A Survey of Computation Offloading in Edge Computing

Tao Zheng[1,2], Jian Wan[1], Jilin Zhang[1], Congfeng Jiang[1], Gangyong Jia[1]

1 School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China
2 College of Information Science and Technology, Zhejiang Shuren University, Hangzhou 310015, China

***Abstract*—Computation offloading is a critical technology for emerging edge computing and Internet of things (IoT). It is views as a solution of the limited resources of IoT devices by offloading tasks to other devices or servers. It can brings many benefits, such as prolonging battery life, reducing the latency and improving application performance. In practice, the effect of computation offloading is affected by many factors, which makes many offloading unable to achieve their expected objectives. In this paper, we present a comprehensive survey of computation offloading in edge computing including offloading scenarios, influence factors and offloading strategies. Particularly, we discuss key issues through the offloading process, such as *whether, where, what* to offload.**

***Index Terms*—Computation Offloading, Edge Computing, Offloading Scenarios, Offloading Strategies.**

I. Introduction

Nowadays, with the advent of the Iot, mangy Iot applications are affecting and changing our lives, such as smart homes, epidemic prevention and control, and telecommunication [1][2]. These applications not only consume considerable energy, memory and computing resources, but also have strict timeliness requirements [3]. Although IoT device is becoming more powerful, for a single device, the battery, cpu and memory are still insufficient when running large application [4]. Computation offloading is considered as one of the solutions to the above problems, which transfer the computation tasks to other system for execution. On the other hand, Cloud computing has grown rapidly and matured in the past decade.It has strong storage and processing capabilities and provides resources and services in a centralized manner. Therefore, cloud servers were considered as the best offloading destination in the early days, Taking Mobile Cloud Computing (MCC)[5] as example, mobile devices could offload computing tasks to the cloud to solve the shortage of resources. However, with the rapid development and popularization of IoT, hundreds and millions of Iot devices will need cloud services. According to CISCO, there will be more than 50 billion Iot devices need to be connected to the Internet by 2020 [6]. This will bring many challenges to the traditional cloud computing, such as transmission delay, bandwidth limitation, data privacy protection and other issues [7][8]. In order to effectively address these challenges, a new computing paradigm is emerging, known as edge computing [9], which utilizes edge servers close to users, such as Cloudlet and MEC, to provide computing and storage services at the edge of the network. Compared with cloud computing mode, edge computing has features such as low latency, high bandwidth and security.

In edge computing, the core of problem lies in how to make offloading decision, which is affected by many factors, such as task characteristics, network conditions and platform differences. For example, The diversity and complexity of the task may diminish the potential gains from offloading. Similarly, the unstable network will hurt the benefits of offloading. Moreover, if an offloading decision does not take variations in these factors, it also will lead to poor performance. Thus, how to make offloading decision according to various factors and their unpredictable variations, so as to achieve the desired objectives is still a research problem.

In this paper, we present a comprehensive survey of computation offloading in edge computing including offloading scenarios, offloading factors and offloading strategies. Particularly, we describe the computation offloading in term of process and discuss its key issues. The contributions of this paper are as follows:

1) We present a comprehensive survey of computation offloading in edge computing including offloading scenarios, offloading factors and offloading strategies, and discuss key issues through the offloading process, such as whether, where, what to offload.

2) We discuss how to partition program and select the offloadable components, and classify offloading strategies.

The rest of this paper is arranged as follows: In Section II,we lists several offloading scenarios where offloading may occur. In Section III, we analyze the factors that affect computation offloading. In Section IV, we discuss how to partition program and classify the offloading strategies. Finally, this paper conclusion in Section V.

