Wireless Networks (2022) 28:1563–1585
https://doi.org/10.1007/s11276-022-02920-2

ORIGINAL PAPER

A survey of research on computation offloading in mobile cloud computing

Xiaomin Jin[1,2,3] · Wenqiang Hua[1,2,3] · Zhongmin Wang[1,2,3] · Yanping Chen[1,2,3]

Accepted: 7 February 2022 / Published online: 6 March 2022

Abstract

Mobile devices (MDs), represented by smartphones, have been widely used in various industries. However, MDs are constrained by their limited resources and cannot execute computation-intensive applications. Mobile cloud computing (MCC), which provides MDs with a rich pool of resources that can be accessed through wireless networks, is proposed to extend MDs' capacity by computation offloading. MCC helps MDs breakthrough their resource constraints, frees them from heavy local workloads, and allows them to take more responsibility for connecting mobile users (MUs) and the information domain. MCC computation offloading has attracted wide attention because of its tremendous potential, and a lot of related research has been done. In this paper, we provide a survey of the research on computation offloading in MCC so that readers can spend less time to have a comprehensive understanding of this field, and know its key technologies and future directions. We first summarize the MCC architecture and offloading granularity, which are the most fundamental concepts of MCC. The computation offloading system is decomposed into three basic components: MUs, application service operators, and cloud operators. We then conducted a comprehensive literature review on offloading decision, admission control, resource management, and edge equipment deployment, which are four key technologies for these components. Wireless network connection and heterogeneity are the basic features of MCC, which increase the possibility of failure and privacy leakage during the computation offloading process. We also review the auxiliary technologies for computation offloading in terms of fault tolerance and privacy protection. Finally, we present the research outlook of the systematic prototype and other technologies from the perspective of "device-pipe-cloud".

Keywords Mobile cloud computing · Computation offloading · Offloading decision · Resource management · Edge equipment deployment · Fault tolerance · Privacy protection

✉ Xiaomin Jin
xmjin@xupt.edu.cn

Wenqiang Hua
huawenqiang@xupt.edu.cn

Zhongmin Wang
zmwang@xupt.edu.cn

Yanping Chen
chenyp@xupt.edu.cn

[1] School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xi'an 710121, Shaanxi, China

[2] Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing, Xi'an 710121, Shaanxi, China

[3] Xi'an Key Laboratory of Big Data and Intelligent Computing, Xi'an 710121, Shaanxi, China

1 Introduction

With the rapid development of wireless network and computer technologies, using mobile devices (MDs) has become highly popular in many industries. Cisco predicted that the number of MDs worldwide will grow from 8.8 billion in 2018 to 13.1 billion in 2023 [1]. At the same time, due to the progress of the Internet of Things (IoT), a large number of devices are connected to mobile Internet, making the MD concept further expand. With the popularity of MDs, mobile Internet has entered a high-speed development stage. For instance, according to Internet Trend Report 2019, the number of mobile users (MUs) in China has exceeded 817 million with a year-on-year growth rate of 9%, and their mobile data traffic consumption increased by 189% [2]. Benefiting from the continuous improvement of chip manufacturing techniques, MDs are equipped with more powerful CPUs and larger memories, enabling MDs to handle more business for MUs. However, the faster CPU has the greater energy consumption since CPU power increases super-linearly with its frequency [3]. MDs are usually powered by batteries, and the battery volume and capacity are limited to support MDs' portability. As one of the most intuitive feelings, compared with feature phones of the previous generation, although today's smartphones have more functions, they have shorter working hours after one charge. Different from the semiconductor technology following by Moore's Law, the battery technology has not made breakthroughs in the short term, and the annual growth rate of the battery capacity is only 5% [4]. Furthermore, due to a series of factors such as CPU architecture and heat dissipation, regardless of the fact that MD processing capacity has been improved, it is still weak to execute some computation-sensitive applications. MDs' limited resources cannot satisfy the increasingly complex MU requirements.

Computation offloading migrates computing tasks to the external platform to extend available MD resources, which is an effective way to solve the problem of limited MD resources [5]. Cloud computing, as the foundation of future information industry, is a business computing model, which can provide rich resources to MDs. Cloud computing is a pay-per-use model that supplies available, convenient, on-demand network access to a shared pool of configurable computing and storage resources [6]. The concept of cloud computing was first proposed by Google CEO Schmidt in 2006, and remarkable technological progresses have been achieved after more than 10 years of development. At present, there are many mature commercial cloud computing services, such as Amazon Web Services, Microsoft Azure, Alibaba Cloud, Tencent Cloud, etc. Based on cloud computing and computation offloading, mobile cloud computing (MCC), which provides MDs with a rich pool of resources that can be accessed through wireless networks, is proposed to address the problem that MDs' resources are limited. MCC is an association of cloud computing, mobile computing, and wireless network and can migrate offloading units (OUs) to the cloud via computation offloading [7–9]. MCC helps MDs breakthrough their resource constraints, frees them from heavy local workloads, allows them to take more responsibility for connecting MUs and the information domain, and makes them become a simple modem that connects humans to the electromagnetic signal-based network. Benefiting from MDs' portability, MUs can connect with the resource-rich cloud anytime and anywhere in MCC, and enjoy the convenience of informatization better. MCC has attracted wide attention from industry and academia because of its tremendous potential. According to the assessment of Allied Analytics LLP, the mobile cloud market is valued at $12.07 billion in 2016 and is expected to reach $72.55 billion by 2023, with a compound annual growth rate of 30.1% from 2017 to 2023 [10]. It can be inferred that the mobile cloud market will become more prosperous with the progress of MCC.

Computation offloading is the core of MCC and determines the ultimate MCC effect. Researchers have studied computation offloading from different perspectives and provided numerous research results. Wu presented a survey of current research work on multi-objective decision making for time-aware and energy-aware computation offloading in MCC [11]. Mach and Becvar surveyed the work on computation offloading from the perspective of offloading decision, computing resource allocation, and mobility management [12]. Bhattacharya and De focused on the variability and unpredictability of MCC environment in offloading decision [13]. They categorized the parameters that influence computation offloading as applications characteristics, network properties, and execution platform features, and then surveyed adaptation techniques utilized for computation offloading. Kumar et al. investigated different types of offloading decision algorithms and classified the application types that have been used to demonstrate [14]. Khan performed a survey of the computation offloading strategies impacting the performance of offloaded applications and categorized offloading decision approaches into static and dynamic [15]. Chen and Cheng reviewed the offloading decision algorithms and classified them into three categories based on three decision scenarios (i.e., single user, multiple users, and enhanced server) [16]. Shakarami et al. surveyed the offloading decision approaches from the perspective of game theory and classified these approaches into four main fields based on the game mechanisms (i.e., classical game

mechanisms, auction theory, evolutionary game mechanisms, and hybrid-base game mechanisms) they used [17].

Nevertheless, these surveys ignored the systematicness of computation offloading in MCC and mainly focused on offloading decision approaches. Instead of reviewing one or more discrete technologies, this paper considers the interaction among components of the MCC computation offloading system and summarizes their key technologies. Our work aims at providing a comprehensive survey of research on computation offloading in MCC so that readers can spend less time to have a more comprehensive understanding of this field, and know its key technologies and existing problems. Architecture is the foundation of MCC, and different architectures lead to different computation offloading patterns. The offloading granularity determines the OU in the computation offloading system. Therefore, we first summarize the MCC architecture and offloading granularity. To promote MCC and computation offloading on a large scale, all the transactions should not be coupled together but decomposed into separate components that are responsible for their transactions under principles of high cohesion and low coupling. The computation offloading system has three basic components, i.e., MUs, application service operators (ASOs), and cloud operators (COs) [18]. Offloading decision, admission control, resource management, and equipment deployment are their technical challenges. We then summarize the key technologies used to solve these four challenges. In the theoretical research, it is usually assumed that there will be no failures in the computation offloading process, and the MU data will not be stolen. However, in the real-life MCC computation offloading system, it is prone to fail in the process of data transmission or/and distributed OU execution since MDs are connected to the cloud through wireless networks, and MDs are heterogeneous with cloud data centers. MDs usually store MUs' high-privacy personal information. Privacy leakages and malicious attacks may occur in the process of data transmission or/and OU execution. Fault tolerance and privacy protection, which are two critical auxiliary technologies for computation offloading, are also summarized. Finally, we present a research outlook for MCC computation offloading from the perspective of systematic prototype and "device-pipe-cloud". Our contributions can be summarized as follows:

(1) We consider the interaction among components of the MCC computation offloading system and conduct a comprehensive literature review on four technologies involved in computation offloading from the perspective of system by giving an explicit comparison and feature analysis of them. Besides, we summarize the MCC architecture and offloading granularity, which are the fundamental concepts of MCC.

(2) In MCC, MDs are connected to the cloud through wireless networks, and MDs are heterogeneous with cloud data centers. These two characteristics make failures and privacy leakages inevitable in the real-life process of data transmission or/and distributed OU execution. We summarize fault tolerance and privacy protection for computation offloading, and analyze their internal technical theories.

(3) We discuss the future research trend in systematic MCC prototype and analyze technologies that need to be paid attention to in the future from the perspective of "device-pipe-cloud".

The remainder of this paper is organized as follows. Section 2 summarizes the MCC architecture and offloading granularity. Section 3 summarizes four technologies of offloading decision, admission control, resource management, and equipment deployment. Section 4 summarizes fault tolerance and privacy protection for computation offloading. Section 5 discusses the future research directions. Section 6 concludes this paper. Table 1 shows the abbreviations used throughout this paper.

2 MCC architecture and offloading granularity

2.1 MCC architecture

After the concept of MCC was proposed, researchers have given many definitions [8, 18, 19]. These definitions are different due to various research contents and scenarios, but their core ideas are the same, that is, MCC offloads OUs from the MD to external platforms for execution, thereby enhancing the MD capability. According to the available external platforms, MCC architectures can be classified into four categories: one-layer architecture, two-layer architecture, three-layer architecture, and hybrid architecture [20], which are shown in Fig. 1. These MCC architectures are compared in Table 2.

(1) One-layer Architecture

In the one-layer architecture, several neighboring MDs form a network, in which these MDs make up the external platform to provide their idle resources to each other. The formed network is a self-organizing dynamic network that allows MDs to join and to leave at any time. This architecture, which uses MDs as the cloud servers to complete computing tasks through cooperation among MDs, is a variant of traditional cloud computing. A typical example is Hyrax [21], which applied Hadoop in MCC. Hyrax uses a group of smartphones to execute computing tasks in parallel and implements large-scale distributed applications through the cooperation among these smartphones. The one-layer architecture has advantages of short distance and

Table 1 The abbreviation list

Abbreviation	Definition	Abbreviation	Definition
MD	Mobile Device	B2B	Business to Business
IoT	Internet of Things	B2C	Business to Consumer
MU	Mobile User	QoS	Quality of Service
MDP	Markov Decision Process	MCOP	Min-Cost Offloading Partitioning
MCC	Mobile Cloud Computing	GCS	Global Cloud Server
OU	Offloading Unit	LISPS	Local Internet Service Provider Server
ASO	Application Service Operator	GS	Gateway Server
CO	Cloud Operator	TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
AP	Access Point	ITU	International Telecommunications Union
ETSI	European Telecommunications Standards Institute	WLAN	Wireless Local Area Network
MEC	Multi-access/Mobile Edge Computing	CMT	Concurrent Multiple Transfer
3GPP	3rd Generation Partnership Project	SCTP	Stream Control Transmission Protocol
VM	Virtual Machine	MPTCP	Multi-path Transmission Control Protocol
OS	Operating System	ILP	Integer Linear Programming

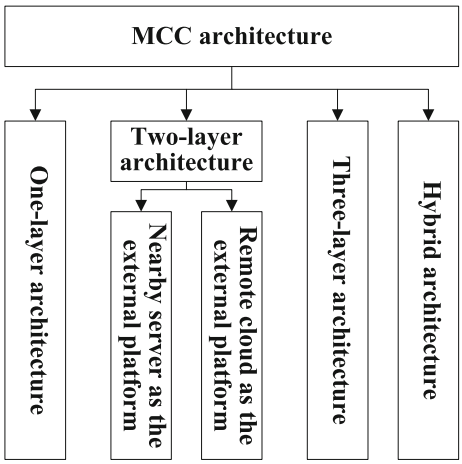

Fig. 1 Categories of the MCC architectures

fast data transmission, which solves the problem of MD resource limitation to a certain extent. However, the external platform in this architecture is composed of MDs, which have few resources and are difficult to provide sufficient resources. Moreover, MDs are owned by different MUs so that problems of permission and privacy are prone to occur in real life. How to persuade MUs to contribute their precious MD resources will be a major challenge.

(2) Two-layer Architecture

In the two-layer architecture, MDs are in the first layer, and the external platform is in the second layer. The external platform can be the server nearby MDs or the remote cloud. The two-layer architecture can be classified into two subclasses according to the external platforms.

