iTaskOffloading: Intelligent Task Offloading for a Cloud-Edge Collaborative System

Yixue Hao, Yingying Jiang, Tao Chen, Donggang Cao, and Min Chen

Abstract

With the development of technologies such as the Internet of Things and artificial intelligence, mobile applications are becoming more and more intelligent. Compared to traditional applications realized by mobile cloud computing technology, these novel applications have a higher requirement for a task offloading scheme. However, the traditional task offloading schemes are hard pressed to meet latency and personalization requirements of these new applications. For intelligent application, how to realize personalized and fine-grained task offloading is still a challenging problem. Therefore, we propose a scheme called intelligent task offloading (iTaskOffloading) for a cloud-edge collaborative system, which can provide personalized task offloading. To be specific, we first propose the architecture of iTaskOffloading which includes the local device layer, edge cloud layer, remote cloud layer, and cognitive engine. Then we analyze the method of iTaskOffloading, which contains coarse-grained computing and fine-grained computing. Finally, we build a testbed to evaluate the proposed iTaskOffloading scheme using a typical intelligent application of emotion detection. The experimental results show that compared to the traditional cloud computing scheme, iTaskOffloading has less task duration.

Introduction

Nowadays, with the development of technologies such as the Industrial Internet of Things (IIoT) [1] and artificial intelligence (AI), current mobile applications (e.g., virtual reality and emotion detection) are more and more intelligent. These intelligent applications need not only large-scale and personalized data but also support for AI algorithms. Since mobile devices suffer from limited battery and computing capacity, it is necessary to offload these intelligent applications to the cloud [2]. However, when a large number of devices are connected to the cloud, large latency may be incurred because the cloud is far from users [3, 4].

In order to solve these problems, both academia and industry propose edge computing [5, 6]. Compared to cloud computing [7], edge computing satisfies the needs of low latency of users' tasks by deploying servers on the edge. At present, most of the interesting studies on edge computing focus on solving the problem of energy efficiency from the perspective of communication [8, 9]. However, they do not consider how to solve the task offloading problem of intelligent application with personalized data and an AI algorithm.

Aiming at the problem of task offloading, each intelligent application can act as an independent task [10]. By taking the personalized data of users into consideration, the processing of each task can be different [11]. Correspondingly, the computing methods of the tasks in various network conditions need to be differentiated. Thus, for the intelligent application, we need to consider using intelligent technology to decide "What" is needed and "How" to compute the task. In addition, to achieve efficient task execution, we need to choose a computing location among local device, edge cloud, and remote cloud (i.e., "where to compute"). However, existing works do not consider the problem of this intelligent application task offloading. Therefore, for intelligent application, how to design a personalized task offloading strategy is still a challenging problem.

In this article, in order to solve the problem of task offloading in intelligent application, we propose an intelligent task offloading (iTaskOffloading) scheme for a cloud-edge collaborative system. To be specific, we first combine the cognitive engine with the traditional cloud-edge collaborative system, and propose the architecture of iTaskOffloading. This architecture can not only recognize the resources from local device, edge cloud, and remote cloud, but also understand the task of intelligent application. Then, based on the above proposed architecture, we give the method of iTaskOffloading and design a scheme of fine-grained task offloading for the separability of intelligent application tasks to enable personalized task offloading. Finally, we build a real testbed and demonstrate the performance of iTaskOffloading.

In summary, the main contributions of this article include:

- Based on the cloud-edge collaborative system, we propose the iTaskOffloading architecture, which includes the local device layer, edge cloud layer, remote cloud layer, and cognitive engine. Using the cognitive engine, we can realize the cognition of resources and the intelligent application task.
- We give the method of iTaskOffloading, which includes what to compute, how to compute, and where to compute the task. Furthermore, in the scheme, we propose the fine-grained task offloading, which can realize personalized task offloading.

Digital Object Identifier:
10.1109/MNET.001.1800486

Yixue Hao, Yingying Jiang, and Min Chen are with Huazhong University of Science and Technology; Tao Chen is with VTT Technical Research Centre of Finland; Donggang Cao (Corresponding author) is with Peking University.

- Based on the intelligent application of emotion detection, a real testbed is built to verify iTaskOffloading. The experimental results show that the iTaskOffloading scheme has less latency than traditional cloud computing.

The remainder of this article is organized as follows. In the next section, we propose the system architecture. Following that, an offloading strategy is explained. The testbed and related experiments are then presented. Further research topics are then listed. Finally, our conclusion is given.