II. OFFLOADING SCENARIOS

Computation tasks can be performed in different locations, such as on IoT devices, edge sever or cloud. The offloading destination depends on the trade-off between different objectives and influence factors. In this section, We present some typical offloading scenarios in edge computing. Fig.1 presents an overview of different types of offloading scenarios. The figure is divided into three layers according to the offloading destination, including IoT devices layer, edge computing layer and cloud computing layer. IoT devices layer consists of sensors, mobile devices and nodes, which are numerous and wide distribution. These IoT devices generate large amount of data when running applications. Due to the limited computing and storage capacity of single device, data is usually offloaded to externel servers or devices in the form of tasks. Edge computing layer consists of edge server, intelligent gateway, cloudlet, MEC and fog, which can provide a middle computing capacity, enough storage space and relatively fast response time to meet IoT application requirements. Cloud server can provide high computing capacity and sufficient storage, but it needs to tolerate the long latency and cost of data transmission. In Fig.1, We use different color dotted lines to indicate the direction of offloading. The green dotted lines represent the data is offloaded between IoT devices, such as in the case of a smart phone offloading tasks to nearby mobile devices. The red dotted lines represent the data is offloaded to edge server, which can be from IoT devices or between edge servers. The yellow dotted lines represent the data is offloaded to cloud, which can be from IoT devices, edge servers or between clouds servers. In the following, we will discuss each offloading scenario in detail and present its direction and objectives.

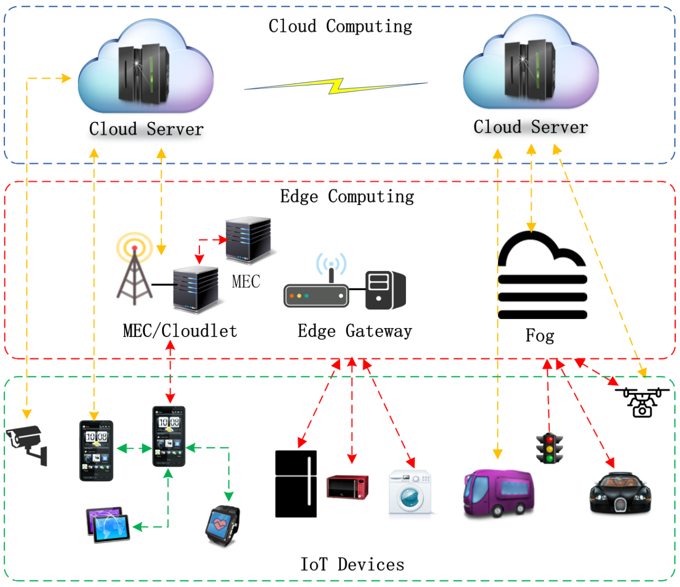

Fig. 1: Offloading Scenarios in Edge Computing

A. *Offloading between IoT devices*

Nowadays, IoT devices have become an essential part of our daily lives. They play not only the role of data producers, but also as data consumers. In edge computing, most of the data generated by IoTs will be processed at the edge of the network instead of transmitting to the cloud [10]. Mobile devices, pads and laptops can provide a certain amount of computing resources, and most of these computing resources around us are idle at most time [11]. When a mobile device encounters a complex task beyond its capability, it can choose to divide the application into smaller tasks and offload to nearby mobile devices with idle computing resources. This can alleviate resource scarcity of single device and improve overall resource utilization in edges.

B. *Offloading from IoT device to edge server*

Due to the limited resouorces of most IoT devices, they cannot handle complex computation tasks. For example, Machine learning (ML) requires heavy computations, especially for training and inference based on deep neural networks (DNN). Single device with limited resources (e.g., smart phone) cannot run the tasks smoothly by itself. In the case of cloud computing, computation tasks must be uploaded to the cloud server for execution. It is inevitable to increase the cost of bandwidth and transmission latency, resulting in inability to meet the real time requirements of IoT, especially in some industrial applications with high real time requirement. In Fig 1, IoT devices can offload computation tasks to nearby edge servers, such as Fog nodes [12], Cloudlets [13] or MEC servers, in which all the tasks will be further processed and analyzed. It effectively reduces the cost and latency. Jeong, et al. [14] employ edge computing to offload DNN computation tasks from IoT devices to nearby generic edge servers, such as MEC or cloudlets. This can effectively reduce the burden of embedded devices to execute ML algorithm, and improve app performance further.