A nearby server named cloudlet, which refers to the resource pool composed of small-scale data center clusters at the Internet edge and aims to bring cloud services around MUs, was proposed by Satyanarayanan et al. in 2009 [27]. In [27], Satyanarayanan et al. implemented a cloudlet prototype based on the dynamic virtual machine (VM) synthesis. In this prototype, when the MD wants to use the cloudlet, it first generates the parameters required to override the VM and then sends them to the cloudlet. After receiving these parameters, the cloudlet combines these parameters with its basic VM to generate the same VM as the MD, then executes the MD applications using its VM, and finally returns results to the MD. After the concept of cloudlet was proposed, some researchers are committed to improving it. For example, Hua et al. proposed a scheduling mechanism based on statistical prediction to solve the problem of a long time consuming for cloudlet synthesizing VM [28]. According to the prediction information, the service application VM on cloudlet is pre-synthesized to reduce the MU's waiting time. Cloudlets can be composed of PCs, workstations, or small servers, which are deployed around wireless network access points (APs), such as Wi-Fi hotspots in libraries or coffee shops, to provide cloud service for MDs connected to APs. Also, in the disaster relief or war scenario, temporary cloudlets (e.g., unmanned aerial vehicles) can be deployed [31–33]. Compared with the one-layer architecture, using nearby servers can alleviate the shortage of external platform resources while retaining the advantage of low delay.

Table 2 Comparison of the MCC architectures

Work	Architecture	Characteristics
[21–26]	One-layer architecture	1. Low delay 2. Low-level resources 3. Self-organizing dynamic network 4. Low construction cost
[27–33]	Two-layer architecture (nearby server)	1. Low delay 2. Medium-level resources 3. Deploy nearby server and has a high construction cost
[34–39]	Two-layer architecture (remote cloud)	1. High delay 2. High-level resources 3. Use the existing public cloud and has a low construction cost
[40–46]	Three-layer architecture	1. Low delay 2. High-level resources 3. Use the existing public cloud partly and has a high construction cost
[47–51]	Hybrid architecture	1. High flexibility 2. Construct for specific application scenarios

The architecture, in which the remote cloud is the external platform, is the classical MCC architecture. The remote cloud is rich in resources and has many commercial products, which is convenient for the practical construction of MCC. In traditional cloud computing, the user device (e.g., desktop computer) is connected to the cloud via wired networks and is electrified by plugs and sockets, which suppress the need for energy-efficient data transmission techniques. In this MCC architecture, MDs connect to the remote cloud via wireless networks, which consume MD energy and have limited bandwidth. If the energy and time saved by computation offloading are less than those consumed by data transmission through wireless networks, computation offloading is invalid. Wireless networks have a serious impact on the energy- and time-saving effects of computation offloading in MCC. The question of whether it is worth the effort to offload the computation to the remote cloud must be answered first. Kumar and Lu answered this question through theoretical analysis and experiment in [34]. They found that MCC can potentially save energy for MDs but not all computations are energy-efficient when offloaded to the remote cloud. They take energy-saving as the optimization goal and illustrated that offloading is beneficial when large amounts of computation are needed with relatively small amounts of communication. This MCC architecture completely solves the problem of limited resources and can be considered as providing unlimited resources. However, it is troubled by the problems of high delay and consumption caused by wireless networks.

(3) Three-layer Architecture

The two-layer architecture, which uses the nearby server as the external platform, can alleviate the resource shortage of one-layer architecture while retaining the advantage of low delay. The classical MCC architecture, which is the other two-layer architecture and uses the remote cloud as the external platform, can provide sufficient resources. However, the classical MCC architecture is not suitable for delay-sensitive scenarios (e.g., real-time control, real-time data processing, augmented/virtual reality, etc.) because of its high delay. Through the above analysis, it can be seen that the two-layer architecture that uses the nearby server as the external platform has limited resources but low delay, while the classical MCC architecture has rich resources but high delay. If they are combined, their advantages can be used to overcome each other's shortcomings, and the resources and delay can be balanced perfectly.

Compared with the remote cloud, the nearby server is closer to MDs, and the layer it locates is also called "edge layer". Adding the edge layer to classical two-layer MCC architecture forms a new architecture named multi-access/mobile edge computing (MEC). It is worth noting that MEC refers to mobile edge computing in the previous concept. In 2017, the European Telecommunications Standards Institute (ETSI) extended the MEC concept from initially supporting only the 3GPP mobile network to also supporting the non-3GPP networks (including multiple types of wireless networks and even wired networks), and its name is also modified from mobile edge computing to multi-access edge computing [52]. MEC is an enhancement and extension of MCC and still belongs to the MCC

category. Because this architecture takes advantage of the collaboration between edge cloud and remote cloud, it is also called "cloud-edge collaboration" architecture. Compared with classical MCC, MEC has more advantages. On the one hand, MEC can transfer delay-sensitive OUs to the edge layer to reduce the response time. For example, in industrial manufacturing, some information requires real-time analysis to deal with emergencies timely, and this work can be done at the edge layer. Big data generated in the manufacturing process can be analyzed by machine learning technologies in the remote cloud. Wu et al. introduced a three-layer architecture for data-driven machine health and process monitoring in cyber-manufacturing [42]. The training datasets are streamed into the remote public cloud, in which the diagnostic and prognostic models are built using parallel machine learning algorithms. The predictive models are downloaded to the local private edge cloud and applied to the real-time datasets streamed to the edge cloud for online diagnosis and prognosis. On the other hand, MD data is transmitted to the edge layer through local/private network for processing or preprocessing instead of directly being transmitted to the remote cloud through the public core network, which greatly relieves the pressure of core network and enhances the data security. Just as every coin has two sides, classical MCC also has its advantages. Classical MCC uses the mature commercial cloud computing platform, which helps it to provide low-cost mobile cloud services quickly. Many traditional cloud services can be transplanted to MDs with minor modifications. MEC needs to deploy a large number of edge equipment additionally, which will bring great economic pressure, hinder the development speed and increase the service cost. On the contrary, classical MCC has no pressure to deploy the edge equipment.

(4) Hybrid Architecture

The hybrid architecture combines several different architectures to build MCC suitable for specific application scenarios. Sanaei et al. proposed a hybrid MCC architecture integrating the above three architectures, analyzed the key problems and solutions to achieve it, and demonstrated the application of hybrid MCC through a medical treatment case [47]. Alonso-Monsalve et al. proposed a hybrid MCC architecture, which combines the classical MCC architecture with the utilization of volunteer platforms as resource providers [48]. Their proposed architecture is an inexpensive solution, which highlights benefits in cost savings, elasticity, scalability, load balancing, and efficiency. Zhou et al. proposed a hybrid MCC architecture integrating the above three architectures, in which MDs, cloudlets, and remote cloud form a shared resource network for computation offloading [49]. To incentivize MUs to provide their MD resources, they designed an auction-based computation offloading market, in which MUs can sell and buy MD resources. Feng et al. proposed a hybrid MCC architecture to improve the computational capability of vehicles by using resources from the remote cloud, roadside units, and neighboring vehicles [50]. Their architecture also integrates the above three architectures, in which the roadside units are nearby servers, and the vehicles can offload their OUs to each other. Flores et al. proposed a social-aware hybrid computation offloading system, which integrates cloudlet, remote cloud, and device-to-device networks, and increases the spectrum of offloading opportunities [51]. They designed an credit- and reputation-based incentive mechanism to foster MUs to lease their MD resources as open commodities that may be acquired by others.

2.2 Offloading granularity

Offloading granularity determines the OU in the computation offloading process. At present, there has been some research on the MCC computation offloading prototype, which mainly focuses on the design and implementation of the software system. Researchers used different offloading granularity in their prototype, such as method, class, thread, VM, web service, etc. The offloading granularities in these prototypes are compared in Table 3. Besides these prototypes, more research studied MCC from the theoretical perspective, and there are two commonly used task/application models, which also indicate their offloading granularities. Fig. 2 shows categories of the offloading granularity.

(1) *The independent task model* [26, 30–34, 36, 40, 43, 45, 46, 49, 50, 57–63]. This model abstracts the task into an independent computing module, which has no interaction with other computing modules and arrives following a stochastic process (e.g., the Poisson process). The computing module is usually defined as a 3-tuple, in which one item represents the size of input data, one item represents the necessary CPU cycles to accomplish the computing module, and one item represents the maximum tolerable delay. In this model, the computing module is the OU and represents the offloading granularity.

(2) *The graph model* [35, 64–72]. A real-life mobile application is composed of many components (e.g., classes, threads, or methods), which are the OUs. A component can call other components for execution and needs the output data from other components. Therefore, a mobile application can be abstracted as a graph, in which vertexes represent the components, and the edge represents the interactive relationship between two components. In this model, the vertex represents the offloading granularity. An example of the graph-based application model is illustrated in [72]. The vertex is modeled as a 3-tuple, in which one item represents the CPU cycles, one item represents the indicator of whether it is offloadable, and one item

Table 3 Comparison of the offloading granularity in MCC computation offloading prototypes

Work	Architecture	Granularity	MD OS	Target
Hyrax [21]	One-layer	Class	Android	Port Hadoop to run on MDs
MDCloud [22]	One-layer	Method	Android	Minimize the energy consumption and execution time
Serendipity [23]	One-layer	Class	Android	Minimize the energy consumption or execution time
Honeybee [24]	One-layer	Method	Android	Minimize the energy consumption
Cloudlet [27]	Two-layer	VM	Maemo 4.0 Linux	Minimize the execution time
Cloudlet [28]	Two-layer	VM	Android	Minimize the execution time
Cloudlet [29]	Two-layer	VM	Not mentioned	Minimize the execution time and improve the accuracy
Noname [37]	Two-layer	Method	Android	Minimize the execution time
Noname [38]	Two-layer	Web service	Not mentioned	Minimize the energy consumption and execution time
CloneCloud [53]	Two-layer	Thread	Android	Minimize the energy consumption and execution time
MAUI [54]	Two-layer	Method	Windows Mobile 6.5	Minimize the energy consumption
ThinkAir [55]	Two-layer	Method	Android	Minimize the energy consumption
Cuckoo [56]	Two-layer	Method	Android	Minimize the energy consumption or execution time
EAB [41]	Three-layer	Web service	Android	Minimize the execution time
NoName [42]	Three-layer	Task module	Not mentioned	Apply MCC in the industrial manufacturing
NoName [44]	Three-layer	Task module	Sensor node without OS	Apply MCC in the air quality monitoring
NoName [48]	Hybrid	Task module	Android	Implement the MCC prototype
HyMobi [51]	Hybrid	Method	Android	Implement the MCC prototype and improve the spectrum of offloading opportunities

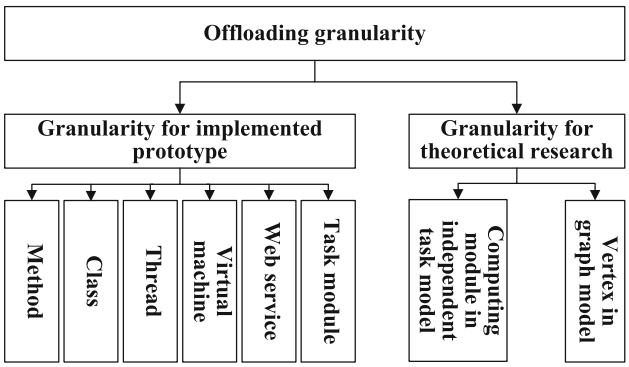

Fig. 2 Categories of the offloading granularity

represents the execution sequence. Some components of the mobile application are unoffloadable because they have to operate MD hardware (e.g., sensors or screen), and the corresponding vertexes need to be marked as unoffloadable. The indicator is a binary, which can be set as 1/0 if the vertex is unoffloadable and can be set as 0/1 if the vertex is offloadable. The edge represents the interactive relationship between two vertexes, and its weight denotes the amount of interactive data.

3 Key technologies in the computation offloading system

To promote the large-scale development of computation offloading in MCC, all the transactions should not be coupled together but decomposed into components. These components are operated in the form of system under principles of high cohesion and low coupling. The computation offloading system has three basic components: MUs, ASOs, and COs [18], and their relationship is shown in Fig. 3. MUs purchase cloud application services provided by ASOs and offload their OUs to ASOs. ASOs rent virtual resources from COs and develop various cloud application services. ASOs do not need to purchase and maintain their computing hardware equipments but rent the required resources from COs. This mode saves the cost of equipment purchase and maintenance for ASOs, and helps them pay more attention to develop the cloud application service and to improve the quality of service (QoS). Only when ASOs continuously develop high-quality cloud application services can they attract more MUs and improve the system viscosity. COs are responsible for infrastructure construction and physical resource operation. COs need to manage the physical resources efficiently to optimize their profits or reduce their energy consumption. In MCC architectures that have the edge layer, COs face the edge equipment deployment problem, which involves

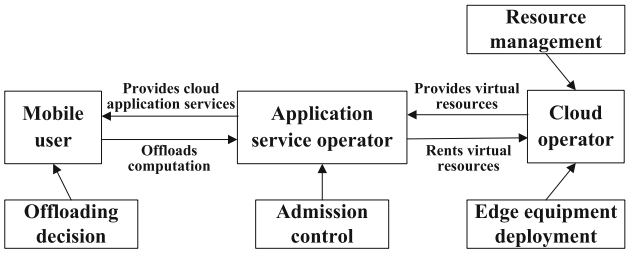

Fig. 3 The basic components and key technologies in the MCC computation offloading system

how to place edge equipments efficiently. These two operators are not completely independent, and they can overlap with each other. For example, a CO can use its resources to develop cloud application services and then becomes an ASO. Offloading decision, admission control, resource management, and edge equipment deployment are these components' technical challenges. In this section, we present a comprehensive review of the existing work that aims to solve these challenges.