System Architecture

In this section, we give the system architecture, including the local device layer, edge cloud layer, remote cloud layer, and cognitive engine.

System Overview

The system architecture includes the local device layer, edge cloud layer, remote cloud layer, and cognitive engine, as shown in Fig. 1. In order to illustrate the system overview, we use emotion detection as an example. First, the user collects multi-modal emotion data through wearable devices. Then, after getting the emotion data, the user needs to offload the computing task. Consider the personalization of emotion data. When different users have various requirements on the latency of emotion detection, the emotion computing task should be offloaded to different locations (i.e., local device, edge cloud, and remote cloud) for further processing. The cognitive engine determines the personalized offloading strategy. Finally, after the emotion task is processed, the result will be fed back to the user, and the user will obtain a personalized healthcare service.

Local Device Layer

The local layer includes local devices such as smartphone, robot, and wearable devices, which can collect data of intelligent applications. Furthermore, due to the relatively low computation capability and storage capability of local devices, when a user has excessive requests or a computing task of great complexity, it is not suitable to process the data on local devices. However, local devices are the closest device to the user, which can produce low communication latency, so it is suitable for processing simple tasks that are very sensitive to latency [12]. Furthermore, since the computing capacity of a local device is limited, the AI algorithm deployed in the local device layer is simple.

Edge Cloud Layer

The edge cloud layer, as a middle layer, can process a computing task. Edge cloud is composed of numerous edge services with computing and storage capability. The edge server can be deployed in a gateway, router, and so on. The computing and storage ability of edge cloud is weaker than that of remote cloud. However, the communication latency is much lower than that of remote cloud. Generally speaking, mobile devices are connected to edge cloud through a single-hop wireless network. Thus, the communication latency is decreased greatly to satisfy the demand of a computing task with latency sensitivity. The computing ability of edge cloud is better than that of a local device. Thus, we deploy a complex AI algorithm on the edge node to process the task.

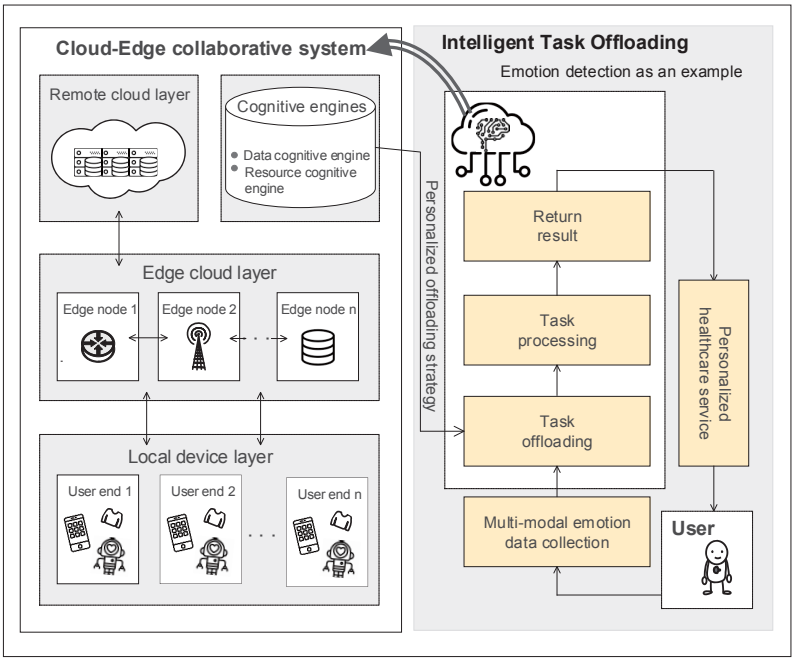

FIGURE 1. System architecture.

Remote Cloud Layer

A remote cloud represents a large data center that is distanct from the user. It can deploy a high-performance AI algorithm and stores a large amount of user historical analysis data. Thus, it can provide highly precise computing and be used for computation-intensive tasks. Taking emotion detection as an example, emotion data is private with the heterogeneous features. Thus, it records a behavioral habit of a user and maps it for further computing rules to achieve more accurate prediction and better user service using the user features or behavior cognition. However, the shortcoming of cloud computing is high latency due to the far distance between the cloud data center and the user. Furthermore, we give the comparison of local computing, edge computing, and cloud computing from the computing capacity, storage capacity, distance between the user and server, communication latency, and algorithm deployment variety, as shown in Table 1.

