Chapter 2
Mobile Edge Computing

Abstract Mobile edge computing is a promising paradigm that brings computing resources to mobile users at the network edge, allowing computing-intensive and delay-sensitive applications to be quickly processed by edge servers to satisfy the requirements of mobile users. In this chapter, we first introduce a hierarchical architecture of mobile edge computing that consists of a cloud plane, an edge plane, and a user plane. We then introduce three typical computation offloading decisions. Finally, we review state-of-the-art works on computation offloading and present the use case of joint computation offloading.

2.1 A Hierarchical Architecture of Mobile Edge Computing (MEC)

To better understand the internal logic of MEC, we first present a hierarchical architecture that vertically divides the edge computing system into three layers: the user layer, the edge layer, and the cloud layer, as shown in Fig. 2.1. The user layer is distinguished by the wireless communication mode between mobile devices and wireless infrastructures. The edge and cloud layers mainly refer to the computing resources of the edge and cloud servers, respectively.

Devices in the user layer include sensors, smartphones, vehicles, smart meters, and radio-frequency identification devices. These devices access edge servers via wireless communication and then offload computation-intensive tasks to the lightweight, distributed edge servers to process. According to wireless network topology and communication modes, the communication between mobile devices and a wireless infrastructure can be split into the following three modes.

- Heterogeneous network: Next generation wireless networks will run applications that require large demand for high data rates. One solution to help reduce the data rate requirement is the densification of the network by deploying small cells. Such densification results in higher spectral efficiency and can reduce the power consumption of a mobile device due to its communication with small nearby cell base stations. This solution significantly improves network coverage. The concurrent operation of macro base stations (MBSs) and micro, pico, femto, and

Y. Zhang, *Mobile Edge Computing*, Simula SpringerBriefs on Computing 9,
https://doi.org/10.1007/978-3-030-83944-4_2

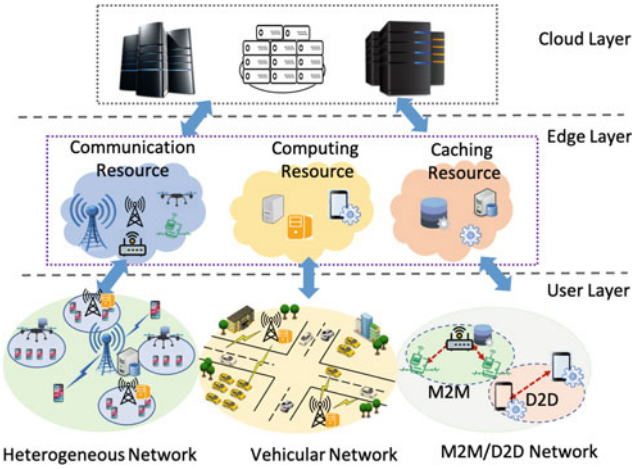

Fig. 2.1 Hierarchical MEC architecture

unmanned aerial vehicle–aided base stations is termed a heterogeneous network. In heterogeneous networks, all base stations are equipped with computational resources and artificial intelligence functions. Resource-limited mobile devices can offload their tasks to these heterogeneous base stations, which can then utilize a fine-grained computational resource allocation policy to process the offloaded tasks.

- Vehicular network: Vehicular networks are inseparable from a smart city environment, owing to several applications that improve the quality of life, safety, and security. A vehicular network is formed among moving vehicles, roadside units, and pedestrians, which can be deployed in rural, urban, and highway environments. Vehicle-to-everything communication allows vehicles to communicate with other vehicles and their surroundings via wireless links. Vehicle-to-everything communication has three main scenarios: vehicle to vehicle, vehicle to infrastructure, and vehicle to pedestrian [12]. Commonly used technologies are dedicated short-range communications, IEEE 802.11p, the IEEE 1609 family of standards, and Long Term Evolution (LTE). With advancements in communication technologies, a number of promising applications are emerging for vehicular networks. These vary from safety applications, such as blind spot warning and traffic light violations to entertainment, such as streaming media, or convenience, such as parking space identification. In vehicular networks, ubiquitous edge resources can be deployed on nearby infrastructures to offer vehicles a high quality of service. Compared to common mobile nodes, vehicles can move at quite high speeds, which causes the topology of a vehicular network to frequently change. Detailed policy design must carefully consider such dynamic network topologies.
- Mobile-to-mobile (M2M)/device-to-device (D2D) networks: M2M is an enabling technology for the Internet of Things, which involves autonomous connectivity and

communication among devices from embedded sensors and actuators to powerful computationally rich devices without human intervention. D2D allows devices to communicate with each other through a direct wireless link without traversing the base station or core network. With the technological advancement of smart devices, more computing and caching resources are distributed among the end users. Computational tasks can thus be offloaded not only to edge servers, but also to devices in D2D and M2M networks.