3.1 Offloading decision

In the computation offloading system, MUs face the offloading decision problem. Unreasonable offloading decision can not improve the MD performance but results in more energy and time consumption due to additional data transmission on wireless networks. Different from the traditional "client-server" mode, which offloads all OUs to the server, MUs need to decide whether to offload an OU according to their optimization targets [34]. At the same time, the offloading decision in MCC is different from that in grid computing and multiprocessor, in which its optimization goal is to balance the load and to minimize the edge cut and the offloading volume [73]. MD is much weaker than the cloud, and MDs are connected to the cloud via wireless networks, which force MUs to take time and energy consumption from data transmission on wireless networks into account when making offloading decisions. Existing research can be classified according to the cloud application service site (mono-site or multi-site) and decision mode (static or dynamic). The work on offloading decision is compared in Table 4.

(1) Mono-site Offloading Decision versus Multi-site Offloading Decision

In the mono-site offloading decision, there is only one ASO that provides cloud application service to MDs. The OUs are divided into two parts, i.e., the local MD part and the ASO part. These divided OUs are executed locally on the MD or offloaded to the single ASO. Many ASOs offer the same services for MUs in the cloud application service market [74], and a MU can pay multiple ASOs to offload OUs to these ASOs. In the multi-site offloading decision, OUs are divided into $(k+1)$ parts, i.e., the local MD part and k $(k \geq 2)$ ASO parts. The solution space of the multi-site offloading decision is larger than that of the mono-site offloading decision, which increases rapidly with the number of ASOs and improves the complexity of finding the offloading strategy. ASOs are connected through the high-speed wired networks. For MDs, the communication energy consumption among ASOs is negligible, and the communication time is very short. Therefore, compared with mono-site offloading, the communication consumption of multi-site offloading is less. Besides, multiple ASOs provide MUs with more choices and more reliable services. MUs can continue to offload their OUs to other ASOs when some ASOs crash or disconnect. For example, as one extreme, if one of two ASOs in the multi-site computation offloading crashes or disconnects, OUs can continue to be offloaded to the ASO that works normally. Also, as the other extreme, if the only one ASO in the mono-site computation offloading crashes or disconnects, the computation offloading stops working.

(2) Static Offloading Decision versus Dynamic Offloading Decision

The static offloading decision is made through program analysis in the application development stage, and the offloading strategies do not change. The problem of how to make static offloading decisions is usually converted to the static application partitioning problem, in which a mobile application is abstracted as a graph, and then the graph is partitioned into several parts. The dynamic offloading decision works at runtime, and its offloading strategies change constantly. MUs connect with ASOs via wireless networks, which are not stable and vary for many reasons, such as the wireless channel fading and channel interference [75]. Moreover, MUs move among different environments, and wireless network conditions change constantly. Spatiotemporally varying wireless networks bring uncertainty to MCC, and the dynamic offloading decision is required to adapt to the varying environments. During the execution of mobile applications, MDs require real-time offloading strategies to indicate whether to offload their OUs. The key to making a dynamic offloading decision is to balance the decision time and strategy accuracy [70]. The accuracy of strategies found by offloading decision algorithms is positively correlated with their execution time. Offloading decision algorithms need to consume much time to find the accurate result, which makes them unable to adapt to dynamic mobile cloud environment, especially the fast changing environment. They may be effective when the size of offloading decision problem is small or the environment changes slowly. If the problem size is large or the environment changes fast, their performance will become bad. Therefore, it is important to balance the decision time to adapt to dynamic mobile cloud

Table 4 Comparison of the work on offloading decision

Work	Mono-site	Multi-site	Static	Dyna-mic	Target	Contribution
[59]	✓		✓		Minimize the energy consumption and execution time	Establish a cost solving model and design a decision algorithm based on the consumption weight
[60]		✓	✓		Minimize the average energy consumption, time and cost while satisfying the constraint of bandwidth	Establish a multi-objective optimization model and present a decision algorithm called D-NSGA-II-ELS
[76]	✓			✓	Minimize the energy consumption and execution time	Construct a weighted call graph of different topologies for software applications and propose a Min-Cost Offloading Partitioning (MCOP) algorithm
[61]	✓		✓		Maximize energy saving while maintaining the latency requirement	Establish a constrained optimization model and propose a decision algorithm based on the Lagrange multiplier
[64]		✓	✓		Minimize the energy consumption and execution time	Establish a weighted cost model and propose a decision algorithm based on the genetic algorithm
[65]		✓	✓		Minimize the energy consumption	Propose a decision algorithm based on cyclic random movement genetic algorithm
[62]		✓	✓		Maximize the task utility while satisfying the budget constraint	Establish a constrained optimization model and propose a decision algorithm based on the greedy algorithm
[63]		✓	✓		Minimize the energy consumption and execution time	Formulate the problem using a non-cooperative theoretical game with N players and three pure strategies. Propose a comprehensive proof for the existence of a Nash equilibrium and implement a distributed algorithm
[66]		✓	✓		Minimize the computation and communication costs	Establish a constrained 0-1 integer linear programming (ILP) model and propose a heuristic decision algorithm
[67]		✓	✓		Minimize the energy consumption and execution time	Establish a constrained 0-1 ILP model and propose an energy-efficient offloading decision algorithm
[68]		✓	✓		Minimize the energy consumption, execution time, and service cost	Establish a weighted combinatorial optimization model and propose a heuristic decision algorithm
[57]		✓		✓	Minimize the execution time	Establish a context-aware mixed integer programming model and propose an online decision algorithm based on the rent/buy problem
[58]	✓			✓	Minimize the energy consumption	Establish a weighted ILP model and design a middleware named MACS, which resolves the model when environments change
[77]	✓			✓	Minimize the execution time	Propose a framework for runtime computation repartitioning and design a decision solver that updates offloading strategies according to the network states
[69]		✓		✓	Minimize the energy consumption	Establish a constrained 0-1 ILP model, use the Markov chain to describe wireless channels, and propose a MDP-based decision algorithm
[70]		✓		✓	Minimize the energy consumption and execution time	Establish a weighted combinatorial optimization model and propose a runtime offloading decision algorithm based on the memory-based immigrant adaptive genetic algorithm
[71]		✓		✓	Minimize the energy consumption and execution time	Establish a cooperative runtime decision model and propose a runtime offloading decision algorithm based on the cooperation of online machine learning and genetic algorithm

environment and the strategy accuracy to optimize the offloading target.

3.2 Admission control

After the offloading decision is made, an OU request will be sent to the ASO if the offloading strategy indicate that

the OU should be executed on the cloud. ASOs rent virtual resources from COs and provide cloud application services for MUs using these virtual resources. ASOs are motivated to develop various attractive cloud application service for MUs to achieve more revenues. ASOs also expect to provide services for more OUs as much as possible to increase their revenues. However, ASOs usually rent finite virtual resources from COs to reduce the costs. If ASOs accept and provide services for all OUs, it may lead to resource overload and then affect the QoS. Therefore, the ASO needs admission control to determine whether to accept a new OU according to its load conditions. If an OU is accepted, the ASO allocates resources and then executes it. Admission control couples virtual resource allocation and locates on the side of ASOs, making it different from the hardware resource management of COs. Since many ASOs offer same services for MUs in the cloud application service market [74], a MU can pay any ASOs to offload OUs to these ASOs. ASOs are allowed to reject OU requests when accepting the OU is not revenue effective. If an OU request is rejected, it can be sent to other ASOs that provide the same service. Much existing related work studied the admission control problem with the aim of maximizing the revenue. For example, Jin et al. studied the admission control problem with the goal of maximizing the ASO's revenues while guaranteeing the QoS [78]. They considered three features of two-dimensional resources, uncertainty, and incomplete information in model establishment and algorithm design. These work can also be classified by the type of cloud application services. ASOs only provide one type of cloud application service in some work (e.g., [81, 87, 88]) and provide two or more types of cloud application service in other work(e.g., [78–80, 82–86]). In comparison, admission control in the latter work has higher complexity, but it is closer to the real-life computation offloading system and is more universal. The work on admission control is compared in Table 5.

3.3 Resource management and edge equipment deployment

(1) Resource Management

In the computation offloading system, COs are responsible for the operation and maintenance of physical resources and provide on-demand virtual resource leasing services. COs have a large number of computing equipments, and it takes a lot of energy to maintain their normal operation. The increasing demand for large-scale computing and construction of cloud computing facilities has resulted in annual growth in energy consumption. Data centers are one of the most energy-intensive building types, consuming 10 to 50 times energy per floor space of a typical commercial office building [89]. For COs, unreasonable resource management wastes a lot of financial and material resources, and also leads to environmental problems, which will reduce the CO revenues and is not conducive to the benign operation of MCC. Wireless networks have a serious impact on MCC performance [34], and resource management in MCC has to consider both computing and radio resources. Furthermore, changing wireless networks bring uncertainty to resource management in MCC [7]. Si et al. established a stochastic restless bandits-based resource management model, and used a hierarchy of increasingly stronger LP relaxations to solve the resource management strategy [97]. Jin et al. studied the energy-efficient resource management by two steps [94]. They first studied the deterministic resource management and then studied the stochastic resource management with the consideration of uncertainty caused by wireless networks. They established two models of deterministic and stochastic resource management based on the bin packing model, and proposed two algorithms to solve the resource management strategy. The work on resource management is compared in Table 6.

(2) Edge Equipment Deployment

When MCC architecture has the edge layer, the deployment of edge equipment is the most basic technical challenge and the key of "how to build MCC". Edge equipment is more geographically distributed and more numerous. How to deploy edge equipment reasonably is the first problem to be solved when building MCC. COs are responsible for the operation of physical equipment and provide virtual resource rental services. Unreasonable edge equipment deployment not only increases the deployment cost but also increases the complexity of other problems in MCC. In the Juniper white paper, Brown analyzed the importance of edge equipment deployment and pointed out that correct deployment is critical because capabilities deployed at the edge have a higher operational overhead than centralized deployments [98]. Mao et al. reviewed the research about edge equipment deployment, content caching and mobility management [99]. They analyzed the difference between edge equipment deployment and base station deployment. They pointed out that the former is coupled with computing resources and wireless resources and is strictly subjected to the budget. A simple but inefficient approach is to deploy the edge equipment in full. Although this approach can satisfy the resource requirement and is simple enough, it leads to excessive deployment cost and operation cost to COs. The key point to deploy edge equipment is to make a tradeoff between performance requirements and cost. The work on edge equipment deployment is compared in Table 7.

Table 5 Comparison of the work on admission control

Work	Service type	Target	Contribution
[78]	Multi-type	Maximize the revenues while guaranteeing the QoS	Establish a semi-MDP-based model that considers radio resource variations, computing, and radio resources, and propose a reinforcement learning-based strategy algorithm
[79]	Multi-type	Maximize the revenues	Establish a constrained optimization model and propose a convex linear matrix inequality relaxations-based admission algorithm
[80]	Two-type	Reduce the latency and improve the throughput	Establish a MDP-based model and use the simulation-based optimization algorithm to solve the admission strategy
[81]	Mono-type	Minimize the total energy consumption of MDs under latency constraints	Establish a mixed integer programming model and propose a quantized dynamic programming algorithm to solve the admission strategy
[82]	Multi-type	Maximize the revenues	Establish a semi-MDP-based model and use the policy iteration algorithm to solve the admission strategy
[83]	Multi-type	Maximize the revenues while guaranteeing the QoS	Establish a semi-MDP-based model and use the LP to solve the admission strategy
[84]	Mono-type	Minimize the energy consumption of MDs	Establish a multiple-choice integer programming model and use Ben's genetic algorithm to solve the admission strategy
[85]	Multi-type	Maximize the system-wide performance	Establish a constrained optimization model and use a user ranking criteria-based algorithm to solve the admission strategy
[86]	Multi-type	Maximize the total throughput of all MUs	Establish a constrained optimization model, then convert it as a mixed integer programming model, and propose a two-stage admission algorithm to solve the admission strategy
[87]	Mono-type	Keep the system in the stable operating region	Establish a birth-death process-based model and propose two admission algorithms that are suitable for different scenarios
[88]	Mono-type	Maximize the revenues while guaranteeing the QoS	Establish a non-cooperative and non-zero-sum game model and propose a genetic algorithm-based algorithm to solve the admission strategy

4 Fault tolerance and privacy protection for computation offloading

4.1 Fault tolerance for computation offloading

In MCC, MDs are connected to the cloud through wireless networks, and MDs are heterogeneous with cloud data centers, making it prone to fail in the process of data transmission or/and distributed OU execution. Fault tolerance is critical to ensure the robustness of computation offloading in MCC. We summarize the work on fault tolerance into three categories according to the technology on which it is based. The work on fault tolerance is compared in Table 8.

