2021 20th International Conference on Ubiquitous Computing and Communications (IUCC/CIT/DSCI/SmartCNS) | 978-1-6654-6667-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/IUCC-CIT-DSCI-SMARTCNS55181.2021.00030

A Mobile Crowd Sensing Ecosystem based on Fog Computing Infrastructure

1st Liwei Lin
School of Computer Science and Mathematics
FuJian University of Technology
Fuzhou, Fujian, China
llw02@fjut.edu.cn

2nd Xia Lin
Technology Center of Fuzhou Customs District
Fuzhou, Fujian, China
379816876@qq.com

3rd Xiaoding Wang
College of Computer and Cyber Security
Fujian Normal University
Fuzhou, Fujian, China
wangdin1982@fjnu.edu.cn

***Abstract*—Mobile crowd sensing (MCS) applications are rapidly increasing and becoming a vital technique affecting the daily lives of people. However, large scale mobile sensing systems and the rapid growth of diverse applications have generated a huge volume of sensing data that need to be efficiently and effectively processed and managed. This poses the need of a new assistive platform powerful enough to boost the development of MCS. Researchers and practioners have considered to adopt cloud computing as an approach to promote such development. With the assistance of the cloud, massive data can be processed more effectively. However, centralized cloud datacenters still have some issues that need to be addressed: (1) High latency: since most of the sensed data need to be transmitted to the remote cloud, it introduced high latency to MCS applications. (2) Heavy load: the centralized cloud datacenter undertakes most of the load in the MCS system that may lead to hotspot. (3) Limited scalability: the centralized cloud datacenter may not be easily extended as the scale of MCS increases.**

To overcome these issues, in this paper, we propose a novel Mobile Crowd Sensing framework based on Fog computing (MCSF). Instead of using a centralized cloud, MCSF adopts fog computing infrastructure with decentralized micro datacenters, which can be used for large-scale deployment with better scalability. In MCSF, each micro datacenter of the fog just needs to handle the portion of the load within its service area. This reduces the oversubscription probability that occurs with the system of using a centralized cloud datacenter. Since the distributed micro datacenters are close to mobile nodes, fog computing is more effective than cloud computing for assisting MCS applications, as it reduces latency. MCSF enables the data to be preprocessed in micro datacenters and mobile nodes, which can reduce the amount of transmitted data and thereby reducing the transmission latency.

***Keywords*—mobile crowd sensing, fog computing, secondary allocation, community**

I. Introduction

Mobile devices developed in recent years incorporate various sensors from GPS, camera, to microphone, and have a light-weight portable size. These features enable mobile devices capable of data sensing and processing. It has been reported that the number of user-companioned devices, including smartphones, smart vehicles, wearable devices, and so on, reach approximately 6.58 per person in 2020 [1].

The quick growth of mobile devices in number and in various types has made a new service paradigm called Mobile Crowd Sensing (MCS) [2] viable, which extends the vision of participatory sensing by leveraging both participatory sensory data from mobile devices (offline) and user-contributed data from mobile social networking services (online). MCS has penetrated into various fields of peoples' life and work, including environment monitoring [3], transportation and traffic planning [4],healthcare, social recommendation [5], to name only a few. MCS can serve as a useful complementary tool for researching and working. Generally speaking, MCS explores complementary roles and presents a fusion or collaboration between machines and human intelligence in crowd sensing and computing processes [6] [7].

Traditionally, MCS relies on a centralized cloud datacenter for data aggregation and mining [8] [9], as shown in Figure1. All sensing data/results generated by mobile devices are sent to the remote cloud datacenter to be processed and used for responding to the requesters.

In MCS, a task execution procedure can be simplified as follows: a task is requested by a user (requester). Task is submitted to the remote cloud datacenter, and then is allocated to one or several other mobile devices (accepters) to participate and finish the task together. In this case, the task may need to be executed by devices located across different space (e.g. towns or cities), which is called cross-space task. A large number of cross-space tasks will generate large data flow in the network.

