the resource scheduling technology in mobile edge network, and propose a task offloading mechanism of self-organizing cloud based on SDN. The mechanism adopts a hierarchical task allocation strategy of centralized decision-making by the global controller and dynamic scheduling by the cluster head within the cluster. Terminals are clustered based on resource attributes, and clusters are constructed in the mobile self-organizing cloud by adopting an unsupervised classification method, centralized offloading decisions are made according to task requirement information, the allocation relationship between tasks and terminal clusters is established, and task decisions are given to terminal clusters with the best execution efficiencies.

Simulation results show that compared with the other three scheduling methods, this method has certain advantages in the effectiveness, stability, and balance of offloading, and can be more effectively applied to mobile edge networks, thereby improving network performance and user service satisfaction.

In the next research, we will mainly solve the following problems:

(1) We will further subdivide the task cooperation scenarios among clusters of self-organizing cloud terminals, including mechanisms such as task segmentation, task migration and task reallocation among clusters, so that the method has better scalability and can be applied to large-scale networks;

(2) The application of social networking task scheduling in edge network is to be discussed, including the application of community classification technology in terminal cluster construction, and the research on task allocation mechanism of social perception enhancement according to social encounter mode;

(3) We will study the possibility of combining the self-organizing cloud task offloading mechanism proposed in this paper with other autonomous learning pattern mining technologies, and analyse whether the matching is more accurate and whether the performance of task scheduling and allocation is better.

Although we have considered the charging problem of self-organized cloud services, we don’t carry out specific research on the division of cloud service revenue into members and the incentive measures of cluster heads, and will further carry out corresponding research in the future.

VII. DATA AVAILABILITY

No underlying data were collected or produced in this study. Our experiment is conducted in a simulation mode, and the settings of the data used to support the findings of this study are available from the corresponding author upon request.

The main parameters are set as follows:Using Infocomm track set[33], a cell with a radius of 300m is served by the base station. The total number of mobile users in the cell is N= 98 (number: 0-97). User's sojourn time in the cell is 180 minutes. The user moves in the cell by random walking. The walking ratio is in line with the uniform distribution of [0,1] m/s. In each round, 10% of the users are randomly selected as the request nodes.

The node 98 acts as the base station and global controller. The location of the node remains fixed and can establish a cellular connection with any terminal node. The terminal cluster construction and decision algorithm is loaded in the node to simulate the decision-making role of the global controller. Load the task scheduling algorithm on other terminals to simulate the scheduling function of cluster header. The scheduling policy inside cluster adopts the scheduling strategy of ‘minimum load, priority scheduling’.

Assuming that the resource allocation of the terminal remains unchanged in each simulation round, in order to reflect the performance difference, the computing performance of the terminal is selected in {500,600,..., 4000} MIPs, and the memory is selected in {1,2,..., 8} GB. The cellular communication bandwidth conforms to the

uniform distribution of [0,100] Mbps, and the D2D communication ratio is 24 Mbps.

In each simulation round, 10% of the nodes in the device set are randomly selected as the request nodes. Each request node generates tasks randomly, and the ratio of task generation follows the average value of 0 λ The Poisson distribution of I, λ I was randomly selected from {0.01, 0.02,..., 0.1} (times/s), the number of tasks generated each time was randomly selected from {20, 40,..., 200}, and the completion time limited τk is 20 minutes, and task computing load Ik is randomly selected from {1000, 2000,..., 10000} MI. The computing resource requirements of tasks are selected from {1000, 1100, ..., 2000} MIPs, the requirements of memory are selected from {1,2,3,4} GB, and the requirements of cellular bandwidth are selected from {10, 20,..., 50} Mbps. The task input data volume is randomly selected from [400, 500] KB, and the returned data is randomly selected from [10,100] KB.

To reduce the information transmission between cluster head and global controller in cellular network, according to the minimum time granularity of task request interval, the cluster head information collection and reporting cycle and self-organizing cloud task decision cycle are set to 10 seconds.