Cognitive Engine

By using the software-defined networking (SDN) technology and AI algorithm, we can deploy a cognitive engine in three layers of a network framework. Cognitive engines can be divided into two types: the resource cognitive engine and the data cognitive engine, as shown in Fig. 1. The data cognitive engine first processes real-time data flow in the network environment, and introduces data analysis in edge cloud and remote cloud. Then it carries out task logic intelligently, realizes cognition of task data and resource data through all kinds of cognitive computing methods, allocates dynamic guidance resources, and provides cognition services. Through SDN, network functions virtualization (NFV), self-organizing networks (SONs), and network slicing

Computing type	Computing capacity	Storage capacity	Distance between the user and server	Communication latency	Algorithm deployment variety
Local computing	Low	Low	No hop	Low	Low
Edge computing	Medium	Medium	One hop	Medium	Medium
Cloud computing	High	High	Multiple hops	High	High

TABLE 1. Comparison of local, edge, and cloud computing.

technology, the resource cognitive engine can transfer comprehensive resource data to the data cognitive engine, to realize the perception of the computing resources, communication resources, and network resources (e.g., network type, communication quality). Meanwhile, it receives the analysis result from the data cognitive engine and guides the real-time dynamic optimization and allocation of resources.

Methods of iTaskOffloading

In this section, we give the method of iTaskOffloading, which includes "what to compute," "how to compute," and "where to compute."

What to Compute

The iTaskOffloading scheme is for the intelligent application; thus, in order to illustrate what to compute, we give emotion detection as an example, as shown in Fig. 2. The computing data include speech emotion data (D1), facial emotion data (D2), physiological data (D3), and so on. Specifically, the emotion detection needs input of the personal time series data (e.g., speech, facial, and physiological data) and uses a deep learning algorithm to detect the user's emotion. In theory, using multi-dimensional data and large amounts of historical data can guarantee a high-precision emotion recognition rate. Thus, the emotion detection involves the multi-level computing issues including computing amount and its complexity at each level. Analyzing this kind of task in real time, some data (e.g., physiological data) can use a simple AI algorithm, while other data (e.g., facial data) can use a complex AI algorithm. Thus, we can divide the tasks requiring to be offloaded into two categories: light task load and heavy task load. Light task load means that the amount of data to be offloaded is small, the required AI algorithm is simple, and the number of concurrent user requirements is lower. Correspondingly, heavy task load means that the amount of data to be offloaded is large, the required AI algorithm is complex, and the number of concurrent user requirements is high. Therefore, compared to a traditional task, intelligent task application is more compute-intensive and needs optimal task offloading.

How to Compute

Given the basic scope of a computing task, the next step is to consider how to do the computation. Regarding the question of "how to compute," we propose two concepts, coarse-grained task offloading and fine-grained task offloading, to increase the computational flexibility of the iTaskOffloading method.

Coarse-Grained Task Offloading: As shown in Fig. 2, depending on whether the task is separable, we can divide the methods of task offloading into coarse-grained task offloading and fine-grained task offloading. When the task is inseparable, it corresponds to coarse-grained. It is worth noting that an inseparable task means that for a specific task, it can only be executed in either a local device, edge cloud, or remote cloud, but cannot be divided into different subtasks to be executed in different locations. For intelligent applications, the corresponding tasks can generally be divided into different subtasks. Therefore, it is necessary to design fine-grained task offloading. The specific introduction is given as follows.

Fine-Grained Task Offloading: The intelligent application task is a complex task that can be divided into multiple small subtasks. For example, a speech emotion processing task can be divided into three subtasks: voiceprint recognition, speech recognition, and speech emotion detection. Since each subtask has its own characteristics, it requires different communication, computing, and storage resources. Based on the divisibility of tasks, these subtasks can be suitable for flexible computing by local device, edge cloud, and remote cloud. Under this condition, the task can implement fine-grained task offloading.

Specifically, we can divide the intelligent application task into multiple different subtasks, each of which can be computed at a different location. For instance, voiceprint recognition can be computed on the edge cloud, while speech recognition can be done on the remote cloud. Task offloading is based on the characteristics of the computing requirement of the subtasks and the current computing resources of edge cloud and remote cloud. In this way, computing resources can be fully utilized, ensuring recognition accuracy, and tasks can be offloaded distributively to shorten the time delay.