The edge layer is located in the middle of the hierarchical architecture and consists of multiple distributed edge servers to provide distributed intelligent wireless computing for users. Edge servers can be deployed in the network infrastructure, such as base stations, roadside units, wireless access points, and gateways, or they can be mobile phones, vehicles, tablets, and other devices with computing and storage capabilities. Generally, edge servers are widely distributed in hotspots such as cafes, shopping centers, bus terminals, streets, and parks. Given the proximity of edge servers to end users, computing-intensive and delay-sensitive tasks can be offloaded and accomplished with low latency and high efficiency. There are three types of resources in the edge layer: communication resources, caching resources, and computing resources. Communication resources refer to bandwidth, spectrum, and transmission power. Computing resources mainly refer to CPU cycles. Caching resources are related to the memory capacity on edge servers. Since edge servers are ubiquitously distributed, their computing and caching resources capacities are usually limited. The full use of edge resources requires the joint optimization of communication, caching, and computing resources.

The central cloud layer consists of multiple servers with strong processing, caching, and computing capabilities. With a global view, this layer can leverage advanced techniques such as data mining and big data, for a network-level orchestration shift from reactive to proactive network operation, by predicting events or pre-allocating resources. With their high computing capability and sufficient caching resources, cloud servers can process delay-tolerant applications and store larger or less popular content. Further, the central cloud layer can effectively manage and control multiple edge servers and provide them with secure connections.

2.2 Computation Model

Computation offloading is an approach to offload computation-intensive and delay-sensitive tasks to resource-rich edge servers and/or cloud servers to process. This approach can help prolong the battery life of mobile devices and reduce task processing latency. The key problems in computation offloading are in deciding whether to offload, the amount of the computation task that needs offloading, and which server to offload to. Basically, computation offloading can result in the following three types of decisions [13], as shown in Fig. 2.2:

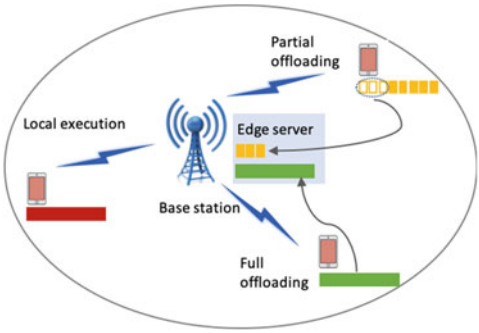

Fig. 2.2 Computation offloading decision

- Local execution: The entire computation task is completed locally. If the computational resources of the edge servers are unavailable or the wireless channel is of poor quality, which can result in high transmission latency, local execution can be preferred.
- Full offloading: The entire computation task is offloaded and processed by an edge server.
- Partial offloading: Part of the computation task is processed locally while the rest is offloaded to an edge server.

The computation offloading decision is very difficult, since it requires considering multiple factors, such as application requirements, the quality of the communication link, and the computing resource capacities of edge servers.

2.2.1 Computation Model of Local Execution

As the noted above, the CPU is the primary engine for computation. The CPU's performance is controlled by CPU cycles f_m. The state-of-the-art mobile CPU architecture adopts an advanced dynamic frequency and voltage scaling technique, which allows for stepping up or down CPU cycles, increasing and reducing energy consumption, respectively. In practice, the value of f_m is bounded by a maximum value, f_{max}, which reflects the limitation of the mobile's computation capability. A computation task can be described as $D \triangleq (d, c, T)$, where d denotes the data size of the computation task, c is the required number of CPU cycles for computing one bit of the computation task, and T denotes the maximum latency allowed to accomplish the task. The local execution time for a computing task D can now be expressed as

$$T^L = \frac{dc}{f_m} \tag{2.1}$$

which indicates that more CPU cycles are required to reduce the execution latency.

Since devices are energy constrained, the energy consumption of local execution is a critical performance metric for computing efficiency. According to [14], the energy consumption of each CPU cycle is given by ςf_m^2, where ς is the effective switched capacitance, depending on the chip architecture. The energy consumption for executing task D with f_m CPU cycles can be derived as

$$E^L = \varsigma dcf_m^2 \tag{2.2}$$

From (2.1) and (2.2), if T^L is greater than the maximum latency or if the device's battery capacity is less than E^L, the device should offload the task to edge servers to process. Otherwise, local execution can support the computation task.

2.2.2 Computation Model of Full Offloading

In this section, we present two computation models of the full offloading for a single-user MEC system and a multi-user MEC system, respectively.

The single-user MEC system is the simplest case and consists of a single device and a single edge server. Denote F_e as the computational resource capacity of the edge server. The device offloads the entire computation task to the edge server to process. The task computation time is thus given by

$$t^{F,computing} = \frac{dc}{F_e} \tag{2.3}$$

Since the process of offloading involves wireless transmission, the total task execution time is the sum of the task computation time and the task transmission time, which can be expressed as

$$T^{F,s} = \frac{dc}{F_e} + \frac{d}{r^s} \tag{2.4}$$

where r^s is the wireless transmission data rate between the device and the edge server. The energy consumption for completing the offloaded computation task also includes two parts: the energy consumption for computation and the energy consumption for wireless transmission. The total energy consumption can be expressed as

$$E^{F,s} = \varsigma dcF_e^2 + p\frac{d}{r^s} \tag{2.5}$$

where p is the transmission power of the device.