C. *Offloading from edge server to IoT device*

In some applications, IoT devices need to obtain some external information for calculation, such as temperature, humidity and other environmental parameters [15]. This information is difficult for an IoT device to measure and store in large quantities duo to its limited function and resources. In cloud computing paradigms, IoT devices need to request the cloud to provide this information. However, frequent access to the cloud will bring high costs and increase the burden of cloud server. Therefore, the effective solution is to access distributed edge servers, such as Cloudlet or MEC. Edge servers are deployed near IoTs, which has strong computation capacity and can collect information from multiple sources. It can effectively reduce the burden of equipment and the transmission delay as well as the cost of connecting to cloud [16].

D. *Offloading between edge servers*

Iot devices can benefit from edge computing duo to its computational capacity and fast response time. However, the computational capacity of a single edge server is also limited, so it is necessary to combine multiple edge nodes to keep the load balance and share data to provide cooperative services, such as collaborative edge [10]. Take the Coronavirus prevention as an example. The hospital can share the information through collaborative edge, such as infected people, symptoms and beds. At the same time, Centers for Disease Control (CDC) can search and monitor Coronavirus contacts based on the collaborative edge information and timely update the contact information. The community use collaborative Edge to take action to isolate the Coronavirus contacts. CDC can coordinate among hospitals, communities and pharmaceutical companies through collaborative edge, so as to further effectively schedule resources in all aspects and improve prevention and control efficiency.

E. Offloading from IoT device to cloud

In some cases, IoT devices need to offload tasks to remote cloud server for data storage or processing. Earlier research has focused on this area, such as MCC [16], MAUI [17], ThinkAir [18], etc. Take MCC for example, Mobile devices can offload heavy tasks to the powerful distant centralized clouds (e.g., Amazon EC2, Microsoft Azure, and Google) and storage large chunks of data on Cloud storage [19]. MCC brings the benefits of extending battery lifetime, enabling sophisticated applications and providing higher data storage capabilities to the mobile users [20]. In mobile games, mobile devices can completely offload computing intensive tasks to the cloud, such as graphics rendering, and display the results on the screen to interact with users [16]. It can enhance the game experience and save the energy consumption in mobile devices. Chun et al. [21] propose the CloneCloud frameworks which can clone part of application and offload them to cloud. When running to the clone section, application thread can be offloaded from mobile device to its clone in the cloud, and re-offloaded back to the mobile device at the current execution point of program to continue execution. The CloneCloud can reduce the overhead of the application execution. Their experimental results show that the clone cloud can achieve up to 20 times the speed of application execution and reduce the energy consumption on mobile devices by 20 times.

F. Offloading between edge server and cloud

Edge servers (e.g., MEC, Cloudlet and Fog) have computational capacity and storage to fulfill most of the tasks on the edge, but they still need the cloud services to store and access data in many scenarios. For example, the daily patient's log will be offloaded from healthcare system to the cloud for long term observation. Driver behavior monitoring logs also will be uploaded to the cloud for storage and analysis, so as to provide effective basis for vehicle insurance. In another scenario, the edge server and cloud service need to work together to accomplish a task due to the distributed deployment of resources. For example, a fog can obtain data from real-time monitoring by drones during snow disasters or forest fires, and transmit it to the cloud for disaster assessment. When the disaster causes serious damage to the scene, the original roads and houses are no longer recognizable. Fog computing needs the original geographic information provided by the cloud to direct the drones to conduct on-site search and rescue work. In this way, the fog and cloud assign tasks to each other and work together to provide services [1].