Furthermore, considering the ultra-low latency and ultra-high reliability requirements of intelligent application users, we use cognitive engine and SDN technology to provide personalized task offloading. Specifically, when mobile devices offload the tasks, it sends control instructions to the edge cloud. The feedback accuracy and task duration of different computing methods deployed on the edge cloud and remote cloud are different. Since the duration of task computing on a cloud is long, we can first satisfy the basic needs of users by low-resolution computing and use the edge cloud to calculate and feed back the rough recognition results to satisfy users' need for a fast response in real time. When the results of cloud computing are complete, we add high-resolution computing as a follow-up more detailed and precise analysis. Thus, fine-grained computing can fully meet the characteristics of a user's personalized computing tasks.

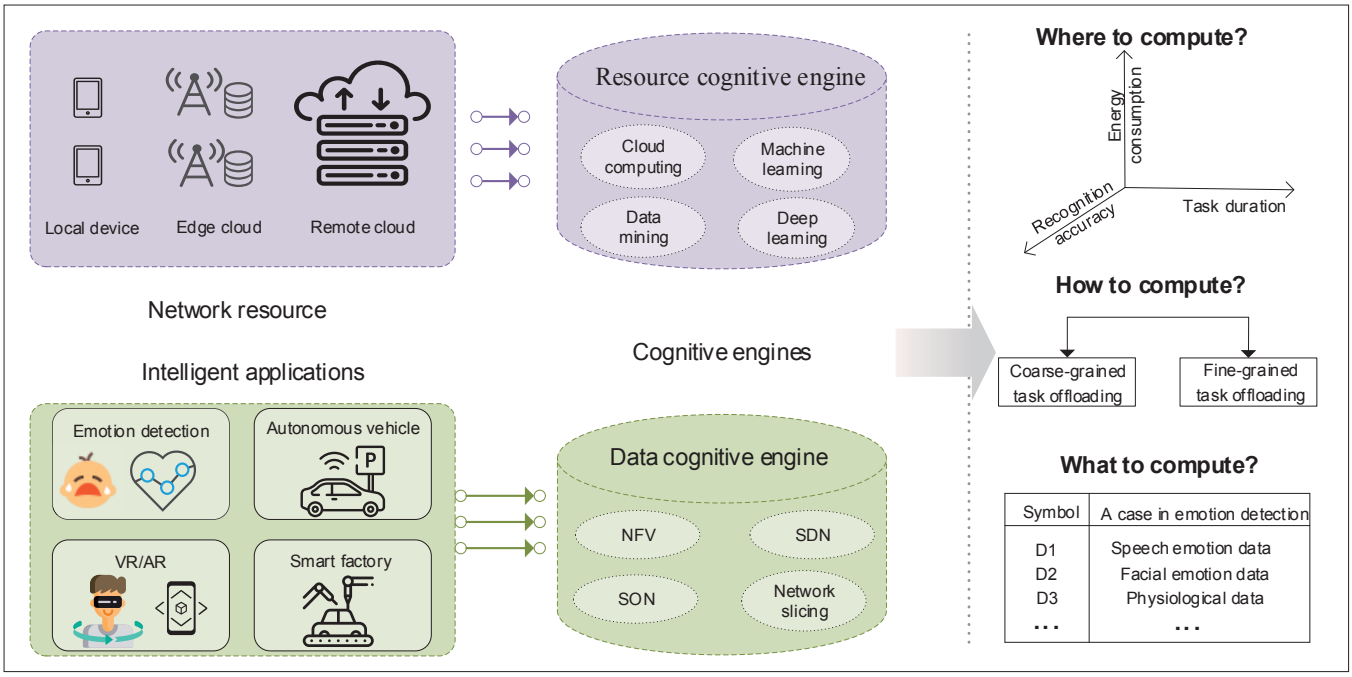

FIGURE 2. iTaskOffloading methods: What to compute? How to compute? Where to compute?