However, this centralized cloud-based MCS has the following drawbacks:

- **High latency:** Transmitting the data from a mobile device to a remote cloud datacenter and the response to the requesters add latency to MCS applications.
- **Heavy load:** MCS applications can generate huge

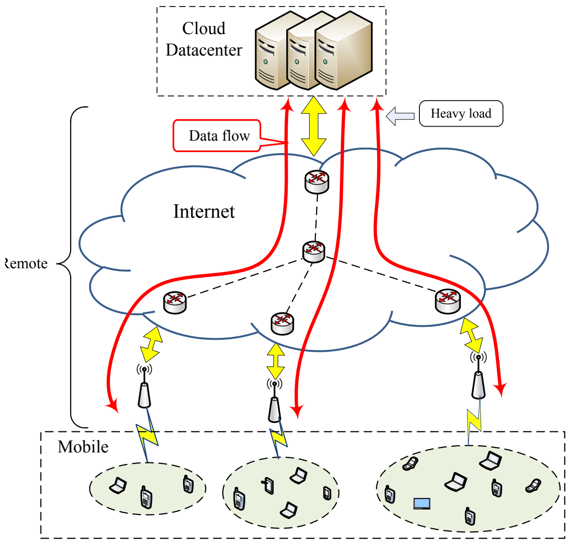

Fig. 1: Architecture of cloud based MCS system.

amount of data, including raw data collected from sensing devices and the respnses to the requests. All of these huge amount of data need to be sent to the centralized data center to be processed and only the datacenter could respnse or answer myriad and various users' requests. This may cause the problems of incast and outcast.

- **Limited scalability:** The centralized cloud datacenter may not be easily extended as the scale of MCS increases.

The fog computing is an emerging architecture [10] [11] that can be leveraged to overcome the issues addressed above by inserting a set of networked micro clouds in between the crowd datacenter and the mobile devices. Fog computing is a decentralized architecture, and it can extend cloud computing capacity to better serve mobile traffic applications. In this article, we propose a novel **M**obile **C**rowd **S**ensing ecosystem architecture, shown in Figure 2, based on **F**og computing (MCSF). In MCSF, tasks are directly managed by a local micro datacenter, which naturally reduce the latency. As fog is a decentralized architecture, the communication and computation loads are distributed to local micro datacenters, in order to reduce the probability of overload. Building micro datacenters needs much lower cost than building a large scale cloud datacenter and is with better scalability. Additional local micro datacenters, when needed, can easily join the fog network. This approach can improve the scalability of MCS system efficiently.

The rest of this paper will elaborate on the architecture of MCSF, and present the benefits of this architecture in terms of a better responsiveness and reduce power consumption, owing to combining fog computing and mobile crowd sensing applications. Then we introduce the API use case and aptitude of MCSF platform. We will finally summarize our contributions in proposing MCFS.

II. Overview of MCSF Ecosystem

The architecture of an MCSF system can be divided into three layers, namely crowd center layer, the fog network layer and the mobile layer. The top layer is crowd center. It provides resource information query service for fog network. The middle layer is fog network. It is the key layer of the MCSF ecosystem. Fog network consists of a set of networked micro datacenters. The main functions of fog network are mobile node management, servicing the task requester, preprocessing the sensing data, allocation of tasks and cooperation with other micro datacenters. The third layer is mobile layer. Each mobile device belongs to a mobile space dominated by a micro datacenter. A mobile node can be task requester or accepter. In MCSF, crowd center connects with fog network, and fog network connects with mobile nodes in mobile layer. As shown in Figure 2, we will introduce the layers in detail in the following three subsections.

A. Crowd Center

The crowd center is the top layer of MCSF system. The crowd center maintains the resource information of each micro datacenter in a fog network and provides the information query service for the micro datacenters in this network. In contrast to cloud data centers, the main duty of a crowd center is to maintain information of the micro datacenters in the fog network for query purposes. Resource information querying is one of the steps required for the task allocation procedure. This can overcome the shortcoming of centralized datacenters that is introduced before. When a task is launched, the agent in micro datacenter queries the resource information in the crowd center. When the state of a fog network changes (e.g., a new micro datacenter join the MCFS system), the crowd center will update its information to ensure accuracy of the service.

The centralized datacenter bears most of the computation and traffic load in cloud-based MCS ecosystem. Unlike centralized cloud datacenter, the crowd center only bears the query load. The bulk of the computation and communication load are offloaded to corresponding micro datacenters in the fog network. The flow generated by queries is much lighter than the sensing data flow, and the crowd center does not need to store and process the sensing data. The functions of fog network will be introduced in the next subsection.

B. Fog Network Layer

The fog network is the key component of the MCSF. It consists of interconnected micro datacenters located in different geographic locations. Compared with a cloud datacenter, micro datacenters are closer to mobile nodes, and a mobile crowdsensing management platform is deployed for the mobile nodes to accomplish crowdsensing tasks cooperatively between the micro datacenters.