G. Offloading from cloud to IoT devices

While it is not common for the cloud to assign tasks directly to Iot devices, this can happen when the cloud requires the underlying device or sensor to perform some particular task and return the result to the cloud. In live video analytics, response time is one of important indicators whether live video analytics can satisfy the Quality of Service [22].Cloud computing is no longer suitable for video analysis because of its long data transmission latency and privacy issues. The cloud will send the task of searching for child to all devices in the target area. After receiving the task, each surveillance camera will search the local data. When the information of missing children is found, the results will be returned to the cloud. Compared with the traditional cloud computing mode, this method adopts edge computing and parallel mode, and has a faster corresponding speed [10]. In addition, some tasks may depend on the underlying equipment, such as image, sound and signal acquisition. These tasks need to be offloaded from cloud to the edge devices [23].

III. INFLUENCE FACTORS

In the offloading process, the crucial step is to determine whether to offload. However, An offloading decision is affected by many factors, some of which are transient and keep on changing during the offloading process. In this section, we will discuss these influence factors .

A. Device Factor

Device factor refers to device characters and states, such as heterogeneity, location and mobility. Firstly, Edge devices are numerous and diverse, including mobiles devices, sensors and IoT devices. They have different hardware architectures, process abilities, storage capacities and operating systems. Thus, the heterogeneity of device will affect offloading effect to some extent, such as execute time and energy consumption. Secondly, the position of device may affect the strength of received signal, for example, if the position of device is close to server or signal transmitter, the received signal by device is strong and easy to offload. On the contrast, when the position is far away from server or signal transmitter, the received signal is weak and difficult to offload. Moreover, the mobility of device can also affect the performance of offloading and cannot be ignored in some cases [24]. For example, In vehicular Edge Computing Networks, a device moves out of current service scope before the offloading result back, it probably needs to find a new edge server to offload task again. This will lead to high latency. Therefore, the device context is an important factor that needs to be considered for computation offloading.

B. Network Factor

For computation offloading, network condition is an important factor that significantly impact on offloading decisions [25]. It includes link quality, bandwidth, network interference etc. Firstly, link quality is critical for data transmission. Edge devices generally communicate with servers through wireless. Different from wired channels, wireless channels have reflection, refraction and multipath fading[26], which make the channels have strong time-varying characteristics and lead to inter-symbol interference [27].When the wireless channel is in deep attenuation, the reduced execution delay of remote execution may not be enough to compensate for the increased transmission delay due to the sharp drop of transmission data rate [15].In such case, offloading task will be delayed until the channel gain is advantageous or switch to another wireless channel with a better quality for offloading. Secondly, the bandwidth between the IoT device and the server determines the data transmit rate, and affect the data transmit time. In some cases, the data transmit time between device and server may dominate the latency of offloading, as a result, it will lead latency too long to satisfy the time constraint [28]. Moreover, network interference is another influence factor, which is usually difficult to predict and affected by device mobility, bandwidth variation, network congestion, and the distance between devices and the servers. Network interference also seriously impact offloading system to satisfy the applications with latency constraint requirement [29].

C. Service Factor

The effect of computation offloading mainly depends on the selection of server nodes, which provide service for processing tasks [30]. These server nodes can be edge servers or IoT devices in the vicinity, or cloud server [31]. Theoretically, an application or task can be offloaded to any server node to execute. But in practice, edge device should consider server nodes' computing capacity, available resources, distance and access technologies before making the offloading decision. The computing capacity represents the speed of processing at server node; it is an important but not the only factor to determine whether to offload. Lack of resources also probably affects the response time. For example, when a task is offloaded to server node, but the CPU is overloaded or occupyed by other threads, the task will be suspended and wait for CPU, so it will increase the response time and affect offloading effects. Moreover, the location and access technologies of server node will also affect offloading latency and energy consumption. As we describe in offloading scenarios in Section II. Computation offloading can take place at different scenarios, such as offloading between devices, or from device to edge sever, or from device to cloud. The distances between device and server are different in these offloading scenarios. This will result in different transmission latency. For example, edge server is closer to the user devices and has less transmission latency, but its computing capacity is relatively lower than remote cloud center, and need to spend more time on executing latency. On the contrast, the cloud center has high computing capacity and less executing latency, but it must suffer from Round-trip delay and pay the cost in using network resources. Different access technologies also affect energy consumption of offloading. Therefore, the selection of server node needs to be considered for offloading system to provide better application performance.