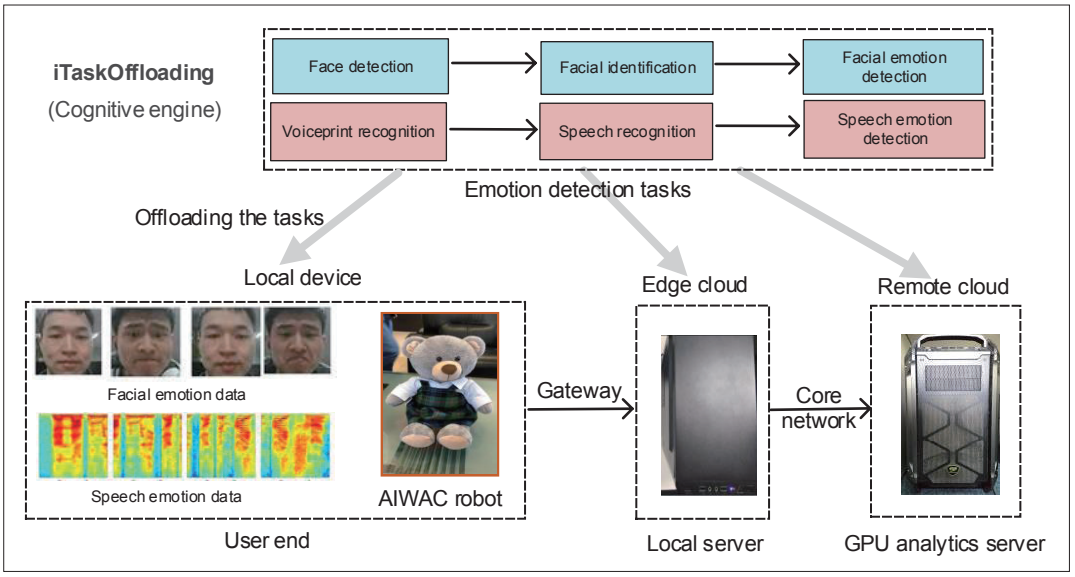

FIGURE 3. System testbed based on emotion detection.

Where to Compute

The selection of where to offload a computing task (i.e., local device, edge cloud, or remote cloud) is always based on the individual optimization objectives of users, as shown in Fig. 2. Specifically, compared to traditional task offloading optimization objectives, the intelligent application optimization objectives include reducing the task duration and energy consumption and improving recognition accuracy. It should be noted that for fine-grained task offloading, the optimization objectives of every subtask may be different. That is because some subtasks focus more on task duration, but some subtasks focus on energy consumption. Furthermore, where to offload the subtasks depends on the optimization objectives of the whole task. Therefore, we need to perform a joint optimization of multiple subtasks and one whole task, and ultimately determine where the subtasks should be processed.

Considering different computing and storage capabilities of local device, edge cloud, and remote cloud, different computing AI algorithms are deployed and adopted at different locations. Furthermore, the heterogeneous edge cloud with different computing and storage capability, which are in the edge cloud layer, can be connected to each other and offload tasks distributively so that task duration can be reduced.

Testbed and Experiment Result

In this section, we build a testbed for emotion detection to evaluate the iTaskOffloading scheme.

System Testbed

Emotion Detection Task: In this article, we regard emotion detection as an intelligent application to verify the proposed iTaskOffloading scheme, as shown in Fig. 3. To be specific, for the complex

Computing type	Platform	Operating system	Hardware	
Local computing	Robot	Android 4.4	CPU	4-core,1.2 GHz
			DDR memory	1 GB DDR3 SDRAM
			NAND flash	32G NAND Flash
Edge computing	Local server	CentOS 7	CPU	8-core, 3.4 GHz
			DDR memory	16 GB DDR3 SDRAM
			Hard disk	1050G NAND Flash
Cloud computing	GPU analytics server	Ubuntu 16.04	GPU	NVIDIA GTX1080ti*2
			DDR memory	32 GB DDR3 SDRAM
			CPU	6-core, 3.5 GHz

TABLE 2. Configuration of the testbed.

task of users' emotion detection, we utilize two data models that include facial expression and speech to detect a user's emotion. In order to accomplish fine-grained task offloading, we divide the task of facial expression recognition into three subtasks: face detection, facial identification, and facial emotion detection. Specifically, first we preprocess the data to perform face detection and remove background and irrelevant regions. Then the user's face needs to be identified, and other irrelevant faces need to be removed. Finally, an emotion detection algorithm is used to recognize facial expression.

Similarly, we divide the task of speech emotion recognition into three subtasks: voiceprint recognition, speech recognition, and speech emotion detection. Specifically, first we preprocess the data to perform voiceprint recognition and obtain the user's voice required to detect in multi-user conditions. Then the voice of the user is analyzed, and finally speech emotion will be identified by employing an emotion detection algorithm. After completing facial expression and speech emotion recognition, we will feed back the recognition results to the user.