Referring to the performance comparison between terminal equipment and cloud server , assuming that the relative performance of terminal cluster and terminal equipment is about 10:1, the unit resource price of terminal is set as: μI =1 cents/K MI, μb =0.5 cents/Mbps, μr = 0.5 cents /GB/s, unit service price of terminal cluster μC is 10 cents/unit/s, and K is set from 1 to 10.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Funding Statement

This research was funded by the Research Foundation of Nanjing Vocational University of Industry Technology [No: YK20-05-04];

Open Foundation of Industrial Software Engineering Technology Research and Development Centre of Jiangsu Education Department [No: ZK20-04-13];

2023 Research Project of Chinese Vocational Technology Education Association (No: ZJ2023B167); Modern Educational Technology Research Program of Jiangsu Province in 2022[No.2022-R-98629].

References

[1] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," IEEE Pervasive Computing, vol. 8, pp. 14-23, 2009.

[2] Z. Yang, D. Niyato, and W. Ping, "Offloading in Mobile Cloudlet Systems with Intermittent Connectivity," IEEE Transactions on Mobile Computing, vol. 14, pp. 2516-2529, 2015.

[3] D. Fesehaye, Y. Gao, K. Nahrstedt, and G. Wang, "Impact of Cloudlets on Interactive Mobile Cloud Applications," in 2012 IEEE 16th International Enterprise Distributed Object Computing Conference, Beijing, 2012, pp. 123-132.

[4] I. Ku, Y. Lu, and M. Gerla, "Software-Defined Mobile Cloud: Architecture, services and use cases," in 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), 2014, pp. 1-6.

[5] S. Cong, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling Remote Computing among Intermittently Connected Mobile Devices," in MobiHoc '12 Proceedings of the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing, Hilton Head, 2012, pp. 145-154.

[6] A. Mtibaa, K. A. Harras, and A. Fahim, "Towards Computational Offloading in Mobile Device Clouds," in 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, 2013, pp. 331-338.

[7] Y. Nakamoto, "Hyrax - A Hyperlink-Based Application Framework for Smart Devices," in Third IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS'05), 2005, pp. 58-62.

[8] S. Ghasemi-Falavarjani, M. Nematbakhsh, and B. Shahgholi Ghahfarokhi, "Context-aware multi-objective resource allocation in mobile cloud," Computers & Electrical Engineering, vol. 44, pp. 218-240, 2015/05/01/ 2015.

[9] E. Miluzzo, R. Cáceres, and Y. F. Chen, "Vision: mClouds - computing on clouds of mobile devices," in Proceedings of the third ACM workshop on Mobile cloud computing and services, Low Wood Bay, Lake District, UK, 2012, pp. 9-14.

[10] X. Guo, L. Liu, C. Zheng, and T. Ristaniemi, "Data offloading and task allocation for cloudlet-assisted ad hoc mobile clouds," Wireless Networks, vol. 24, pp. 79-88, 2018.

[11] N. Fernando, W. L. Seng, and W. Rahayu, "Dynamic Mobile Cloud Computing: Ad Hoc and Opportunistic Job Sharing," in 2011 Fourth IEEE International Conference on Utility and Cloud Computing, Victoria, 2011, pp. 281-286.

[12] S. Loke, K. Napier, A. Alali, N. Fernando, and W. Rahayu, "Mobile Computations with Surrounding Devices," ACM Transactions on Embedded Computing Systems, vol. 14, pp. 1-25, 02/17 2015.