D. User Factor

User's preference is also an important factor in determining whether or not to offload [32]. However, user's preference is sometimes elusive, which is influenced by age, gender, experience, etc. To illustrate the effect of user preferences on computation offloading, we list the four reference factors that users are most concerned about: security, Monetary cost, latency and energy consumption. Firstly, security is critical to users, which refers to protect their private data and integrity of computational data during the offloading process. Moreover, offloading destination must be a reliable place for task execution. Secondly, monetary cost is also an important factor for users to consider whether or not to offload, which mainly consists of Network fee and Service fee. Network fee refers to the cost of network communication and traffic, which is charged by the user. Service fee refers to the cost of renting service on the server, which usually depends on the number and time of virtual server usage. Moreover, latency and energy consumption are the concerns of users. The latency is the total time taken by the offloading system to complete the execution of the task, which consists of uploading time, execution time and downloading time. Users usually compare offloading latency with the local execution latency, and select whether to perform task migration. As the same, the energy consumption also needs to compare offloading energy consumption with local execution. We will discuss them in detail in the following section.

IV. OFFLOADING SELECTION

After analysising the influence factors, the next step is determining *what* to offload. However, Offloading an entire application may not always be beneficial [33]. The device may spend more energy or time in the offloading process than it does locally [34]. This is duo to select inappropriate application to offload and the costs for transferring may not be negligible [35]. To reduce the execution time and transfer cost, the application should be partitioned to transfer and parallel execution. Yang et al.[36] proved that the program partitioning and parallel processing have a positive impact on the transmission time and offloading efficiency, and further reduce the execution time and energy consumption of the application. Therefore, how to partition the program or application and select *what* to offload is the important step in offloading process.

A. Program partition and selection

A program or application can be partitioned through a diversity of strategies. We classify these strategies as two types. The first type is static partitioning schemes, in which the offloadable components are predefined regardless of the execution environment. For example, software engineers add special static annotations on methods (such as *@offloadable or @Remote*) [31], which denote those methods should be offloaded. However, applications are always composed of some non-offloadable component(s) which cannot be offloaded and need to be executed locally, such as user input, face detection and positioning, even these components are mutually dependent and invocations. we use a Component Dependency Graphs (CDGs) to describe their relationship. The vertices of the CDG represent different components or tasks of the

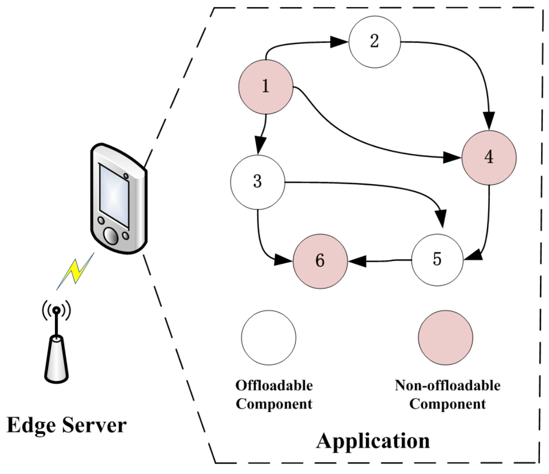

Fig. 2: Component Dependency Graphs (CDGs) of an Application

application and the edges of the CDG represent the invocations among these components. An example of a CDG representation is shown in Fig.2. The red vertices represent the non-offloadable components (such as 1st, 4th, 6th component in Fig.2) which must be executed locally. The white vertices represent the offloadable components (such as 2nd, 3rd, 5th component in Fig.2) which can be offloaded. Moreover, the edges represent the invocations among these components.