Each micro datacenter dominates and serves a particular mobile space. For example, in Figure 2, micro datacenter communicates and dominates mobile space, and the mobile nodes in this mobile space are managed by the micro datacenter. Micro datacenter records mobile node information in the mobile space that it dominates. It manages the mobile nodes, the services for the task requesters, pre-processes the

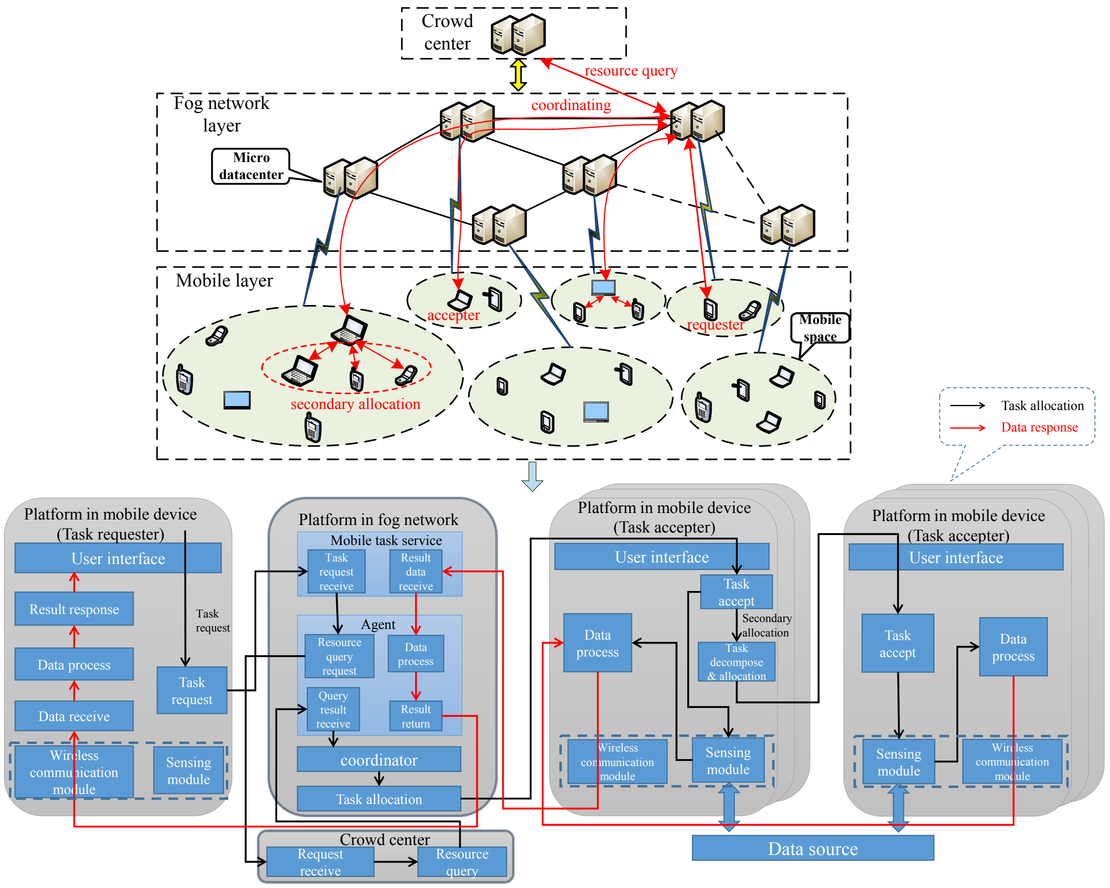

Fig. 2: Architecture and platforms of MCSF Ecosystem.

sensing data, allocates tasks and cooperates with other micro datacenters.

1) Mobile Node Management: Mobile node management is the basic issue of MCSF. Each mobile node in MCSF connects with its dominating micro datacenter directly through some wireless communication method (e.g., wifi or a base station). The micro datacenter records the mobile node information in its dominated mobile space.

Node Join: When a new mobile node joins an MCSF ecosystem, it will register on the fog network. Each mobile node reports its personal information, including its community tag, its device function configuration tag and its location information.

Node Quit: When a node applies to exit the ecosystem, the crowd center and micro datacenter will delete that node's information from the corresponding database.

Node Roaming: Mobile nodes may roam to other mobile areas. When a node leaves the current mobile space and moves to another mobile space, the dominating micro datacenter in the previous mobile space will delete the information of that node and the destination micro datacenter will record the node's information.