In the experiment, in order to achieve multi-layer deployment of the emotion detection algorithm, for facial expression recognition we use two different sizes of a deep neural network (in this article, we use VGG16 [13]) algorithm to perform emotion recognition, where the size of complex model is 126 MB and the simple model is 4 MB. For speech emotion recognition, we adopt two different sizes of a deep neural network (in this article, we use AlexNet [14]) algorithm to perform speech recognition, where the size of the complex model is 24.9 MB while the simple model is 4.6 MB.

Hardware Platform: Figure 3 shows the hardware platform of the testbed. From the figure, we can see that the hardware platform is composed of three parts: AIWAC robot [15], local server (i.e., edge cloud), and GPU analytics server (remote cloud). Specifically, the AIWAC robot collects complex multi-modal emotional data. Its system is an Android 4.4 operating system with limited computing and storage capacities. Local service whose operating system is CenOS 7 has medium storage and computing resource. We use a GPU analytics server whose operating system is Ubuntu 16.04 as the remote cloud, which is equipped with strong computing and storage capabilities. Table 2 lists the specific hardware parameters of local device, edge cloud, and remote cloud. In the experiment, we set the upper bound of the number of concurrent tasks that can be handled by local device to 5, edge cloud to 60, and remote cloud to 500.

Experimental Results

Based on the above experimental platform, aimed at the emotion detection task, we perform experiments for the proposed fine-grained task offloading and coarse-grained task offloading, and make a comparison between iTaskOffloading and traditional cloud computing. In the experiment, we use task duration as the evaluation index. Furthermore, we deploy a simple speech and facial recognition model on local device and edge cloud, and a complex speech and facial recognition model on remote cloud. The result is shown in Fig. 4.

From Fig. 4a, we can see that in the case of a light task load, the iTaskOffloading scheme has better performance than cloud computing, and the fine-grained task offloading surpasses the coarse-grained task offloading by nearly 10 percent. This is because iTaskOffloading can flexibly offload tasks to edge cloud or remote cloud according to the types of the tasks, while cloud computing offloads all tasks to the cloud, resulting in large transmission delay. Furthermore, fine-grained task offloading can divide the task to be processed into multiple subtasks, and flexibly deploy the subtasks in the local device, edge cloud, and remote cloud according to the characteristics of the subtasks, which reduces the task duration. When the task load increases to the heavy level in Fig. 4b, which is the most frequent situation in reality, the performance of the fine-grained task offloading scheme is significantly better (by 30 percent) than that of the cloud computing. This can be explained by the fact that as the amount of tasks increases, fine-grained task offloading is more intelligent and personalized in task processing, thus reducing the task duration. Therefore, iTaskOffloading shows relatively good performance.