B. *Offloading Strategies*

From above analysis, we know that the program partitioning and parallel processing can improve the offloading performance. However, in some cases, Some programs are highly integrated or relatively simple, and can not be divided into multiple tasks for parallel execution. They must be executed on the local device or offloaded to the server as a whole. Therefore, the offloading strategies can be classified as full offloading and partial offloading:

1) Full offloading: The program cannot be partitioned and offloaded as whole to the server. Full offloading strategies can be divided into support single user and multi-users.

Supporting for single user,You,et al.[37] proposed a noval framework of wirelessly powered MCC to optimize the performance of both local computing and offloading under the energy harvesting and deadline constraint. The result show that they gain substantial performance and close to optimal performace in both static and dynamic channel. Consider the impact of offloading on program execution time, Lin, et al. [38] proposed Ternary Decision Maker (TDM) offloading framework, which response to measure impact factors of offloading and make offloading decision. The TDM offload tasks in term of module in order to reduce the response time and power consumption. Experimental results show that TDM is superior to other existing methods in offloading decision-making and achieves a reduction in execution time of 75% and battery consumption of 56%.

Supporting for the multi-users, Zhao,et al.[39] proposed a *reformulation-linearization-technique-based Branch-and-Bound*(RLTBB) method, which jointly optimized offloading selection, radio resource and computational resource allocation, in order to minimize the energy consumption on multi mobile devices. Simulation results demonstrated that RLTBB could achieve 95.4% of energy saving on average. Xu, El et al.[40] studied the multi-user computing offloading problem for MCC in multi-channel wireless interference environment. They proposed an offloading method based on game theory to achieve efficient computation offloading in a distributed manner. Numerical results show that their method has better offloading performance and scale with the increase of user size.

2) Partial offloading: The program can be partitioned into tasks and offload these respective tasks to server. Partial offloading strategies also can be divided into supporting single user and multiple users.

Supporting for single user, Gu et al.[41] propose an adaptive offloading system (DOS), which is consist of distributed offloading platform and offloading inference engine. The distributed offloading platform is responsible for monitoring application execution, application-partitioning and resource allocation. The offloading inference engine is responsible for triggering offloading and selecting partitioning. Their evaluations showed that DOS can effectively improve the resource utilization ratio of mobile devices. Compared with other existing methods, their method has the advantage of lower resource consumption. Mahmoodi, et al.[42] proposed the joint scheduling and computation offloading (JSCO) based on mobile applications, which introduce wireless parameter and uses CDG to optimize the task scheduling and offloading decisions. This method makes full use of the advantages of mobile and cloud parallel processing to reduce the execution time. Simulation results show that the JSCO can reduce power consumption by 54% and increase execution speed by up to 25% compared to local execution. Zhang, et al.[43] formulated a generic energy-saving offloading scheduling problem for real-time video applications and proposed an adaptive scheduling method based on the dynamic wireless network conditions. They verified the effectiveness of the method through tracking drive simulation experiments.

Supporting for the multi-users, Huang, et al.[25] presented a dynamic offloading method based on Lyapunov optimization to realize energy saving under the requirement of ensuring the application execution time. The experimental results show that this method can meet the requirements of application execution time and save more energy than other existing methods. Liu, et al.[44] studied a power-constrained delay minimization problem, and proposed an one-dimensional search algorithm to find the optimal offloading policy. There is significant reduction in the execution latency compared to the previous works.

V. Conclusion

Computation offloading is a critical technology for edge computing, which can prolong battery life, reduce the latency and improve application performance. In practice, the effect of computation offloading is affected by many factors, which makes many offloading unable to achieve their expected objectives. In this paper, we presented a comprehensive survey of computation offloading in edge computing. we describe the computation offloading in term of process and discuss its key issues, such as where, whether and what to offload. Around these issues, we respectively elaborate on the application scenarios, influence factors and strategies of offloading.