2) Servicing the Task Requester: When a requester submits a task, the dominating micro datacenter will create an agent service for the requester. The task requester will then offload the computation and communication load to the agent.

3) Pre-processing the Sensing Data: In many MCS applications, the task requester needs only the result rather than the sensing data. For example, for an environmental monitor, the task requester only requires knowledge of whether the monitored area is abnormal. Each micro datacenter receives data from its dominant mobile space, and pre-processes this data in order to obtain an intermediate result. These intermediate results are aggregated by the requesters agent. The agent pre-processes the results and sends the final result back to the requester. This pre-processing method can reduce the overall data transmission in network.

4) Allocation of Tasks: When a micro datacenter receives a task allocation command, it will allocate the tasks to the corresponding community mobile nodes. (We will introduce the community in next subsection)

5) Cooperation With Other Micro Datacenters: As a cross-space task cannot be finished within a single micro datacenter, multiple micro datacenters need to cooperate with each other to finish the task. Therefore, each micro datacenter has a local system to deal with multi micro datacenters cooperation. In a cooperation scenario, an efficient cooperation scheme will improve the performance of the overall system. The local system manages the mobile crowd sensing node in its area, and communicates with the local system platforms in the other micro datacenters and crowd centers.

An example will be given next to illustrate the micro datacenter cooperation procedure. Assume that there are two micro datacenters mcd_1 and mcd_2. A mobile user submits a task to mcd_1. mcd_1 queries the information from the crowd center, and then sends the cooperation request and task requirement information to mcd_2. Depending on the task requirement, the local system in mcd_2 selects appropriate mobile crowd sensing nodes for the task allocation. The task should be executed within the space dominated by mcd_2. In order to finish the task, mcd_1 and mcd_2 should cooperate with each other. In this procedure, micro datacenter mcd_1 cooperates with mcd_2 and crowd center.

C. Mobile Layer

The mobile layer combines mobile devices with human skills and knowledge, where mobile devices can produce and contribute participatory data by leveraging human mobility and intelligence and representing and conducting the humans ability. Mobile devices are labelled with the **community** tag based on its owners professional area. In an MCSF system, tasks can be classified into many professional areas, such as environmental monitoring and analysis, and medical diagnosis. These types of applications need special professional skills or professional software to be handled. These professional skills or software are used to organize the devices in terms of different communities. This community organization pattern can more precisely choose a member node, and thus solve the problem more professionally.

A mobile node can join several communities. In reality, a person may have more than one professional skill, and can manage problems arising from different areas. Thus, his (or her) mobile device can be capable of solving different problems that arise from different fields. The other advantage of community mobile nodes is that when they accept a task, they can use other idle mobile devices and allocate sub-tasks to them, which we call **secondary allocation**.

Secondary allocation: In MCSF, tasks will be allocated to corresponding community mobile nodes that have the professional skills or applications (software) to manage particular professional tasks. When a community mobile node accepts a task, it can re-allocate the task to another mobile node. If there are idle mobile devices in the same mobile space, the community mobile node can divide the task into several sub-tasks and then allocate these tasks to the idle devices. This method can balance energy consumption and can also reduce the running time of the task. This flexible organization can improve device utility and guarantee a better QoS. The community mobile node divides the task into serval smaller sub-tasks and employs other nodes to finish these sub-tasks who then send the intermediate results back to the community mobile node. The community mobile node performs final processing of the intermediate results and then sends the final result to the micro datacenter.

During this secondary allocation, the following factors should be considered for selection of the collaborative nodes in order to optimize the procedures.

- Constraints: The secondary allocation should satisfy the QoS and physical resources of the task requirement.
- Data Transmission: The community mobile node allocates sub-tasks to cooperative nodes. The intermediate data that is generated by these cooperative nodes will be returned to the community mobile node. The community mobile node processes this intermediate data, and returns the final result to the micro datacenter.
- Energy-aware: In order to balance energy consumption in the mobile area, the energy remaining in the mobile device should be taken into consideration and nodes should be chosen that have more remaining energy. This will prevent devices from running out of energy too quickly, allowing more tasks to be accommodated. Mobile devices will consume a lot of energy when transmitting the data, and the longer the communication distance, the higher the energy consumption. Therefore, community mobile nodes should attempt to find collaborative nodes which are close to the community mobile node to save energy consumption in the data transmission.
- Data Pre-process: In secondary allocation, the community node will process the intermediate data that is received from the collaborative nodes. The community mobile node obtains the intermediate results from the collaborative nodes and this intermediate data does not need to be transmitted to the micro datacenters, which also reduces the storage cost.