(1) Checkpoint-based Fault Tolerance

In the computation offloading process without fault tolerance, when a failure occurs, it is required to re-execute the OU or re-transfer the data from the beginning, which leads to more energy consumption and execution time, reduces MCC performance, and even makes MCC invalid. Checkpoint, which stores the parameters needed to restore the OU execution periodically, is usually used to support the fault tolerance. When a failure occurs, the OU restarts its execution from a previously saved checkpoint. The computation offloading process with checkpoint is illustrated in Fig. 4 [112]. Checkpoint needs to send messages periodically to synchronize its parameters, leading to extra data transmission for MCC. The amount of synchronization message and the number of checkpoints determine the consumption of fault tolerance. However, the classical checkpoint technology, which is not specifically designed for MCC, does not optimize the consumption caused by saving checkpoints. Deng et al. found that although the classical checkpoint technology can avoid re-executing OUs from the beginning, it cannot guarantee to save the total execution time and energy consumption [113]. They proposed a fault-tolerant mechanism that makes a trade-off between waiting for reconnection by the fault tolerance and directly restarting an OU from the beginning. Some researchers improved the classical checkpoint technology to make it suitable for MCC. For example, Houssem et al. proposed an efficient collaborative checkpointing algorithm that minimizes the number of checkpoints and avoids blocking MDs [114]. Cao and Singhal proposed the concept of mutable checkpoint to solve the problem that traditional checkpointing algorithms suffer from high extra overhead [115]. They designed an efficient checkpointing algorithm that can save checkpoints anywhere to avoid the

Table 6 Comparison of the work on resource management

Work	Target	Contribution
[90]	Maximize the revenues while satisfying the QoS	Establish a restless bandits-based model and use the heuristic algorithm to develop the resource management strategy
[91]	Increase the MD battery life	Establish an adaptive model and use an artificial neural network to predict future required resources
[75]	Maximize the revenues while meeting MUs' demands	Establish three optimization models based on LP, stochastic programming, and robust optimization, respectively. Use LP to solve the management strategy
[92]	Enable efficient selection of cloud participants and provide a stable cloud	Propose a collaborative autonomic resource management system to allocate resources according to the changing MCC environments
[93]	Manage the QoS assurance	Establish a QoS-aware resource management system based on the fuzzy cognitive map and use a prediction algorithm to predict CO's mode, which should satisfy the basic QoS threshold and have the lowest cost
[94]	Minimize the energy consumption	Establish the deterministic and stochastic resource management optimization models. Propose two algorithms based on the adaptive group genetic algorithm and Monte Carlo simulation
[95]	Fault tolerance	Propose an entropy-based grouping technique for resource management
[96]	Minimize the delay and energy consumption	Propose an efficient hierarchical resource sharing management architecture comprised of three domains include Global Cloud Server (GCS), Local Internet Service Provider Server (LISPS), and Gateway Server (GS). Proposed a fuzzy rule-based scheme and a Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS)-based foglet selection scheme
[97]	Minimize the cloud-based video adaptation and transmission distortion	Establish a stochastic restless bandits-based optimization model and use a hierarchy of increasingly stronger LP relaxations to solve the management strategy

overhead of transferring large amounts of data on the wireless network.

(2) Replication-based Fault Tolerance

Replication, which replicates multiple OU replicas to support fault tolerance, is usually used when the offloading granularity is coarse (e.g., VM and class). Chen et al. developed an energy-optimized fault-tolerant MCC framework based on the "*k*-out-of-*n*" system, which ensures that a system of *n* components operates correctly as long as *k* or more components work [116]. That is to say, their framework ensures MCC work well as long as *k* out of *n* nodes are accessible. Li et al. proposed an energy-efficient fault-tolerant replica management policy with the deadline and budget constraints to solve the management and overhead issues caused by replication [117]. Stahl et al. designed a platform, which supports fault tolerance to stream services by replicating processing, to provide excellent flexibility at the cost of performing large-scale OUs [118].

(3) Other Fault-tolerant Technologies

There are some other technologies used in fault tolerance for MCC. Zhou and Buyya thought only using one fault-tolerant technology is not suitable in computation offloading due to MCC's heterogeneity [119]. They combined checkpoint and replication technologies to provide more efficient fault tolerance for computation offloading. They proposed a group-based fault-tolerant algorithm that considers the properties of different machine groups and adaptively selects either checkpoint or replication as the fault-tolerant policy. Park et al. combined checkpoint and replication technologies to provide more efficient fault tolerance for resource management [95]. They classified MDs into groups according to the availability and mobility, and selected either checkpoint or replication according to the group characteristics. Lakhan and Li combined offloading decision with fault tolerance and proposed an offloading decision algorithm that determines application partitioning at runtime and adopts the fault-aware policy that merges detection and retries strategy to deal with any kind of failures [120]. Raju and Saritha proposed a disease resistance-based fault-tolerant framework named DRFT, which consists of four main modules as the monitoring module, response module, knowledge module, and memory module [121]. DRFT regards the VM failure as the virus in human body and uses the human anti-virus mechanism to repair it.

4.2 Privacy protection for computation offloading

Compared with traditional cloud computing, which is accessed through the wired network, MCC is connected through the wireless network and is more vulnerable. MDs usually contain MUs' high-privacy personal information.

Table 7 Comparison of the work on edge equipment deployment

Work	Scenario	Target	Contribution
[100]	Large-scale IoT	Minimize the number of edge nodes and provide real-time processing service for IoT	Propose an edge equipment deployment approach that considers both data flow differences and wireless network differences
[101]	Cellular network	Minimize the energy consumption with the delay constraint	Establish a constrained optimization model and propose an energy-efficient deployment approach based on the particle swarm optimization algorithm
[102]	Cellular network	Minimize the access delay while balancing the load	Establish a constrained optimization model and use mixed ILP to solve the deployment strategy
[103]	Cellular network	Optimize the tradeoff between the deployment cost and end-to-end delay	Establish a constrained optimization model and use mixed ILP to solve the deployment strategy
[104]	Cellular network	Minimize the deployment cost and end-to-end delay	Establish a constrained optimization deployment model and an integer programming-based resource allocation model. Propose a Lagrangian heuristic algorithm to solve the deployment strategy and a heuristic algorithm to solve the allocation strategy
[105]	General scenarios	Reduce the data link congestion	Propose an AP sorting approach, and then deploy cloudlets regarding the sorting results
[106]	IoT	Minimize the access delay	Establish a constrained optimization model. Propose an enumeration-based optimal deployment algorithm and a ranking-based near-optimal deployment algorithm
[107]	Smart manufacturing	Minimize the delay and deployment cost	Establish a weighted constrained optimization model and propose an improved K-means clustering algorithm-based deployment algorithm
[108]	Smart manufacturing	Minimize the computing response time and realize the load balancing	Establish a weighted constrained optimization deployment model and propose a deployment algorithm based on the space-time characteristics
[109]	Intelligent logistics	Minimize the deployment cost	Establish an integer programming-based model and propose a meta-heuristic algorithm that incorporates discrete monkey algorithm to solve the deployment strategy
[110]	General scenarios	Minimize the deployment cost and delay	Establish a multi-objective integer programming-based model and propose a genetic algorithm-based algorithm to solve the deployment strategy
[111]	Cellular network	Maximize the profit under the constrained budget	Establish a 0-1 knapsack-based deployment model and propose a dynamic programming-based deployment algorithm

Privacy leakages and malicious attacks may occur in the process of data transmission or/and OU execution. Privacy protection is needed to ensure the security of computation offloading in MCC. We classify the privacy protection technologies into two categories: passive protection and active protection. The work on privacy protection is compared in Table 9.

(1) Passive Protection

Passive protection avoids or reduces the loss caused by privacy leakages but does not actively eliminate the privacy leakage. Offloading decision with passive protection takes the cost caused by privacy leakages into the total cost or sets privacy as the constraints. For example, Wu and Huang established a multi-factor multi-site risk-based offloading model based on a comprehensive offloading risk evaluation and proposed an ant-based offloading decision algorithm [122]. They took two risk factors (privacy risk and reliability risk) and two benefit factors (execution time and energy consumption) into the offloading decision process. He et al. established a privacy-aware constrained MDP-based offloading decision model, which optimizes delay and energy consumption while considering the location privacy and usage pattern privacy as constraints [123]. They used Q-learning to solve the offloading strategy and used the Lagrange multiplier to handle the constraints. Ma and Mashayekhy studied the offloading decision problem to minimize the delay and energy consumption while considering data protection [124]. They formulated the offloading decision problem as an integer program model in which privacy protection is set as a constraint. Dhanya and Kousalya established a secure offloading decision model that minimizes the transmission cost and security cost [125]. They took the security as an optimization target and proposed a genetic algorithm-based offloading decision algorithm.

(2) Active Protection

Table 8 Comparison of the work on fault tolerance

Work	Fault-tolerant technology	Contribution
[112]	Classical checkpoint	Analyze and verify the performance of fault-tolerant computation offloading systems in mobile wireless environments
[113]	Classical checkpoint	Design a robust computation offloading with a trade-off fault-tolerant mechanism to optimize the MD execution time and energy consumption
[114]	Improved checkpoint	Propose an efficient collaborative checkpointing algorithm that minimizes the number of checkpoints and avoids blocking the MDs
[115]	Improved checkpoint	Design an efficient checkpointing algorithm that saves checkpoints anywhere to avoid the overhead of transferring large amounts of data on wireless networks
[116]	Replication	Develop an energy-efficient framework that ensures nodes retrieve or processes data stored in MCC as long as k out of n storage/processing nodes are accessible
[117]	Replication	Propose an energy-efficient fault-tolerant replica management policy with the deadline and budget constraints to solve the management and overhead issues caused by replication
[118]	Replication	Design a platform that supports fault tolerance to streaming services by replicating processing
[119]	Group-based	Propose a group-based fault-tolerant algorithm that considers the properties of different machine groups and adaptively selects either checkpoint or replication as the fault-tolerant policy
[95]	Group-based	Propose an entropy-based grouping technique for resource management
[120]	Transient failure application partitioning	Propose an offloading decision algorithm that determines application partitioning at runtime and adopts the fault-aware policy that merges detection and retries strategy to deal with any kind of failures
[121]	Disease resistance-based	Propose a disease resistance-based fault-tolerant framework that regards the VM failure as a virus in human body and use human anti-virus mechanism to repair the failure

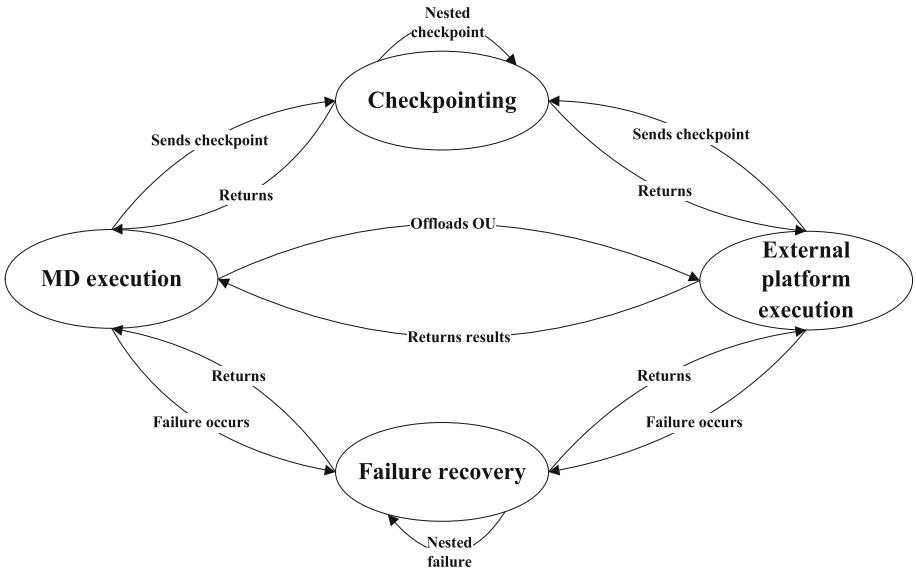

Fig. 4 Computation offloading process with the checkpoint

Active protection eliminates privacy leakages through encryption or non-offloading. For example, Liu and Lu established an energy model for privacy-preserving computation offloading and used homomorphic encryption to protect data in image retrieval before sending data to servers [126]. Wu et al. proposed a trust-aware computation offloading framework that consists of trust evaluation module, filtering module, and selection module [127]. They filtered out resource providers that are untrusted through trust evaluation to ensure that the services provided to MUs are trustworthy. Wang et al. applied deep learning applications on MDs with the help of MCC and proposed a lightweight privacy-preserving mechanism consisting of arbitrary data nullification and random noise addition to protect the sensitive information [128]. Yue et al. implemented a computation offloading system that automatically performs fine-grained privacy-preserving Android application offloading by using static analysis and bytecode

Table 9 Comparison of the work on privacy protection

Work	Protection type	Approach
[122]	Passive protection	Take the privacy risk and reliability risks into the optimization target when making the offloading decision
[123]	Passive protection	Take location privacy and usage pattern privacy as the constraints of the optimization target
[124]	Passive protection	Take privacy protection as a constraint
[125]	Passive protection	Take the security as one optimization target
[126]	Active protection	Encrypt using the homomorphic encryption
[127]	Active protection	Filter out resource providers that are not trusted by MUs
[128]	Active protection	Perturb the local data transformation based on the differential privacy mechanism
[129]	Active protection	Mark the OUs to be protected as unoffloadable OUs
[130]	Active protection	Encrypt using the advanced encryption standard
[131]	Active protection	Encrypt using the anonymous attribute-based encryption

instrumentation techniques [129]. They marked the private data manipulation statements that retrieve and manipulate private OUs as unoffloadable. Saab et al. proposed a minimum-cut algorithm-based runtime offloading decision algorithm that adds the computation cost of performing encryption and decryption to application cost and takes security measures into account [130]. Zhang et al. proposed a privacy protection method named match-then-decrypt, which has a matching operation before decryption operation [131]. They proposed a basic anonymous attribute-based encryption (ABE) construction and then obtained a security-enhanced extension based on strongly existentially unforgeable one-time signatures.