REFERENCES

[1] Aazam, Mohammad , S. Zeadally , and K. A. Harras . "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities." Future Generation Computer Systems (2018): S0167739X18301973.

[2] A. Essameldin, K.A. Harras, The hive: An edge-based middleware solution for resource sharing in the internet of things, in: Proceedings of the 3rd Workshop on Experiences in the Design and Implementation of Smart Objects, ACM, 2017.

[3] Yu, Shuai , X. Wang , and R. Langar . "Computation offloading for mobile edge computing: A deep learning approach." 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC) IEEE, 2018.

[4] Vladimir Marbukh. "Towards efficient offloading in fog/edge computing by approximating effect of externalities." IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) IEEE, 2018.

[5] Sanaei, Zohreh , et al. "Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges." IEEE Communications Surveys & Tutorials 16.1(2014):369-392.

[6] Evans, D. The Internet of Things how the next evolution of the Internet is changing everything [OL].

[7] Satyanarayanan, Mahadev. "A Brief History of Cloud Offload." Acm Sigmobile Mobile Computing & Communications Review 18.4(2015):19-23.

[8] Zhang, Qi, L. Cheng, and R. Boutaba. "Cloud computing: state-of-the-art and research challenges." Journal of Internet Services & Applications1.1(2010):7-18.

[9] Satyanarayanan, Mahadev. "Edge Computing." Computer 50.10(2017):36-38.

[10] Shi, Weisong, et al. "Edge Computing: Vision and Challenges." IEEE Internet of Things Journal 3.5 (2016):637-646.

[11] Lee, Youngki, et al. "CoMon:cooperative ambience monitoring platform with continuity and benefit awareness." International Conference on Mobile Systems, Applications, and Services ACM, 2012:43-56.

[12] Bonomi, Flavio, et al. "Fog computing and its role in the internet of things." Edition of the Mcc Workshop on Mobile Cloud Computing ACM, 2012:13-16.

[13] Satyanarayanan, Mahadev, et al. "The Case for VM-Based Cloudlets in Mobile Computing." IEEE Pervasive Computing 8.4(2009):14-23.

[14] Jeong, Hyuk Jin, et al. "Computation Offloading for Machine Learning Web Apps in the Edge Server Environment." IEEE, International Conference on Distributed Computing Systems IEEE Computer Society, 2018:1492-1499.

[15] Mao, Yuyi, et al. "A Survey on Mobile Edge Computing: The Communication Perspective." IEEE Communications Surveys & TutorialsPP.99(2017):1-1.

[16] Dinh, Hoang T., et al. "A survey of mobile cloud computing: architecture, applications, and approaches." Wireless Communications & Mobile Computing 13.18(2013):1587-1611.

[17] Cuervo, Eduardo, et al. "MAUI:making smartphones last longer with code offload." International Conference on Mobile Systems, Applications, and Services DBLP, 2010:49-62.

[18] Kosta, S, et al. "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading." IEEE INFOCOM IEEE, 2012:945-953.

[19] Zhou, Zhibin, and D. Huang. "Efficient and secure data storage operations for mobile cloud computing." International Conference on Network and Service Management International Federation for Information Processing, 2012:37-45.

[20] Rahimi, M. Reza, et al. "Mobile Cloud Computing: A Survey, State of Art and Future Directions." Mobile Networks & Applications 19.2(2014):133-143.

[21] Chun, Byung Gon, et al. "CloneCloud: elastic execution between mobile device and cloud." Conference on Computer Systems ACM, 2011:301-314.

[22] Ananthanarayanan, Ganesh, et al. "Real-Time Video Analytics: The Killer App for Edge Computing." Computer 50.10(2017):58-67.

[23] Bhattacharya, Arani, and P. De. "A Survey of Adaptation Techniques in Computation Offloading." Journal of Network & Computer Applications78(2016).

[24] Yang, Chao, et al. "Efficient Mobility-Aware Task Offloading for Vehicular Edge Computing Networks." IEEE Access (2019):1-1.