Additionally, a mobile node in MCSF can play the role of both the task requester and the task accepter. In an MCSF system, mobile nodes in a mobile area are dominated by a micro datacenter. Nodes can connect to the micro datacenter through some form of wireless communication such as wifi or a base station, or can also communicate with each other directly, such as using Bluetooth.

III. Crowd Senisng Procedure And API

In this section, we will introduce the overall procedure for task requests and responses. Figure 2 and 3 show the MCSF platform that provides the API that is used for the cooperative communication between mobile nodes, micro datacenters, and the crowd center.

A. Procedure

Task requester uses the $request()$ command to submit a task request to a micro datacenter, and informs the micro datacenter of the resources required. After receiving the request, the micro datacenter creates an agent service for the task requester and submits an application for resources query

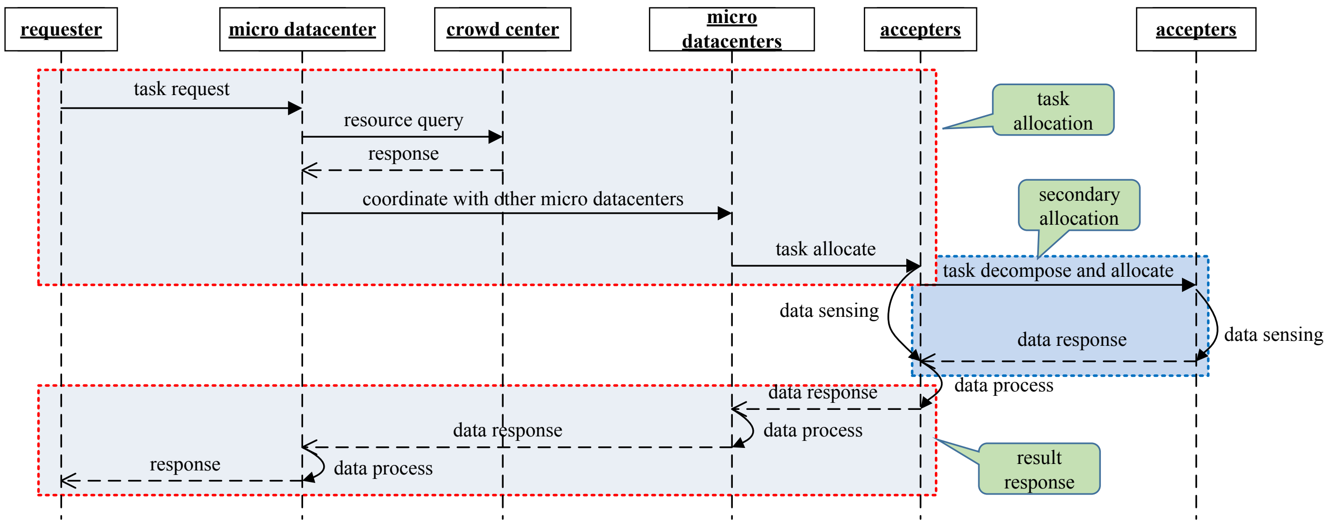

Fig. 3: The sequence diagram of task request and result response.

to the crowd center using the $resourceQuery()$ command. The crowd center returns the resource information (including the required resources location) to the agent. The agent sends the $coordinate()$ request command to the destination micro datacenters. request command to the destination micro datacenters. The coordination micro datacenters allocate the task to the community mobile nodes that they are dominating. These community mobile nodes can then perform a secondary allocation of the task to other mobile nodes based on the status of the neighboring mobile nodes. If idle neighboring mobile nodes exist, the community mobile node can decompose and can re-allocate the sub-tasks to these other nodes by secondary allocation. As shown in Figure 3, the acceptors sense the data and send data or results back to the requester using the same route that was used for the task request and allocation.

B. Other APIs

As well as the methods described above, the MCFS platform also provides applications with additional API commands that are listed here:

- $getRequiredResource(task)$ is used by the crowd center, and returns the micro datacenters information that is needed by the agent for coordination.
- $addTask(microDC\ mcd,\ task)$ is used by mcd a newly arriving $task$ to its task queue.
- $removeTask(microDC\ mcd,\ task)$ is used by mcd to remove a completed $task$ from its task queue.
- $deadline(task)$ return the deadline of $task$.
- $finished(taskId)$ return the finished task Id.
- $ifIdle(dev\ d)$ is used to detect the status of device d. If it is idle, it can accept the task or sub-task. Otherwise, no task will be allocated to it.
- $remainEnergy(dev\ d)$ return the remaining energy of device d.
- $getNeighborIdleNodes(dev\ d,\ task)$ returns the idle neighbor node list for d that $task$ needs..
- $decompose(task, idleList)$ is used to decompose the $task$ according to the $idleList$.
- $secondaryAllocate(subTask,\ dev\ d)$ is used to allocate the $subTask$ to the device d by secondary allocation.
- $releaseDev(dev\ d,\ task|subTask)$ is used to release the device d after the $task$ or $subTask$ is finished by that device.
- $getMoney(nodeId, money\ m)$ is used to add virtual money m to the balance of $nodeId$'s virtual money account, when $nodeId$ accept a task or sub-task.
- $costMoney(nodeId, money\ m)$ is used to minus virtual money m to the balance of $nodeId$'s virtual money account, when $nodeId$ request a task or sub-task.
- $speed(dev\ d)$ return the moving speed of mobile device d.
- $predictDestination(dev\ d)$ is used to predict the next destination mobile space of d during its roaming among the mobile spaces.
- $migrateService(service, microDC\ mcd1, microDC\ mcd2)$ is used to migrate the $service$ for mobile application from micro datacenter $mcd1$ to $mcd2$.
- $distance(dev\ d1, dev\ d2)$ return the distance between mobile device $d1$ and $d2$.
- $energyConsume(dev\ d, data, distance)$ is used to estimate the energy consumption when device d send $data$ with the $distance$ transmission distance.
- $hop(microDC\ mcd1, microDCmcd2)$ return the hops between micro datacenter $mcd1$ and $mcd2$.

C. API Use Cases and Aptitude

The general API framework can provide a rich and powerful development platform for sophisticated and comprehensive use cases with flexible and ubiquitous constraint. In this subsection, we will discuss some development examples and use cases on the MCSF framework with specified system design objectives.

1) Incentive Mechanism Design: Incentive mechanisms motivate the participation of people in crowdsourcing or human tasking systems. It is still an open problem to design an optimal model to encourage mobile nodes people to actually perform tasks and contribute meaningfully, which will be an important part of an MCS system [12] [13]. Good incentive mechanisms will inspire mobile nodes to participate in the task, and improve the performance of the MCS.

One common type of incentive mechanisms for raising user participation in MCSF. The developer can establish an ecosystem to encourage the mobile nodes to take part in the task through the virtual money as fortune reward to the service provider, and virtual money owner can purchase the MCS service in the ecosystem. For example, each mobile user can get the basic virtual money when it joins the MCSF. Virtual money can be used in this ecosystem. When mobile node requests a task, it should cost its own virtual money. Nodes will earn the virtual money through participating tasks or subtasks. Such complex incentive mechanism can be developed on the basic API of $getMoney()$ and $costMoney()$.

2) QoS Aware Task Allocation: Mobility is an inherent factor of mobile applications and close to the QoS. The mobility of nodes will cause more complex problems which may lower the performance of the MCSF. For example, the mobility of a mobile node may cause the task service to break. Additionally, locality-aware applications are also sensitive to mobility. This type of task allocations should consider the moving speed of the node.

Therefore, the task allocation should take mobility factor into consideration to devise a robust and efficient mechanism for MCSF system to further optimize the performance and improve the QoS of the MCSF system. In QoS aware task allocation algorithm,tasks will be allocated to the mobile nodes who will not move out the communication range before the task's deadline. This can ensure that the tasks be executed successfully. This algorithm can be developed on the basic API of $deadline()$ and $speed()$.

3) Cost Manage in Task Allocation: Completing the tasks, especially the cross-space tasks, requires serval micro datacenters to work together. The main cost in MCSF are mobile cost and fog cost, including energy, network, computing and storage cost. Due to the heterogeneous of micro datacenters and their dominate mobile resource, inefficient task allocation will increase the cost of whole system. Minimizing the overall cost in MCSF will largely affect the system's performance.

Reducing the cost of MCFS system can be conducted by efficient tasks allocation. To achieve such objective, the optimization algorithm can be designed in two aspects: (1)Selecting cost efficient cooperative micro datacenters in fog network layer. We could combine the routing scheme and reducing the number of hops of data transmission to reduce the bandwidth cost in fog network.The API $hop()$ can assist the routing scheme development. (2) Allocating/Secondary allocating the task/sub-task to mobile nodes with cost efficient scheme in mobile layer. For example, we can get distance between two devices, and estimate its energy consumption. So MCFS can adopt the allocation scheme with efficient cost. This function can be developed on the API $distance()$ and $energyConsume()$.