5 Future directions

5.1 Systematic and scaled prototype

For a new technology to have longer vitality, it must be integrated into people's daily life and change people's life or work styles. To promote MCC computation offloading, it is necessary to systematize MCC and to decouple the transactions involved during the computation offloading process. We review four key technologies for MUs, ASOs, and COs, which are three basic components of the computation offloading system in MCC. Most of these technologies are theoretical, and the goal is to optimize some system indicators (e.g., minimizing the energy consumption, minimizing the execution time, maximizing the revenues, etc.). As mentioned in Sect. 2.2, current MCC prototypes are small-scale, and their goals are mainly to verify that MCC can indeed save energy or time. MCC is a combination of many well-known technologies, which cannot make a deep impression on people. Just as cloud computing, which was controversial in the early days, the feasibility and practicality of MCC are also controversial now. A systematic and large-scale prototype of MCC computation offloading is needed to show its feasibility. Only when people actually touch and use MCC can they really accept it. Traditional cloud computing is more like the B2B mode, in which its users are enterprise users, and have less impact and requirements on MUs. MCC is more like the B2C mode and requires deep MU participation (e.g., refactoring the mobile applications or changing the MD OS). Therefore, although it can bring greater changes to people's life and work, it is still a challenge to achieve systematic and large-scale MCC prototype.

MCC will eventually become an infrastructure, just like the power system. Whether it can become the essential infrastructure for people's life depends on its user viscosity. The ability of MCC to provide MUs with various applications is the basis to maintain the user viscosity. The current goal of MCC computation offloading is simply to save energy or time, just as the electricity was used to light up incandescent lamps in the early eras. To promote the large-scale development of MCC, it is necessary to provide MUs with more MCC applications like electricity-based "refrigerators", "air conditioners" and "washing machines" in the power system. Therefore, in the construction process of systematic and large-scale MCC prototype, the development of MCC-based applications is also a very important research and business direction.

5.2 Other technologies

This section gives an outlook on technologies that need to be paid attention to in the future. We discuss these technologies from the perspective of "device-pipe-cloud".

(1) Technologies for the "Device"

For MDs, two technologies need to be paid more attention to in the future:

(1.1) *Portability* MCC aims to free MDs from heavy local workloads and to make MD a simple modem connecting human and electromagnetic signal-based network. The portability of MDs must be strengthened further to achieve the purpose of helping humans access the information network anytime and anywhere. One of the most direct ways to improve portability is to reduce the MD volume. However, in the existing touch-based interaction mode, too small a volume is not conducive to human-computer interaction. Therefore, it is necessary to study portable MDs with a new human-computer interaction mode. Wearable devices, such as smart glasses (e.g., Google Glass), are portable MDs that can be used as the carriers for MCC in the future. At present, there has been some research (e.g., [132–136]) attempts to combine MCC and smart glasses. Furthermore, the brain-computer interface, as an interactive way to create a connection between the brain and external platforms, provides more possibilities for the miniaturization and portability of MD. There are also related studies (e.g., [137, 138]) that combine brain-computer interface with MCC.

(1.2) *Distributed computation offloading platform* MCC computation offloading belongs to the category of distributed computing. However, existing mobile OSs, such as Android and iOS, are weak to support MCC, and most of the current MCC prototypes implement computation offloading by refactoring mobile applications. For example, to automate the refactoring process, Zhang et al. implemented a refactoring tool, named DPartner[134], to automatically transform the Android application bytecode into MD and cloud patterns. Although refactoring mobile applications can implement MCC computation offloading functionally, it is inefficient. It is necessary to design a distributed computation offloading platform for MCC. The difficulty lies in the heterogeneity, that is, the computing devices in MCC are quite different, especially in the CPU architecture and computing power of MD and cloud. Therefore, how to design a distributed computation offloading platform that satisfies the MCC requirements is an important research direction in the future. At present, there is some related work (e.g., KubeEdge [140], K3s [141], MicroK8s [142], and FLEDGE [143]) in this direction.

(2) Technologies for the "Pipe"

As the "pipe" connecting MD and cloud, wireless networks have a great impact on MCC computation offloading and even can make computation offloading invalid. There are two characteristics of the wireless network that affect the computation offloading. One is that the wireless network consumes MD energy when transmitting data, and the other is that the wireless network takes more time in data transmission due to its limited bandwidth. The future research on "pipe" should be towards the wireless network technology with low energy consumption and large bandwidth. There are two research directions worthy of attention:

(2.1) *New generation of wireless network technology* For example, 5G is the new generation of cellular mobile communication technology, which aims at improving data rate, reducing delay, saving energy, reducing cost, and improving system capacity. According to the key 5G characteristics defined by International Telecommunications Union (ITU), the peak data rate (the maximum data rate the MU can achieve under ideal conditions) is 20Gbit/s, the MU experience data rate (the lowest ubiquitous data rate in coverage) is 100Mbit/s, the over-the-air delay is 1ms, and the energy efficiency (both the network side and MD side) will increase 100 times [144]. Wi-Fi 6, also known as "802.11ax Wi-Fi", is the new generation of WLAN technology. Compared with previous generations of Wi-Fi technology, Wi-Fi 6 has a faster data rate (the peak data rate is 9.6Gbit/s), lower delay, and more power saving. Wi-Fi 6 applies the target wake time technology, which allows the active planning of communication time between the MD and wireless router to reduce the wireless network antenna usage and signal search time, to reduce the MD energy consumption [145].

(2.2) *Integrating existing technologies to improve the wireless network performance* Concurrent multiple transfer (CMT) provides a solution to integrate existing wireless networks. At present, most MDs are equipped with multiple wireless network interfaces (e.g., smartphones are equipped with the cellular network interface and Wi-Fi interface), but traditional network protocols can only use one network interface to transmit data at the same time. To solve this problem, CMT, which uses multiple wireless network interfaces to transmit data at the same time, is proposed. There have been some CMT protocols, such as stream control transmission protocol (SCTP) [146] and multi-path transmission control protocol (MPTCP) [147]. These technologies provide a realistic basis for the application of CMT in MCC. MD energy consumption can be reduced, and network bandwidth can be increased by reasonably data scheduling or path selection with CMT [148–151]. Jin et al. explored the usage of CMT in MCC computation offloading to combat the challenges caused by wireless communication, and the results show that using CMT can further save energy and time [70]. CMT utilizes the idle wireless network bandwidth and increases its bandwidth by simultaneously using these bandwidths. Large bandwidth can reduce the delay caused by wireless network data transmission. Different wireless networks have different energy characteristics, and MD energy consumption can be reduced by optimizing data scheduling according to these characteristics. Besides enhancing MCC's efficiency, CMT can also improve MCC's

reliability. In MCC, MDs are connected to the cloud through unreliable wireless networks, which is easily influenced by the outside environment. The failed wireless network will disable MCC. MCC can use multiple wireless networks with the help of CMT and switch to good networks when some wireless networks fail.

(3) Technologies for the "Cloud"

As the external resource platform, the bottom layer of the cloud is composed of a large number of data centers. Running current data centers requires a lot of power. For example, data centers have used approximately 2% of the total electricity in America [89]. As mentioned in Sect. 1, the number of MDs will be huger in the future. To provide cloud resources for these MDs, more data centers need to be built, which in turn will lead to more energy consumption and cause more serious environmental problems. *Therefore, energy-saving for the cloud will be an important and urgent problem to be solved*. In addition to using the resource management and edge equipment deployment technologies described in Sect. 3.3 to save energy, some other technologies should be noted. Renewable energy, such as solar energy and wind energy, can be used to supply power for the more dispersed edge equipment [152–154]. The centralized remote cloud data center can save its energy consumption by optimizing the cooling technology [155–157].

6 Conclusion

With people's life and work more and more dependent on MD, the limited resources of MD cause a lot of inconveniences. MCC computation offloading is an efficient way to solve the problem that MD resources are limited. It aims to offload OUs from MDs to the external platforms and to free MDs from heavy local workloads. MCC has attracted wide attention because of its tremendous potential, and a lot of research on it has been done. In this survey, we presented a comprehensive overview and outlook of research on computation offloading in MCC. Researchers have given different definitions of MCC based on their research scenarios. We summarized the MCC architecture and offloading granularity, which are fundamental concepts of MCC, to classify these definitions. The MCC computation offloading system is decomposed into three basic components: MUs, ASOs, and COs. Offloading decision, admission control, resource management, and equipment deployment are their technical challenges. We conducted a comprehensive literature review on four key technologies used to solve these challenges. The wireless network connection and heterogeneity are the most important features of MCC, making failures and privacy leakages occur easier. We reviewed fault tolerance and privacy protection, which are two important technologies to support computation offloading. Finally, we presented the research outlook of the systematic prototype and other technologies from the perspective of "device-pipe-cloud".

Acknowledgements This work was supported by the Natural Science Basic Research Program of Shaanxi (No. 2021JQ-719), the Science and Technology Project of Shaanxi (No. 2019ZDLGY07-08), the Key Special Project of China High Resolution Earth Observation System Young Scholar Innovation Fund (No. GFZX04061502), the National Natural Science Foundation of China (No. 62002289), and the Special Funds for Construction of Key Disciplines in Universities in Shaanxi.

Declarations

Conflict of interest The authors declare that they have no competing interest.

References

1. Cisco (2020). *Cisco annual internet report*, Retrieved March 1, 2021, from https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.
2. Meeker, M. (2019). *Internet trend report 2019*. Retrieved March 1, 2021, from https://www.bondcap.com/report/itr19/.
3. Kwak, J., Kim, Y., Lee, J., & Chong, S. (2015). DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems. *IEEE Journal on Selected Areas in Communications, 33*(12), 2510–2523. https://doi.org/10.1109/JSAC.2015.2478718.
4. IEA. (2017). *Energy technology perspectives 2017*. Retrieved March 1, 2021, from https://www.iea.org/reports/energy-technology-perspectives-2017.
5. Kumar, K., Liu, J., Lu, Y., & Bhargava, B. (2013). A survey of computation offloading for mobile systems. *Mobile Networks and Applications, 18*(1), 129–140. https://doi.org/10.1007/s11036-012-0368-0.
6. Mell, P., & Grance, T. (2011). The NIST definition of cloud computing. *Communications of the ACM, 53*(6), 50–51. https://doi.org/10.6028/NIST.SP.800-145.
7. Qi, H., & Gani, A. (2012). Research on mobile cloud computing: Review, trend and perspectives. In *2nd international conference on digital information and communication technology and its applications* (pp. 195–202). https://doi.org/10.1109/DICTAP.2012.6215350.
8. Khan, A. U. R., Othman, M., Madani, S. A., & Khan, S. U. (2014). A survey of mobile cloud computing application models. *IEEE Communications Surveys and Tutorials, 16*(1), 393–413. https://doi.org/10.1109/SURV.2013.062613.00160.
9. Liu, F., Shu, P., Hai, J., Ding, L., Jie, Y., Di, N., & Bo, L. (2013). Gearing resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications. *IEEE Wireless Communications, 20*(3), 14–22. https://doi.org/10.1109/MWC.2013.6549279.
10. Allied Analytics LLP. (2017). *Mobile cloud market by application-global opportunity analysis and industry forecast*. Retrieved March 1, 2021, from https://www.researchandmarkets.com/reports/4333216/mobile-cloud-market-by-application-global.