[13] N. Dange, K. Devadkar, and D. Kalbande, "Scheduling of task in collaborative environment using mobile cloud," in 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication

(ICGTSPICC), 2016, pp. 579-583.
[14] J. Peng, D. Ma, K.-Y. Liu, Q.-Q. Zhang, and X.-Y. Zhang, "LTE D2D based vehicle networking communication architecture and data distributing strategy," Tongxin Xuebao/Journal on Communications, vol. 37, pp. 62-70, 2016.
[15] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto Clouds: Leveraging Mobile Devices to Provide Cloud Service at the Edge," in 2015 IEEE 8th International Conference on Cloud Computing, 2015, pp. 9-16.
[16] S. M. Asghari, Y.-H. Kao, M. H. Lotfi, M. Noormohammadpour, B. Krishnamachari, B. H. Khalaj, et al. (2013, 2019-8-16). LORD: Leader-based framework for Resource Discovery in Mobile Device Clouds. Available: https://arxiv.org/abs/1308.0959
[17] A. Muthanna, A. A. Ateya, M. A. Balushi, and R. Kirichek, "D2D enabled communication system structure based on software defined networking for 5G network," in 2018 International Symposium on Consumer Technologies (ISCT), 2018, pp. 41-44.
[18] L. Xiaoshuang, H. Hassanein, and S. Akl, "Energy aware dynamic task allocation in mobile ad hoc networks," in 2005 International Conference on Wireless Networks, Communications and Mobile Computing, 2005, pp. 534-539.
[19] I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar, and M. Imran, "Heterogeneity-Aware Task Allocation in Mobile Ad Hoc Cloud," IEEE Access, vol. 5, pp. 1779-1795, 2017.
[20] Z. Lu, J. Zhao, Y. Wu, and G. Cao, "Task Allocation for Mobile Cloud Computing in Heterogeneous Wireless Networks," in 2015 24th International Conference on Computer Communication and Networks (ICCCN), 2015, pp. 1-9.
[21] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, "mCloud: A Context-Aware Offloading Framework for Heterogeneous Mobile Cloud," IEEE Transactions on Services Computing, vol. 10, pp. 797-810, 2017.
[22] A. Fahim, A. Mtibaa, and K. A. Harras, "Making the case for computational offloading in mobile device clouds," in Proceedings of the 19th annual international conference on Mobile computing &networking, Miami, Florida, USA, 2013, pp. 203-205.
[23] Y. Li and W. Wang, "Can mobile cloudlets support mobile applications?," in IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, Toronto, 2014, pp. 1060-1068.
[24] X. Chen and J. Zhang, "When D2D meets cloud: Hybrid mobile task offloadings in fog computing," in 2017 IEEE International Conference on Communications (ICC), 2017, pp. 1-6.
[25] F. Chi, X. Wang, W. Cai, and V. C. M. Leung, "Ad-Hoc Cloudlet Based Cooperative Cloud Gaming," IEEE Transactions on Cloud Computing, vol. 6, pp. 625-639, 2018.
[26] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile Crowd Computing with Work Stealing," in 2012 15th International Conference on Network-Based Information Systems, 2012, pp. 660-665.
[27] Y. Huang, N. Bessis, P. Norrington, P. Kuonen, and B. Hirsbrunner, "Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm," Future Generation Computer Systems, vol. 29, pp. 402-415, 2013.
[28] L. Liao, M. Qiu, and V. C. M. Leung, "Software Defined Mobile Cloudlet," Mobile Networks & Applications, vol. 20, pp. 337-347, 2015.
[29] M. Usman, A. A. Gebremariam, F. Granelli, and D. Kliazovich, "Software-Defined Architecture for Mobile Cloud in Device-to-Device Communication," in 2015 IEEE 20th International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD), 2016, pp. 75-79.
[30] Y. Cui, J. Song, K. Ren, M. Li, Z. Li, Q. Ren, et al., "Software Defined Cooperative Offloading for Mobile Cloudlets," IEEE/ACM Transactions on Networking, vol. 25, pp. 1746-1760, 2017.
[31] A. C. Baktir, A. Ozgovde, and C. Ersoy, "Enabling service-centric networks for cloudlets using SDN," in 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017, pp. 344-352.
[32] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On Reducing IoT Service Delay via Fog Offloading," IEEE Internet of Things Journal, vol. 5, pp. 998-1010, 2018.
[33] R. G. S.James, C.Jon, H.Pan, D.Christophe and C.Augustin, "The cambridge/haggle dataset ", 2009.