4) Seamless Handover in Mobile Nodes Roaming: In M-CSF system, mobile nodes may move from one mobile space to another space. Its service will also be moved from one micro datacenter to another one. This will cause the temporary interruption of the services or increase the service latencies.

Seamless handover in mobile nodes roaming is an important factor to improve user experience. Aiming to achieve this effect, MCSF needs to predict the destination micro datacenter of a mobile device, and migrate the service before its arriving, so as to decrease the service latencies. As soon as the mobile nodes arrive, the destination local service between mobile nodes and micro datacenter can be reestablished quickly. These function module can be developed on the basic API of $predictDestination()$ and $migrateService()$.

IV. Benefit Analysis of MCSF

Currently, cloud based architectures are popular for MCS. There is a lot of architecture research related to the integration of MCS and the cloud from different views. However, the general framework of cloud-based MCS architecture is similar. Therefore, we choose a recent work [14] for the purposes of comparison. The other recent different architecture for MCS is a mobile fog based framework [15]. In Table I. the difference between cloud based, mobile fog based and MCSF is given.

Based on the previous description, in this section we will discuss the benefits of MCSF as follows:

A. Benefits of a Fog Network

1) Scalability : The number of mobile devices is increasing rapidly, and scalability is very important for MCS. As discussed previously, a micro datacenter has a lighter weight and is more scalable than a cloud datacenter. This type of decentralized architecture is less limited by space and geographical position than a cloud datacenter. This flexible manner is suitable for mobile node deployment and organization.

2) Low Latency: A fog network is at the edge of the infrastructure and can closely service mobile nodes. Mobile devices can access the micro datacenter more easily than a cloud data center. Each micro datacenter dominates the mobile devices in their dominant area. This manner can be easily adapted for an increasing MCS.

3) Light Load: Fog computing is a decentralized infrastructure. Compared with centralized cloud-based MCS systems, MCSF scatter the overall traffic, storage and computation load. Each micro datacenter in a fog network only undertakes its own portion of the load. This decentralized method can effectively reduce the oversubscription probability which occurs in a cloud datacenter.

TABLE I: Comparison With Other Architectures

Crowd sensing architecture	Scalability	Data process	Latency	Classified management
Cloud based [14] [9] [16]	Scalability is limited by centralized datacenter infrastructure	Sensing data can be filtered or aggregated by specific mobile devices and remote cloud datacenter	Data is sent to remote cloud datacenter. This causes high latency	Does not take the diversity of tasks into consideration
Mobile fog based [15] [17]	More scalability than cloud based, but is still limited by centralized datacenter infrastructure	Sensing data can be analysed by specific mobile devices and remote cloud datacenter	Data is sent to remote cloud datacenter. This causes high latency	Does not take the diversity of tasks into consideration
MCSF	Decentralized fog network can be scaled much easier than centralized infrastructure	Data can be processed by both community mobile nodes and local micro datercenters. This manner is more flexible and lighten the load for datacenter.	Data is sent to nearby local micro datacenter. This can reduce latency	Mobile nodes and tasks are classified into communities, and task can be secondary allocated according to the local mobile devices' status. Such flexible allocation can reduce the task's response time and improve the resource utilization

4) Reduction of Transmission Data Size by Pre-processing in the Micro Datacenter: All sensing data will be transmitted to the micro datacenter dominating the mobile nodes. The micro datacenter can pre-process and fuse this data to reduce the data transmission size and latency. For both local and cross-space tasks, data pre-processing can reduce the data traffic in the fog networks.

B. Benefits of Secondary Allocation

Within different communities, each mobile node has its own professional skills and knowledge to manage the tasks, and thus the tasks will be allocated to corresponding community mobile nodes. After the community mobile nodes accept these tasks, they can divide the tasks into several sub-tasks, and perform secondary allocation to nearby idle nodes using wireless communication such as Bluetooth. This can balance the energy consumption and reduce the running time of the task.

1) Improvement of the Device Utility: Flexible organization can improve the device utility and guarantee a better QoS. Community mobile nodes divide the task into several smaller sub-tasks, and employ other nodes to finish these sub-tasks and then send the intermediate results back to the community mobile node.

2) Reduction of Transmission Data Size and Energy Consumption: In secondary allocation, the community mobile node becomes a secondary link, since nearby nodes are chosen for the allocation. A short distance will consume less power for the wireless communication. The node can also pre-process the data from the acceptors to reduce the overall transmitted data. This manner can further save the energy for mobile devices.