11. Wu, H. (2018). Multi-objective decision-making for mobile cloud offloading: A survey. *IEEE Access, 6*(2018), 3962–3976. https://doi.org/10.1109/ACCESS.2018.2791504.
12. Mach, P., & Becvar, Z. (2017). Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys and Tutorials, 19*(3), 1628–1656. https://doi.org/10.1109/COMST.2017.2682318.
13. Bhattacharya, A., & De, P. (2017). A survey of adaptation techniques in computation offloading. *Journal of Network and Computer Applications, 78*(2017), 97–115. https://doi.org/10.1016/j.jnca.2016.10.023.
14. Kumar, K., Liu, J., Lu, Y., & Bhargava, B. (2013). A survey of computation offloading for mobile systems. *Mobile Networks and Applications, 18*(1), 129–140. https://doi.org/10.1007/s11036-012-0368-0.
15. Khan, M. A. (2015). A survey of computation offloading strategies for performance improvement of applications running on mobile devices. *Journal of Network and Computer Applications, 56*(2015), 28–40. https://doi.org/10.1016/j.jnca.2015.05.018.
16. Chen, Z., & Cheng, S. (2019). Computation offloading algorithms in mobile edge computing system: A survey. In *2019 international conference of pioneering computer scientists, engineers and educators* (pp. 217–225). https://doi.org/10.1007/978-981-15-0118-0_17.
17. Shakarami, A., Shahidinejad, A., & Ghobaei-Arani, M. (2020). A review on the computation offloading approaches in mobile edge computing: A game-theoretic perspective. *Software: Practice and Experience, 50*(9), 1719–1759. https://doi.org/10.1002/spe.2839.
18. Dinh, H. T., Lee, C., Niyato, D., & Wang, P. (2013). A survey of mobile cloud computing: Architecture, applications, and approaches. *Wireless Communications and Mobile Computing, 13*(18), 1587–1611. https://doi.org/10.1002/wcm.1203.
19. Fernando, N., Loke, S. W., & Rahayu, W. (2013). Mobile cloud computing: A survey. *Future Generation Computer Systems, 29*(1), 84–106. https://doi.org/10.1016/j.future.2012.05.023.
20. Murugesan, S., & Bojanova, I. (2016). *Encyclopedia of cloud computing*. John Wiley & Sons, Inc. https://doi.org/978-1-118-82197-8.
21. Marinelli, E. E. (2009). *Hyrax: Cloud computing on mobile devices using MapReduce*. Carnegie Mellon University.
22. Mtibaa, A., Harras, K. A., & Fahim, A. (2013). Towards computational offloading in mobile device clouds. In *5th international conference on cloud computing technology and science* (pp. 331–338). https://doi.org/10.1109/CloudCom.2013.50.
23. Shi, C., Lakafosis, V., Ammar, M. H., & Zegura, E. W. (2012). Serendipity: Enabling remote computing among intermittently connected mobile devices. In *13th ACM international symposium on mobile ad hoc networking and computing* (pp. 145–154). https://doi.org/10.1145/2248371.2248394.
24. Fernando, N., Loke, S. W., & Rahayu, W. (2019). Computing with nearby mobile devices: A work sharing algorithm for mobile edge-clouds. *IEEE Transactions on Cloud Computing, 7*(2), 329–343. https://doi.org/10.1109/TCC.2016.2560163.
25. Zhang, H., Liu, B., Susanto, H., Xue, G., & Sun, T. (2016). Incentive mechanism for proximity-based mobile crowd service systems. In *35th annual IEEE international conference on computer communications* (pp. 1–9). https://doi.org/10.1109/INFOCOM.2016.7524549.
26. Shi, T., Yang, M., Li, X., Lei, Q., & Jiang, Y. (2016). An energy-efficient scheduling scheme for time-constrained tasks in local mobile clouds. *Pervasive and Mobile Computing, 27*(1), 90–105. https://doi.org/10.1016/j.pmcj.2015.07.005.
27. Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing, 8*(4), 14–23. https://doi.org/10.1109/MPRV.2009.82.
28. Hua, X., Dong, R., Peng, X., & Zhao, W. Y. (2015). Cloudlet scheduling mechanism research based on the statistical forecasting. *Journal of Chinese Computer Systems, 37*(3), 406–411.
29. Praseetha, V. M., & Vadivel, S. (2014). Face extraction using skin color and PCA face recognition in a mobile cloudlet environment. In *4th IEEE international conference on mobile cloud computing, services, and engineering* (pp. 1–5). https://doi.org/10.1109/MobileCloud.2016.11.
30. Jia, M., Cao, J., & Liang, W. (2017). Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing, 5*(4), 725–737. https://doi.org/10.1109/TCC.2015.2449834.
31. Haber, E. E., Alameddine, H. A., Assi, C., & Sharafeddine, S. (2019). A reliability-aware computation offloading solution via UAV-mounted cloudlets. In *8th international conference on cloud networking* (pp. 1–6). https://doi.org/10.1109/CloudNet47604.2019.9064038.
32. Jeong, S., Simeone, O., & Kang, J. (2018). Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning. *IEEE Transactions on Vehicular Technology, 67*(3), 2049–2063. https://doi.org/10.1109/TVT.2017.2706308.
33. Islambouli, R., & Sharafeddine, S. (2019). Optimized 3D deployment of UAV-mounted cloudlets to support latency-sensitive services in IoT networks. *IEEE Access, 7*(2019), 172860–172870. https://doi.org/10.1109/ACCESS.2019.2956150.
34. Kumar, K., & Lu, Y. H. (2010). Cloud computing for mobile users: Can offloading computation save energy? *Computer, 43*(4), 51–56. https://doi.org/10.1109/MC.2010.98.
35. Zhang, W., & Wen, Y. (2018). Energy-efficient task execution for application as a general topology in mobile cloud computing. *IEEE Transactions on Cloud Computing, 6*(3), 708–719. https://doi.org/10.1109/TCC.2015.2511727.
36. You, C., Huang, K., & Chae, H. (2016). Energy efficient mobile cloud computing powered by wireless energy transfer. *IEEE Journal on Selected Areas in Communications, 34*(5), 1757–1771. https://doi.org/10.1109/JSAC.2016.2545382.
37. Saha, S., & Hasan, M. S. (2017). Effective task migration to reduce execution time in mobile cloud computing. In *23rd international conference on automation and computing* (pp. 1–5). https://doi.org/10.23919/IConAC.2017.8081998.
38. Abolfazli, S., Sanaei, Z., Alizadeh, M., Gani, A., & Xia, F. (2014). An experimental analysis on cloud-based mobile augmentation in mobile cloud computing. *IEEE Transactions on Consumer Electronics, 60*(1), 146–154. https://doi.org/10.1109/TCE.2014.6780937.
39. Matos, R., Araujo, J., Oliveira, D., Maciel, P., & Trivedi, K. (2015). Sensitivity analysis of a hierarchical model of mobile cloud computing. *Simulation Modelling Practice and Theory, 50*(2015), 151–164. https://doi.org/10.1016/j.simpat.2014.04.003.
40. Shih, C. S., Wang, Y. H., & Chang, N. (2015). Multi-tier elastic computation framework for mobile cloud computing. In *3rd IEEE international conference on mobile cloud computing, services, and engineering* (pp. 1–10). https://doi.org/10.1109/MobileCloud.2015.20.
41. Takahashi, N., Tanaka, H., & Kawamura, R. (2015). Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing. In *3rd IEEE international conference on mobile cloud computing, services, and engineering* (pp. 1–2). https://doi.org/10.1109/MobileCloud.2015.23.

42. Wu, D., Liu, S., Zhang, L., Terpenny, J., Gao, R., Kurfess, T., & Guzzo, J. (2017). A fog computing-based framework for process monitoring and prognosis in cyber-manufacturing. *Journal of Manufacturing Systems, 43*(2017), 25–34. https://doi.org/10.1016/j.jmsy.2017.02.011.
43. Li, X., Wan, J., Dai, H., Imran, M., Xia, M., & Celesti, A. (2019). A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing. *IEEE Transactions on Industrial Informatics, 15*(7), 4225–4234. https://doi.org/10.1109/TII.2019.2899679.
44. Kristiani, E., Yang, C., Huang, C., Wang, Y., & Ko, P. (2020). The implementation of a cloud-edge computing architecture using openstack and kubernetes for air quality monitoring application. *Mobile Networks and Applications, 2020*(7), 1–23. https://doi.org/10.1007/s11036-020-01620-5.
45. Gu, X., Zhang, G., & Zhao, N. (2019). Cooperative mobile edge computing architecture in IoV and its workload balance policy. In *1st international conference on civil aviation safety and information technology* (pp. 1–7). https://doi.org/10.1109/ICCASIT48058.2019.8973164.
46. He, C., Wang, R., & Tan, Z. (2020). Energy-aware collaborative computation offloading over mobile edge computation empowered fiber-wireless access networks. *IEEE Access, 8*(2020), 24662–24674. https://doi.org/10.1109/ACCESS.2020.2969214.
47. Sanaei, Z., Abolfazli, S., Khodadadi, T., & Xia, F. (2013). Hybrid pervasive mobile cloud computing: Toward enhancing invisibility. *Information, 16*(11), 1–12. https://doi.org/10.6084/m9.figshare.1038331.v1.
48. Alonso-Monsalve, S., García-Carballeira, F., & Calderón, A. (2018). A heterogeneous mobile cloud computing model for hybrid clouds. *Future Generation Computer Systems, 87*(2018), 651–666. https://doi.org/10.1016/j.future.2018.04.005.
49. Zhou, B., Srirama, S. N., & Buyya, R. (2019). An auction-based incentive mechanism for heterogeneous mobile clouds. *Journal of Systems and Software, 152*(2019), 151–164. https://doi.org/10.1016/j.jss.2019.03.003.
50. Feng, J., Zhi, L., Wu, C., & Ji, L. (2017). HVC: A hybrid cloud computing framework in vehicular environments. In *5th IEEE international conference on mobile cloud computing, services, and engineering* (pp. 1–8). https://doi.org/10.1109/MobileCloud.2017.9.
51. Flores, H., Sharma, R., Ferreira, D., Kostakos, V., & Yong, L. (2017). Social-aware hybrid mobile offloading. *Pervasive and Mobile Computing, 36*(C), 25–43. https://doi.org/10.1016/j.pmcj.2016.09.014.
52. Pawani, P., Jude, O., Madhusanka, L., Mika, Y., & Tarik, T. (2018). Survey on multi-access edge computing for Internet of Things realization. *IEEE Communications Surveys and Tutorials, 20*(4), 2961–2991. https://doi.org/10.1109/COMST.2018.2849509.
53. Chun, B. G., Ihm, S., Maniatis, P., Naik, M., & Patti, A. (2011). CloneCloud: Elastic execution between mobile device and cloud. In *6th conference on computer systems* (pp. 301–314). https://doi.org/10.1145/1966445.1966473.
54. Cuervo, E., Balasubramanian, A., Cho, D. K., Wolman, A., & Bahl, P. (2010). MAUI: Making smartphones last longer with code offload. In *8th international conference on mobile systems, applications, and services* (pp. 49–62). https://doi.org/10.1145/1814433.1814441.
55. Kosta, S., Aucinas, A., Pan, H., Mortier, R., & Zhang, X. (2012). ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *2012 IEEE INFOCOM* (pp. 945–953). https://doi.org/10.1109/INFCOM.2012.6195845.
56. Kemp, R., Palmer, N., Kielmann, T., & Bal, H. (2010). Cuckoo: A computation offloading framework for smartphones. *In 2010 international conference on mobile computing, applications, and services* (pp. 59–79). https://doi.org/10.1007/978-3-642-29336-8_4.
57. Zhou, B., Dastjerdi, A., Vahid, C., Rodrigo, N., & Buyya, R. (2018). An online algorithm for task offloading in heterogeneous mobile clouds. *ACM Transactions on Internet Technology, 18*(2), 1–23. https://doi.org/10.1145/3122981.
58. Kovachev, D., Yu, T., & Klamma, R. (2012). Adaptive computation offloading from mobile devices into the cloud. In *10th international symposium on parallel and distributed processing with applications* (pp. 784–791). https://doi.org/10.1109/ISPA.2012.115.
59. Thu, M., & Htoon, E. (2018). Cost solving model in computation offloading decision algorithm. In *9th annual information technology, electronics and mobile communication conference* (pp. 1–5). https://doi.org/10.1109/IEMCON.2018.8615089.
60. Liu, L., Du, Y., & Fan, Q. (2019). A constrained multi-objective computation offloading algorithm in the mobile cloud computing environment. *KSII Transactions on Internet and Information Systems, 13*(9), 4329–4348. https://doi.org/10.3837/tiis.2019.09.001.
61. Khoda, M., Razzaque, M. A., Almogren, A., Hassan, M. M., Alamri, A., & Alelaiwi, A. (2016). Efficient computation offloading decision in mobile cloud computing over 5G network. *Mobile Networks and Applications, 21*(5), 777–792. https://doi.org/10.1007/s11036-016-0688-6.
62. Yang, X., & Bi, R. (2019). Budget-aware equilibrium offloading for mobile edge computing. In *2019 IEEE international conference on smart Internet of Things* (pp. 1–5). https://doi.org/10.1109/SmartIoT.2019.00067.
63. Messous, M. A., Senouci, S. M., Sedjelmaci, H., & Cherkaoui, S. (2019). A game theory based efficient computation offloading in an UAV network. *IEEE Transactions on Vehicular Technology, 68*(5), 4964–4974. https://doi.org/10.1109/TVT.2019.2902318.
64. Goudarzi, M., Zamani, M., & Haghighat, A. T. (2017). A genetic-based decision algorithm for multisite computation offloading in mobile cloud computing. *International Journal of Communication Systems, 30*(10), 1–13. https://doi.org/10.1002/dac.3241.
65. Zhang, W., Bing, G., Shen, Y., Li, D., & Li, J. (2018). An energy-efficient algorithm for multi-site application partitioning in MCC. *Sustainable Computing Informatics and Systems, 18*(6), 45–53. https://doi.org/10.1016/j.suscom.2018.02.008.
66. Sinha, K., & Kulkarni, M. (2011). Techniques for fine-grained, multi-site computation offloading. In *11th IEEE/ACM international symposium on cluster, cloud and grid computing* (pp. 184–194). https://doi.org/10.1109/CCGrid.2011.69.
67. Niu, R., Song, W., & Liu, Y. (2013). An energy-efficient multisite offloading algorithm for mobile devices. *International Journal of Distributed Sensor Networks, 2013*(3), 72–81. https://doi.org/10.1155/2013/518518.
68. Enzai, N. I. M., & Tang, M. (2016). A heuristic algorithm for multi-site computation offloading in mobile cloud computing. *Procedia Computer Science, 80*(C), 1232–1241. https://doi.org/10.1016/j.procs.2016.05.490.
69. Terefe, M. B., Lee, H., Heo, N., Geoffrey, C., & Oh, S. (2016). Energy-efficient multisite offloading policy using Markov decision process for mobile cloud computing. *Pervasive and Mobile Computing, 27*(C), 75–89. https://doi.org/10.1016/j.pmcj.2015.10.008.
70. Jin, X., Liu, Y., Fan, W., Wu, F., & Tang, B. (2017). Multisite computation offloading in dynamic mobile cloud environments. *Science China Information Sciences, 60*(8), 089301. https://doi.org/10.1007/s11432-016-0009-6.
71. Jin, X., Wang, Z., & Hua, W. (2019). Cooperative runtime offloading decision algorithm for mobile cloud computing.