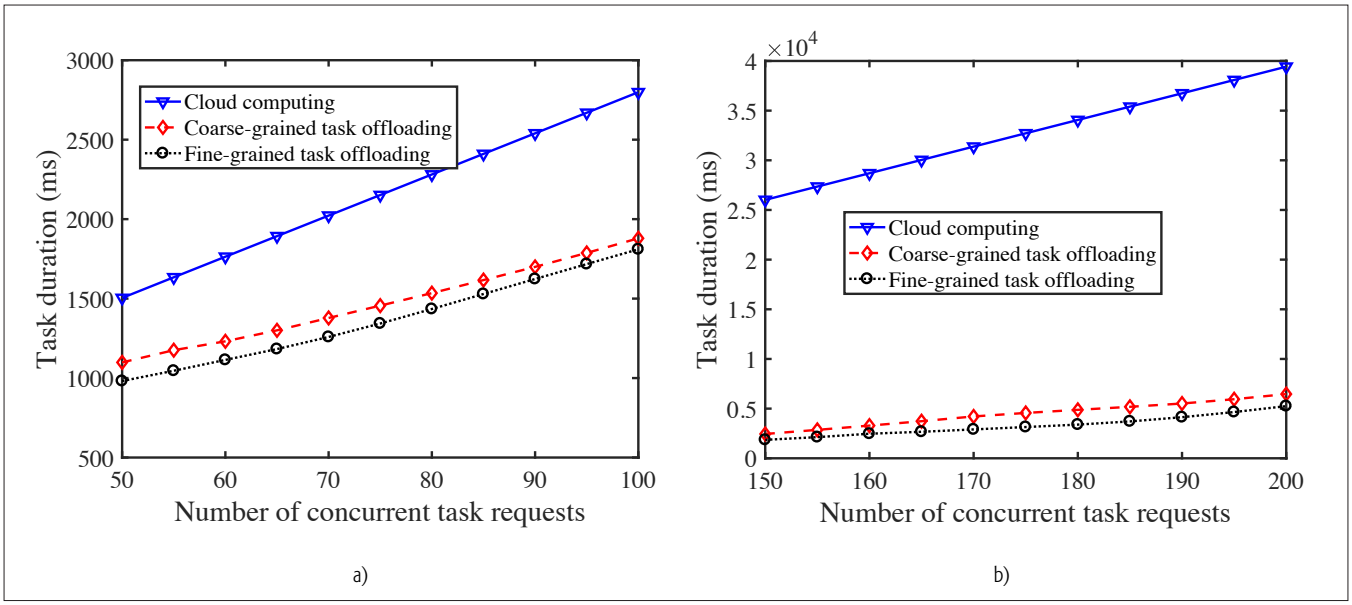

FIGURE 4. Comparison of task duration against different numbers of concurrent task requests with: a) light task load; b) heavy task load.

Open Research Issues

Management of Heterogeneous Resource

Although a multi-layered network can provide adaptable and adjustable network resources, it also faces the problem of managing the heterogeneous resources (i.e., communication, storage, and computing resource). This is because there are numerous devices in the network, such as mobile phones, personal computers, and wearable devices, and these devices can be distributed at different locations in the network with constraints of ultra-low latency and ultra-high reliability. Thus, we need to consider how to effectively and automatically manage different types of resources. Furthermore, within the edge network, resource scheduling and task handover also need to be considered. Moreover, the remote cloud and edge cloud need to balance communication, storage, and computation resources.

Intelligent Protocol for Mobility Management

More intelligent protocols are necessary for intelligent application task offloading. In the case of multiple users, regarding users' task complexity, delay sensitivity, and other factors, different users should be assigned different priorities. Network resources can be allocated based on the priority to improve the overall quality of experience. Furthermore, considering user mobility, incomplete acquisition of the information on user trajectory may lead to the inaccurate prediction of user mobility, resulting in higher latency or even failure of task offloading. With the increasing number of mobile devices, one of the potential solutions is to use statistical information on user trajectory to selectively pre-fetch possible data and establish an intelligent protocol based on mobility trajectory prediction. Finally, task offloading may cause high latency or task failure due to unstable network connections and constant change of network access points. Therefore, more intelligent protocols are required to guarantee smooth and successful resource allocation.

Security and Privacy

Since local devices have limited computing capacity and power, they often need to offload tasks to the edge cloud or remote cloud for further processing. In this process, the problem of privacy leakage and security challenges is urgent and serious. There are some problems need to be solved, for example, how to protect the task on device so that unauthorized people cannot access it, how to ensure that a task is securely and reliably shared between local devices and edge cloud or remote cloud, and how to securely store the task data on edge cloud and remote clouds.

Conclusion

In this article, we first introduce the architecture of iTaskOffloading which includes local device, edge cloud, remote cloud, and cognitive engine. Then we discuss the method of iTaskOffloding, which includes coarse-grained task offloading and fine-grained task offloading. After that, taking emotion detection as an example, we build a real testbed to verify the efficiency of the AITO scheme. Finally, we list some open research issues, which include management of heterogeneous resource, intelligent protocol for mobility management, security, and privacy.

Acknowledgments

This work is supported by the National Key R&D Program of China (2017YFE0123600). The work by Dr. Yixue Hao was partially supported by the National Natural Science Foundation of China (Grant No. 61802138), and the China Postdoctoral Science Foundation (No. 2018M632859, No. 2019T120657). The work by Dr. Tao Chen was partially funded by the EU Horizon 2020 programme under grant agreements no. 814956 (5G-DRIVE).

References

[1] Y. Wang *et al.*, "Traffic and Computation Co-Offloading with Reinforcement Learning in Fog Computing for Industrial Applications," *IEEE Trans. Industrial Informatics*, vol. 15, no. 2, 2019, pp. 976–86.