[25] Huang, Dong, P. Wang, and D. Niyato. "A Dynamic Offloading Algorithm for Mobile Computing." IEEE Transactions on Wireless Communications11.6 (2012):1991-1995.

[26] Sourour, E. A, and M. Nakagawa. "Performance of orthogonal multicarrier CDMA in a multipath fading channel." IEEE Transactions on Communications 44.3(1994):356-367.

[27] Li, Chang , et al. "User-Centric Intercell Interference Nulling for Downlink Small Cell Networks." IEEE Transactions on Communications 63.4(2015):1419-1431.

[28] Khan, Minhaj Ahmad . A survey of computation offloading strategies for performance improvement of applications running on mobile devices.. Academic Press Ltd. 2015.

[29] Namboodiri, Vinod, and T. Ghose. "To cloud or not to cloud: A mobile device perspective on energy consumption of applications." World of Wireless, Mobile and Multimedia Networks IEEE, 2012:1-9.

[30] Zhao, Pengtao, et al. "Energy-Saving Offloading by Jointly Allocating Radio and Computational Resources for Mobile Edge Computing." IEEE Access PP.99 (2017):1-1.

[31] Wang, Wei, et al. "Edge Caching at Base Stations with Device-to-Device Offloading." IEEE Access (2017):1-1.

[32] Alam, Muhammad Mahtab , D. B. Arbia , and E. B. Hamida . "Research Trends in Multi-Standard Device-to-Device Communication in Wearable Wireless Networks." International Conference on Cognitive Radio Oriented Wireless NetworksSpringer, Cham, 2015.

[33] Barbera, Marco V. , et al. "Mobile offloading in the wild: Findings and lessons learned through a real-life experiment with a new cloud-aware system." Infocom, IEEE IEEE, 2014.

[34] Flores, Huber, and S. Srirama. "Mobile code offloading: should it be a local decision or global inference?." Proceeding of the 11th annual international conference on Mobile systems, applications, and services ACM, 2013:539-540.

[35] Kwon, Yongin, et al. "Precise execution offloading for applications with dynamic behavior in mobile cloud computing." Pervasive & Mobile Computing 27.C(2016):58-74.

[36] Yang, Seungjun, et al. "Fast dynamic execution offloading for efficient mobile cloud computing." IEEE International Conference on Pervasive Computing and Communications IEEE, 2013:20-28.

[37] You, Changsheng , K. Huang , and H. Chae. "Energy Efficient Mobile Cloud Computing Powered by Wireless Energy Transfer." IEEE Journal on Selected Areas in Communications(2016):1-1.

[38] Lin, Ying Dar, et al. "Time-and-Energy-Aware Computation Offloading in Handheld Devices to Coprocessors and Clouds." IEEE Systems Journal9.2 (2015):393-405.

[39] Zhao, Pengtao, et al. "Energy-Saving Offloading by Jointly Allocating Radio and Computational Resources for Mobile Edge Computing." IEEE Access PP.99 (2017):1-1.

[40] Chen, Xu, et al. "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing." IEEE/ACM Transactions on Networking 24.5(2015):2795-2808.

[41] Gu, Xiaohui , et al. "Adaptive offloading for pervasive computing." Pervasive Computing IEEE 3.3(2015): 66-73.

[42] Mahmoodi, S. Eman, R. N. Uma, and K. P. Subbalakshmi. "Optimal Joint Scheduling and Cloud Offloading for Mobile Applications." IEEE Transactions on Cloud Computing PP.99(2016):1-1.

[43] Zhang, Lei, et al. "On Energy-Efficient Offloading in Mobile Cloud for RealTime Video Applications." IEEE Transactions on Circuits and Systems for Video Technology (2016):1-1.

[44] Liu, Juan , et al. "Delay-Optimal Computation Task Scheduling for Mobile-Edge Computing Systems." (2016).