Mobile Information Systems, 2019(1), 8049804. https://doi.org/10.1155/2019/8049804.
72. Huang, D., Wang, P., & Niyato, D. (2012). A dynamic offloading algorithm for mobile computing. *IEEE Transactions on Wireless Communications, 11*(6), 1991–1995. https://doi.org/10.1109/TWC.2012.041912.110912.
73. Vuchener, C., & Esnard, A. (2013). Graph repartitioning with both dynamic load and dynamic processor allocation. In *2013 international conference on parallel computing* (pp. 243–252). https://doi.org/10.3233/978-1-61499-381-0-243.
74. Baranwal, G., & Vidyarthi, D. P. (2014). A framework for selection of best cloud service provider using ranked voting method. In *2014 IEEE international advance computing conference* (pp. 831–837). https://doi.org/10.1109/IAdCC.2014.6779430.
75. Kaewpuang, R., Niyato, D., Wang, P., & Hossain, E. (2013). A framework for cooperative resource management in mobile cloud computing. *IEEE Journal on Selected Areas in Communications, 31*(12), 2685–2700. https://doi.org/10.1109/JSAC.2013.131209.
76. Wu, H., Knottenbelt, W. J., & Wolter, K. (2019). An efficient application partitioning algorithm in mobile environments. *IEEE Transactions on Parallel and Distributed Systems, 30*(7), 1464–1480. https://doi.org/10.1109/TPDS.2019.2891695.
77. Lei, Y., Cao, J., Tang, S., Di, H., & Suri, N. (2016). Run time application repartitioning in dynamic mobile cloud environments. *IEEE Transactions on Cloud Computing, 4*(3), 336–348. https://doi.org/10.1109/TCC.2014.2358239.
78. Jin, X., Hua, W., & Wang, Z. (2020). Task admission control for application service operators in mobile cloud computing. *EURASIP Journal on Wireless Communications and Networking, 2020*(1), 217. https://doi.org/10.1186/s13638-020-01827-w.
79. Guo, S., Wu, D., Zhang, H., & Yuan, D. (2018). Resource modeling and scheduling for mobile edge computing: A service provider's perspective. *IEEE Access, 6*(2018), 35611–35623. https://doi.org/10.1109/ACCESS.2018.2851392.
80. Qi, Y., Tian, L., Zhou, Y., & Yuan, J. (2019). Mobile edge computing-assisted admission control in vehicular networks: The convergence of communication and computation. *IEEE Vehicular Technology Magazine, 14*(1), 37–44. https://doi.org/10.1109/MVT.2018.2883336.
81. Lyu, X., Tian, H., Ni, W., Zhang, Y., Zhang, P., & Liu, R. (2018). Energy-efficient admission of delay-sensitive tasks for mobile edge computing. *IEEE Transactions on Communications, 66*(6), 2603–2616. https://doi.org/10.1109/TCOMM.2018.2799937.
82. Liu, Y., & Lee, M. (2015). An adaptive resource allocation algorithm for partitioned services in mobile cloud computing. In *2015 IEEE symposium on service-oriented system engineering* (pp. 209–215). https://doi.org/10.1109/SOSE.2015.19.
83. Liu, Y., Lee, M., & Zheng, Y. (2016). Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system. *IEEE Transactions on Mobile Computing, 15*(10), 2398–2410. https://doi.org/10.1109/TMC.2015.2504091.
84. Wang, J., Yue, Y., Wang, R., Yu, M., & Yu, R. (2019). Energy-efficient admission of delay-sensitive tasks for multi-mobile edge computing servers. In *25th international conference on parallel and distributed systems* (pp. 747–753). https://doi.org/10.1109/ICPADS47876.2019.00110.
85. Chen, X., Li, W., Lu, S., Zhi, Z., & Fu, X. (2016). Efficient resource allocation for on-demand mobile-edge cloud computing. *IEEE Transactions on Vehicular Technology, 67*(9), 8769–8780. https://doi.org/10.1109/TVT.2018.2846232.
86. Lyazidi, M. Y., Aitsaadi, N., & Langar, R. (2016). Resource allocation and admission control in OFDMA-based cloud-RAN. In *2016 GLOBECOM* (pp. 1–6). https://doi.org/10.1109/GLOCOM.2016.7842217.
87. Khojasteh, H., Misic, J., & Misic, V. B. (2015). Task filtering as a task admission control policy in cloud server pools. In *2015 international wireless communications and mobile computing conference* (pp. 727–732). https://doi.org/10.1109/IWCMC.2015.7289173.
88. Baranwal, G., & Vidyarthi, D. P. (2016). Admission control in cloud computing using game theory. *Journal of Supercomputing, 72*(1), 1–30. https://doi.org/10.1007/s11227-015-1565-y.
89. Office of energy efficiency and renewable energy. (2011). *Data centers and servers*. Retrieved March 1, 2021, from https://www.energy.gov/eere/buildings/data-centers-and-servers.
90. Si, P., Zhang, Q., Yu, F. R., & Zhang, Y. (2014). QoS-aware dynamic resource management in heterogeneous mobile cloud computing networks. *China Communications, 11*(5), 144–159. https://doi.org/10.1109/cc.2014.6880470.
91. Sood, S. K., & Sandhu, R. (2015). Matrix based proactive resource provisioning in mobile cloud environment. *Simulation Modelling Practice and Theory, 50*(2015), 83–95. https://doi.org/10.1016/j.simpat.2014.06.004.
92. Khalifa, A., & Eltoweissy, M. (2013). Collaborative autonomic resource management system for mobile cloud computing. In *4th international conference on cloud computing, GRIDs and virtualization* (pp. 115–121).
93. Zhang, P., & Yan, Z. (2011). A QoS-aware system for mobile cloud computing. In *2011 IEEE international conference on cloud computing and intelligence systems* (pp. 518–522). https://doi.org/10.1109/CCIS.2011.6045122.
94. Jin, X., Liu, Y., Fan, W., Wu, F., & Tang, B. (2018). Energy-efficient resource management in mobile cloud computing. *IEICE Transactions on Communications, E101–B(4),* 1010–1020. https://doi.org/10.1587/transcom.2017EBP3177.
95. Park, J., Yu, H., Hyongsoon, K., & Eunyoung, L. (2016). Dynamic group-based fault tolerance technique for reliable resource management in mobile cloud computing. *Concurrency and Computation: Practice and Experience, 28*(10), 2756–2769. https://doi.org/10.1002/cpe.3205.
96. Ahmad, A., Paul, A., Khan, M., Jabbar, S., Rathore, M., Chilamkurti, N., & Min-Allah, N. (2017). Energy efficient hierarchical resource management for mobile cloud computing. *IEEE Transactions on Sustainable Computing, 2*(2), 100–112. https://doi.org/10.1109/TSUSC.2017.2714344.
97. Si, P., Yu, F. R., & Zhang, Y. (2014). Joint cloud and radio resource management for video transmissions in mobile cloud computing networks. In *2014 IEEE international conference on communications* (pp. 1–6). https://doi.org/10.1109/ICC.2014.6883661.
98. Brown, G. (2016). *Mobile edge computing use cases and deployment options*. Retrieved March 1, 2021, from https://www.juniper.net/assets/us/en/local/pdf/whitepapers/2000642-en.pdf.
99. Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys and Tutorials, 19*(4), 2322–2358. https://doi.org/10.1109/COMST.2017.2745201.
100. Zhao, Z., Min, G., Gao, W., Wu, Y., Duan, H., & Ni, Q. (2018). Deploying edge computing nodes for large-scale IoT: A diversity aware approach. *IEEE Internet of Things Journal, 5*(5), 3606–3614. https://doi.org/10.1109/JIOT.2018.2823498.
101. Li, Y., & Wang, S. (2018). An energy-aware edge server placement algorithm in mobile edge computing. In *2018 IEEE international conference on edge computing* (pp. 66–73). https://doi.org/10.1109/EDGE.2018.00016.
102. Wang, S., Zhao, Y., Xu, J., Jie, Y., & Hsu, C. (2019). Edge server placement in mobile edge computing. *Journal of Parallel*