[2] H. Shi and Y. Li, "Discovering Periodic Patterns for Large Scale Mobile Traffic Data: Method and Applications," *IEEE Trans. Mobile Computing*, vol. 17, no. 10, 2018, pp. 2266–78.
[3] M. Chen *et al.*, "Opportunistic Task Scheduling over Co-Located Clouds in Mobile Environment," *IEEE Trans. Service Computing*, vol. 11, no. 3, 2018, pp. 549–61.
[4] H. Li, K, Ota, and M. Dong, "Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing," *IEEE Network*, vol. 32, no.1, 2018, pp. 96–101.
[5] M. Chen *et al.*, "A Dynamic Service-Migration Mechanism in Edge Cognitive Computing," *ACM Trans. Internet Technology*, vol. 19, no. 2, 2019, Article 30.
[6] H. Huang, and S. Guo, "Proactive Failure Recovery for NFV in Distributed Edge Computing," *IEEE Commun. Mag.*, vol. 57, no. 5, May 2019, pp. 131–37.
[7] R. Shea *et al.*, "Cloud Gaming: Architecture and Performance" *IEEE Network*, vol. 27, no. 4, July/Aug. 2013, pp. 16–21.
[8] M. Chen and Y. Hao, "Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network," *IEEE JSAC*, vol. 36, no. 3, Mar. 2018, pp. 587–97.
[9] M. Chen *et al.*, "Cognitive Information Measurements: A New Perspective," *Info. Sciences*, vol. 505, 2019, pp. 487–97.
[10] Y. Liu *et al.*, "On the Resource Trade-off of Flow Update in Software-Defined Networks," *IEEE Commun. Mag.*, vol. 54, no. 6, June 2016, pp. 88–93.
[11] Y. Liu, M. Lee, and Y. Zheng, "Adaptive Multi-Resource Allocation for Cloudlet-Based Mobile Cloud Computing System," *IEEE Trans. Mobile Computing*, vol. 15, no. 8, 2016, pp. 2398–2410.
[12] D. Xu *et al.*, "A Survey of Opportunistic Offloading," *IEEE Commun. Surveys & Tutorials*, vol. 20, no. 3, 2018, pp. 2198–2236.
[13] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv preprint arXiv: 1409.1556, 2014.
[14] S. Zhang *et al.*, "Learning Affective Features with a Hybrid Deep Model for Audio-Visual Emotion Recognition," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 28, no. 10, 2018, pp. 3030–43.
[15] M. Chen *et al.*, "AIWAC: Affective Interaction Through Wearable Computing and Cloud Technology," *IEEE Wireless Commun.*, vol. 22, no. 1, Feb, 2015, pp. 20–27.

Biographies

Yixue Hao (yixuehao@hust.edu.cn) received his B.E. degree from Henan University, China, and his Ph.D degree in computer science from Huazhong University of Science and Technology (HUST), China, 2017. He is currently working as a postdoctoral scholar in the School of Computer Science and Technology at HUST. His research includes 5G networks, the Internet of Things, and mobile edge computing.

Yingying Jiang received her B.E. degree from the School of Information and Safety Engineering, Zhongnan University of Economics and Law (ZUEL), China, in 2017. Currently, she is a Ph.D candidate at the Embedded and Pervasive Computing (EPIC) Lab in the School of Computer Science and Technology, HUST. Her research includes healthcare big data, cognitive learning, and so on.

Tao Chen [SM'09] (tao.chen@vtt.fi) received his B.E. degree in telecommunications engineering from Beijing University of Posts and Telecommunications, China, in 1996, and his Ph.D. degree from the University of Trento, Italy, in 2007. He is currently a senior researcher with the VTT Technical Research Centre of Finland. He is the Project Coordinator of the EU H2020 5G PPP COHERENT Project. His current research interests include software-defined networking for 5G mobile networks, dynamic spectrum access, social-aware mobile networks, and energy efficiency and resource management in heterogeneous wireless networks.

Donggang Cao (caodg@pku.edu.cn) is a research professor at the Software Institute, School of Electronics Engineering and Computer Science, Peking University, Beijing. He received his Ph.D. degree in computer software and theory from Peking University in 2004. His research interests include system software, parallel and distributed computing, cloud computing, and so on.

Min Chen [SM'09] (minchen2012@hust.edu.cn) has been a full professor in the School of Computer Science and Technology at HUST since February 2012. He is Chair of the IEEE Computer Society STC on Big Data. His Google Scholars Citations reached 17,600+ with an h-index of 66. He received the IEEE Communications Society Fred W. Ellersick Prize in 2017 and the IEEE Jack Neubauer Memorial Award in 2019. His research focuses on cyber physical systems, IoT sensing, 5G networks, SDN, healthcare big data, and so on.