V. Conclusion

In this paper, we proposed a novel mobile crowdsensing ecosystem architecture based on fog computing (MCSF) aiming at overcome the shortcomings of cloud based MCS architectures. Compared with a centralized cloud, fog computing can be deployed more easily for large scale applications. In MCSF architecture, loads are distributed among the local micro datacenters. This decreases the probability of oversubscription that occurs to cloud datacenters. Therefore, fog computing can service MCS applications more efficiently and MCSF can reduce the latency of MCS applications. In the MCSF, the mobile nodes and tasks are classified within the community according to their professional area attributes. This classification enables more effective management of diverse applications. MCSF allows community mobile nodes and micro datacenters to pre-process the sensing data, which can further reduce the latency and energy consumption for mobile nodes.

Acknowledgment

This work is supported by National Natural Science Foundation of China under Grant No. U1905211 and 61702103, Natural Science Foundation of Fujian Province under Grant No. 2020J01167 and 2020J01169.

References

[1] D. Evans, "The internet of things how the next evolution of the internet is changing everything. cisco white papers, 2011."

[2] T. N. Nguyen and S. Zeadally, "Mobile crowd-sensing applications: Data redundancies, challenges, and solutions," *ACM Transactions on Internet Technology (TOIT)*, vol. 22, no. 2, pp. 1–15, 2021.

[3] N. Maisonneuve, M. Stevens, M. E. Niessen, and L. Steels, "Noisetube: Measuring and mapping noise pollution with mobile phones," in *Information Technologies in Environmental Engineering*. Springer, 2009, pp. 215–228.

[4] Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao, "Automatically characterizing places with opportunistic crowdsensing using smartphones," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 481–490.

[5] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher, "Greengps: a participatory sensing fuel-efficient maps application," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 151–164.

[6] B. Guo, Z. Yu, X. Zhou, and D. Zhang, "From participatory sensing to mobile crowd sensing," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*. IEEE, 2014, pp. 593–598.

[7] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, R. Huang, and X. Zhou, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 7:1–7:31, Aug. 2015. [Online]. Available: http://doi.acm.org/10.1145/2794400

[8] X. Hu, X. Li, E. C.-H. Ngai, V. C. Leung, and P. Kruchten, "Multidimensional context-aware social network architecture for mobile crowdsensing," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 78–87, 2014.

[9] A. Antonic, K. Roankovic, M. Marjanovic, K. Pripuic *et al.*, "A mobile crowdsensing ecosystem enabled by a cloud-based publish/subscribe middleware," in *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*. IEEE, 2014, pp. 107–114.

[10] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[11] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Application management in fog computing environments: A taxonomy, review and future directions," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–43, 2020.

[12] K. Han, C. Zhang, J. Luo, M. Hu, and B. Veeravalli, "Truthful scheduling mechanisms for powering mobile crowdsensing," *IEEE Transactions on Computers*, vol. 65, no. 1, pp. 294–307, 2016.

[13] X. Zhang, Z. Yang, W. Sun, Y. Liu, S. Tang, K. Xing, and X. Mao, "Incentives for mobile crowd sensing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 54–67, 2016.

[14] A. Antonić, M. Marjanović, K. Pripužić, and I. P. Žarko, "A mobile crowd sensing ecosystem enabled by cupus: Cloud-based publish/subscribe middleware for the internet of things," *Future Generation Computer Systems*, vol. 56, pp. 607–622, 2016.

[15] P. P. Jayaraman, J. B. Gomes, H.-L. Nguyen, Z. S. Abdallah, S. Krishnaswamy, and A. Zaslavsky, "Scalable energy-efficient distributed data analytics for crowdsensing applications in mobile environments," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 3, pp. 109–123, 2015.

[16] M. Marjanović, L. Skorin-Kapov, K. Pripužić, A. Antonić, and I. P. Žarko, "Energy-aware and quality-driven sensor management for green mobile crowd sensing," *Journal of Network and Computer Applications*, vol. 59, pp. 95–108, 2016.

[17] P. P. Jayaraman, J. B. Gomes, H. L. Nguyen, Z. S. Abdallah, S. Krishnaswamy, and A. Zaslavsky, "Cardap: A scalable energy-efficient context aware distributed mobile data analytics platform for the fog," in *East European Conference on Advances in Databases and Information Systems*. Springer, 2014, pp. 192–206.