and Distributed Computing, 127(5), 160–168. https://doi.org/10.1016/j.jpdc.2018.06.008.
103. Fan, Q., & Ansari, N. (2017). Cost aware cloudlet placement for big data processing at the edge. In *2017 IEEE international conference on communications* (pp. 1–6). https://doi.org/10.1109/ICC.2017.7996722.
104. Fan, Q., & Ansari, N. (2019). On cost aware cloudlet placement for mobile edge computing. *IEEE/CAA Journal of Automatica Sinica, 6*(4), 926–937.
105. Yang, G., Sun, Q., Ao, Z., Wang, S., & Li, J. (2016). Access point ranking for cloudlet placement in edge computing environment. In *2016 IEEE/ACM symposium on edge computing* (pp. 1–2). https://doi.org/10.1109/SEC.2016.16.
106. Zhao, L., Sun, W., Shi, Y., & Liu, J. (2018). Optimal placement of cloudlets for access delay minimization in SDN-based Internet of Things networks. *IEEE Internet of Things Journal, 5*(2), 1334–1344. https://doi.org/10.1109/JIOT.2018.2811808.
107. Jiang, C., Wan, J., & Abbas, H. (2020). An edge computing node deployment method based on improved K-means clustering algorithm for smart manufacturing. *IEEE Systems Journal*. https://doi.org/10.1109/JSYST.2020.2986649.
108. Wang, J., Li, D., & Hu, Y. (2020). Fog nodes deployment based on space-time characteristics in smart factory. *IEEE Transactions on Industrial Informatics, 17*(5), 3534–3543.
109. Lin, C. C., & Yang, J. W. (2018). Cost-efficient deployment of fog computing systems at logistics centers in Industry 4.0. *IEEE Transactions on Industrial Informatics, 14*(10), 4603–4611. https://doi.org/10.1109/TII.2018.2827920.
110. Bhatta, D., & Mashayekhy, L. (2019). Generalized cost-aware cloudlet placement for vehicular edge computing systems. In *2019 IEEE international conference on cloud computing technology and science* (pp. 1–8). https://doi.org/10.1109/CloudCom.2019.00033.
111. Laha, M., Kamble, S., & Datta, R. (2020). Edge nodes placement in 5G enabled urban vehicular networks: A centrality-based approach. In *2020 national conference on communications* (pp. 1–5). https://doi.org/10.1109/NCC48643.2020.9056059.
112. Ou, S., Wu, Y., Yang, K., & Zhou, B. (2008). Performance analysis of fault-tolerant offloading systems for pervasive services in mobile wireless environments. In *2008 IEEE international conference on communications* (pp. 1–5). https://doi.org/10.1109/ICC.2008.356.
113. Deng, S., Huang, L., Taheri, J., & Zomaya, A. Y. (2015). Computation offloading for service workflow in mobile cloud computing. *IEEE Transactions on Parallel and Distributed Systems, 26*(12), 3317–3329. https://doi.org/10.1109/TPDS.2014.2381640.
114. Houssem, M., Nadjib, B., Makhlouf, A., & Khan, P. (2015). A new efficient checkpointing algorithm for distributed mobile computing. *Control Engineering and Applied Informatics, 17*(2), 43–54.
115. Cao, G., & Singhal, M. (2001). Mutable checkpoints: A new checkpointing approach for mobile computing systems. *IEEE Transactions on Parallel and Distributed Systems, 12*(2), 157–172. https://doi.org/10.1109/71.910871.
116. Chen, C. A., Won, M., Stoleru, R., & Xie, G. G. (2013). Energy-efficient fault-tolerant data storage and processing in dynamic networks. In *14th ACM international symposium on mobile ad hoc networking and computing* (pp. 281–286). https://doi.org/10.1145/2491288.2491325.
117. Li, C., Wang, Y. P., Chen, Y., & Luo, Y. (2019). Energy-efficient fault-tolerant replica management policy with deadline and budget constraints in edge-cloud environment. *Journal of Network and Computer Applications, 143*(2019), 152–166. https://doi.org/10.1016/j.jnca.2019.04.018.
118. Stahl, P., Broberg, J., & Landfeldt, B. (2017). Dynamic fault-tolerance and mobility provisioning for services on mobile cloud platforms. In *5th IEEE international conference on mobile cloud computing, services, and engineering* (pp. 1–8). https://doi.org/10.1109/MobileCloud.2017.7.
119. Zhou, B., & Buyya, R. (2017). A group-based fault tolerant mechanism for heterogeneous mobile clouds. In *14th EAI international conference on mobile and ubiquitous systems: Computing, networking and services* (pp. 373–382). https://doi.org/10.1145/3144457.3144473.
120. Lakhan, A., & Li, X. (2020). Transient fault aware application partitioning computational offloading algorithm in microservices based mobile cloudlet networks. *Computing, 2020*(102), 105–139. https://doi.org/10.1007/s00607-019-00733-4.
121. Raju, D. N., & Saritha, V. (2016). Architecture for fault tolerance in mobile cloud computing using disease resistance approach. *International Journal of Communication Networks and Information Security, 8*(2), 112–118.
122. Wu, H., & Huang, D. (2014). Modeling multi-factor multi-site risk-based offloading for mobile cloud computing. In *10th international conference on network and service management and workshop* (pp. 1–8). https://doi.org/10.1109/CNSM.2014.7014164.
123. He, X., Liu, J., Jin, R., & Dai, H. (2017). Privacy-aware offloading in mobile-edge computing. In *2017 GLOBECOM* (pp. 1–6). https://doi.org/10.1109/GLOCOM.2017.8253985.
124. Ma, W., & Mashayekhy, L. (2019). Privacy-by-design distributed offloading for vehicular edge computing. In *12th IEEE/ACM international conference on utility and cloud computing* (pp. 1–10). https://doi.org/10.1145/3344341.3368804.
125. Dhanya, N. M., & Kousalya, G. (2015). Adaptive and secure application partitioning for offloading in mobile cloud computing. *Adaptive and Secure Application Partitioning, 536*(1), 45–53. https://doi.org/10.1007/978-3-319-22915-7_5.
126. Liu, J., & Lu, Y. H. (2010). Energy savings in privacy-preserving computation offloading with protection by homomorphic encryption. In *2010 international conference on power aware computing and systems* (pp. 1–5).
127. Wu, D., Shen, G., Huang, Z., Cao, Y., & Du, T. (2015). A trust-aware task offloading framework in mobile edge computing. *IEEE Access, 7*(2019), 150105–150119. https://doi.org/10.1109/ACCESS.2019.2947306.
128. Wang, J., Zhang, J., Bao, W., Zhu, X., Cao, B., & Yu, P. S. (2018). Not just privacy: Improving performance of private deep learning in mobile cloud. In *24th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1–10). https://doi.org/10.1145/3219819.3220106.
129. Yue, D., Mu, Z., Yin, Y., & Tang, T. (2015). Privacy-preserving offloading of mobile app to the public cloud. In *7th USENIX workshop on hot topics in cloud computing* (pp. 1–7).
130. Saab, S. A., Saab, F., Kayssi, A., Chehab, A., & Elhajj, I. H. (2015). Partial mobile application offloading to the cloud for energy-efficiency with security measures. *Sustainable Computing: Informatics and Systems, 8*(2015), 38–46. https://doi.org/10.1016/j.suscom.2015.09.002.
131. Zhang, Y., Chen, X., Li, J., Wong, D., Li, H., & You, I. (2017). Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing. *Information Sciences, 379*(1), 42–61. https://doi.org/10.1016/j.ins.2016.04.015.
132. Fiandrino, C., Allio, N., Kliazovich, D., Giaccone, P., & Bouvry, P. (2019). Profiling performance of application partitioning for wearable devices in mobile cloud and fog computing. *IEEE Access, 7*(2019), 12156–12166. https://doi.org/10.1109/ACCESS.2019.2892508.

133. Chang, W., Yu, Y., Chen, J., Zhang, Z., Ko, S., Yang, T., Hsu, C., Chen, L., & Chen, M. (2019). A deep learning based wearable medicines recognition system for visually impaired people. In *2019 IEEE international conference on articial intelligence circuits and systems* (pp. 1–2).
134. Hou, X., Lu, Y., & Dey, S. (2017). Wireless VR/AR with edge/cloud computing. In *26th international conference on computer communication and networks* (pp. 1–8). https://doi.org/10.1109/ICCCN.2017.8038375.
135. Chang, W., Chen, L., Hsu, C., Chen, J., & Lin, C. (2020). MedGlasses: A wearable smart-glasses-based drug pill recognition system using deep learning for visually impaired chronic patients. *IEEE Access, 8*(2020), 17013–17024. https://doi.org/10.1109/ACCESS.2020.2967400.
136. Golkarifard, M., Yang, J., Huang, Z., Movaghar, A., & Hui, P. (2019). Dandelion: A unified code offloading system for wearable computing. *IEEE Transactions on Mobile Computing, 18*(3), 546–559. https://doi.org/10.1109/TMC.2018.2841836.
137. Blondet, M., Badarinath, A., Khanna, C., & Jin, Z. (2013). A wearable real-time BCI system based on mobile cloud computing. In *6th annual international IEEE EMBS conference on neural engineering* (pp. 1–4). https://doi.org/10.1109/NER.2013.6696040.
138. Borulkar, N., Pandey, P., Davda, C., & Chettiar, J. (2018). Drowsiness detection and monitoring the sleeping pattern using brainwaves technology and IoT. In *2nd international conference on I-SMAC* (pp. 1–4). https://doi.org/10.1109/I-SMAC.2018.8653772.
139. Zhang, Y., Huang, G., Liu, X., Zhang, W., Mei, H., & Yang, S. (2012). Refactoring Android Java code for on-demand computation offloading. *ACM Sigplan Notices, 47*(10), 233–247. https://doi.org/10.1145/2384616.2384634.
140. Xiong, Y., Sun, Y., Xing, L., & Huang, Y. (2018). Extend coud to edge with KubeEdge. In *2018 IEEE/ACM symposium on edge computing* (pp. 373–377). https://doi.org/10.1109/SEC.2018.00048.
141. K3s, Retrieved March 1, 2021, from https://k3s.io/.
142. MicroK8s, Retrieved March 1, 2021, from https://microk8s.io/.
143. Goethals, T., Turck, F. D., & Volckaert, B. (2020). Extending Kubernetes clusters to low-resource edge devices using virtual Kubelets. *IEEE Transactions on Cloud Computing*. https://doi.org/10.1109/TCC.2020.3033807.
144. International Mobile Telecommunications. (2003). *Framework and overall objectives of the future development of IMT-2000 and systems beyond IMT-2000*. Retrieved March 1, 2021, from https://grouper.ieee.org/groups/802/secmail/pdf00204.pdf.
145. Ahmed, T., Krishnan, M. S., & Anil, A. K. (2020). A predictive analysis on the influence of Wi-Fi 6 in fog computing with OFDMA and MU-MIMO. In *4th international conference on computing methodologies and communication* (pp. 1–4). https://doi.org/10.1109/ICCMC48092.2020.ICCMC-000133.
146. Iyengar, J. R., Amer, P. D., & Stewart, R. (2020). Concurrent multipath transfer using SCTP multihoming over independent endto-end paths. *IEEE/ACM Transactions on Networking, 14*(1), 951–964. https://doi.org/10.1109/TNET.2006.882843.
147. Ford, A., Raiciu, C., Handley, M., Bonaventure, O. & Paasch, C. (2013). *TCP extensions for multipath operation with multiple addresses*, Retrieved March 1, 2021, from https://www.rfc-editor.org/info/rfc6824.
148. Wu, J., Cheng, B., & Wang, M. (2016). Energy minimization for quality constrained video with multipath TCP over heterogeneous wireless networks. In *36th international conference on distributed computing systems* (pp. 1–10). https://doi.org/10.1109/ICDCS.2016.25.
149. Lim, Y. S., Chen, Y. C., Nahum, E. M., Towsley, D., & Gibbens, R. J. (2015). Design, implementation, and evaluation of energy-aware multi-path TCP. In *2015 ACM conference on emerging networking experiments and technologies* (pp. 1–13). https://doi.org/10.1145/2716281.2836115.
150. Sarkar, D., & Paul, S. (2006). QRP04-3: Architecture, implementation, and evaluation of cmpTCP westwood. In *2006 GLOBECOM* (pp. 1–5). https://doi.org/10.1109/GLOCOM.2006.437.
151. Yang, W., Li, H., Li, F., Wu, Q., & Wu, J. (2010). RPS: Range-based path selection method for concurrent multipath transfer. In *6th international wireless communications and mobile computing* (pp. 1–5). https://doi.org/10.1145/1815396.1815612.
152. Li, W., Yang, T., Delicato, F. C., Pires, P. F., Tari, Z., Khan, S. U., & Zomaya, A. Y. (2018). On enabling sustainable edge computing with renewable energy resources. *IEEE Communications Magazine, 56*(5), 94–101. https://doi.org/10.1109/MCOM.2018.1700888.
153. Li, L., Rodero, I., Parashar, M., & Menaud, J. M. (2017). Leveraging renewable energy in edge clouds for data stream analysis in IoT. In *17th IEEE/ACM international symposium on cluster, cloud and grid computing* (pp. 1–10). https://doi.org/10.1109/CCGRID.2017.92.
154. Peng, C., Li, D., Tian, F., & Guo, Y. (2017). Renewable energy powered IoT data traffic aggregation for edge computing. In *2018 international conference in communications, signal processing, and systems* (pp. 1–5). https://doi.org/10.1007/978-981-13-6508-9_105.
155. Jiang, W., Jia, Z., Feng, S., Liu, F., & Jin, H. (2019). Fine-grained warm water cooling for improving datacenter economy. In *46th ACM/IEEE annual international symposium on computer architecture* (pp. 474–486). https://doi.org/10.1145/3307650.3322236.
156. Angelis, F. D., & Grasselli, U. (2016). The next generation green data center: A multi-objective energetic analysis for a traditional and CCHP cooling system assessment. In *16th international conference on environment and electrical engineering* (pp. 1–5). https://doi.org/10.1109/EEEIC.2016.7555443.
157. Chiriac, V. A., & Chiriac, F. (2012). Novel energy recovery systems for the efficient cooling of data centers using absorption chillers and renewable energy resources. In *13th intersociety conference on thermal and thermomechanical phenomena in electronic systems* (pp. 814–820). https://doi.org/10.1109/ITHERM.2012.6231510.

Xiaomin Jin received the B.E. and the Ph.D. degrees from Beijing University of Posts and Telecommunications, Beijing, China, in 2012 and 2018, respectively. Now he is a lecturer at School of Computer Science and Technology, Xi'an University of Post and Telecommunication. His research interests include mobile cloud computing and mobile devices.

Wenqiang Hua received the Ph.D. degree from Xidian University, Xi'an, China, in 2018. Now he is a lecturer at School of Computer Science and Technology, Xi'an University of Post and Telecommunication. His research interests include machine learning, deep learning, and PolSAR image processing.

Zhongmin Wang received the Ph.D. degree from the Beijing Institute of Technology, Beijing, China, in 2000. Now he is a professor at School of Computer Science and Technology, Xi'an University of Posts and Telecommunications. His current research interests include embedded intelligent perception, big data processing and application, and affective computing.

Yanping Chen received the Ph.D. degree from Xi'an Jiaotong University, Xi'an, China, in 2007. Now she is a professor at School of Computer Science and Technology, Xi'an University of Posts and Telecommunications. Her current research interests include service computing and network management.