In the multi-user MEC system, several devices can be associated with the same edge server and offload their tasks to the edge server simultaneously. In this case, each device is assigned only to a part of the edge server's computational resources. Denote the computation task of device i as $D_i \triangleq (d_i, c_i, T_i)$, where d_i denotes the

data size of the computation task on device i, c_i is the required number of CPU cycles for computing one bit of the computation task, and T_i denotes the maximum latency allowed to accomplish the task. Let f_e^i be the computational resources that the edge server allocates to device i. Since the process of offloading involves wireless transmission, the total task execution time of device i can be expressed as

$$T_i^{F,m} = \frac{d_i c_i}{f_e^i} + \frac{d_i}{r_i^m} \tag{2.6}$$

where r_i^m is the wireless transmission data rate between device i and the edge server.

The corresponding energy consumption of completing the offloaded computation task of device i can be expressed as

$$E_i^{F,m} = \varsigma d_i c_i (f_e^i)^2 + p_i \frac{d_i}{r_i^m} \tag{2.7}$$

where p_i is the transmission power of device i.

Different from the single-user MEC system, devices in the multi-user MEC system share the same computational resources and wireless channel. Therefore, computational resource allocation, channel assignment, bandwidth allocation, and power control should be jointly optimized. Since the total computational resources of the edge server are limited, there is a computational resource constraint (i.e., $\sum_i f_e^i \leq F_e$). A more complex model considering a multi-user multi-MEC server was proposed in [14, 15]. With the dense deployment of MEC servers, a joint user association and computation offloading scheme was designed in [14], and a joint communication resource allocation and computation offloading scheme was designed in [15].

2.2.3 A Computation Model for Partial Offloading

Partial offloading is a very complex process that can be affected by different factors, such as the offloadability of an application [16], the dependency of the offloadable parts [17], and user preferences and channel connection quality [32]. To simplify the description, we assume each computation task can be offloaded and arbitrarily divided into two parts. One part is executed on the device and the other is offloaded to an edge server for edge execution.

Let λ $(0 \leq \lambda \leq 1)$ be the offloading ratio, which represents the ratio of the offloaded task to the total task. That is, an amount λd is offloaded to the edge server to be computed and the rest, $(1 - \lambda)$ d, is computed locally. The task computation time upon partial offloading can be expressed as

$$t^{P,computing} = \frac{(1-\lambda)dc}{f_m} + \frac{\lambda dc}{F_e} \tag{2.8}$$

Since one part of the computation task (i.e., λd) involves wireless transmission, the total time for completing this task can be expressed as

$$T^P = \frac{(1-\lambda)dc}{f_m} + \frac{\lambda dc}{F_e} + \frac{\lambda d}{r} \tag{2.9}$$

The energy consumption required for completing this task consists of three parts:

$$E^P = \varsigma(1-\lambda)dcf_m^2 + \varsigma\lambda dcF_e^2 + p\frac{\lambda d}{r} \tag{2.10}$$

where the first term indicates the local energy consumption for processing the amount $(1-\lambda)\ d$, the second term indicates the energy consumption for processing the amount λd on the edge server, and the third term is the energy consumption of the wireless transmission. In partial offloading, the key problem is to decide the offloading ratio, considering system constraints. For example, if the energy or computational resources of the device are almost used up, offloading the task to the edge server is desirable (i.e., the offloading ratio should be close to one). If the quality of the wireless channel is poor or the available computational resources of the edge server are limited, local execution could be a better choice. Note that the above models can be easily extended to the multi-user MEC system.

2.3 Offloading Policy

The key problem in edge computing is making the offloading decision. According to the previous description, the results of the offloading decision are either local execution, full offloading, or partial offloading. Combining local execution and full offloading, the problem can be modeled as a zero–one binary offloading problem. Partial offloading can be modeled as a continuous offloading decision making problem. First, we introduce the research on binary offloading in the next section.

2.3.1 Binary Offloading

Binary offloading mainly involves small-scale computation tasks that have high computational resource requirements. Such tasks will be offloaded in entirety to the edge server. Computing offloading can effectively reduce the task completion delay and save the energy consumption of devices. When the device does not choose offloading (i.e., local execution), the task completion delay involves only the local task computation time. When the device chooses offloading, the task completion delay involves three parts: (1) the wireless transmission time of the computation task from the device to the edge server, (2) the task computation time spent on the edge server,

and (3) the wireless transmission time of the computation result from the edge server to the device. Similarly, when the device does not offload the task, the total energy consumption required to complete the task includes only local task computation energy consumption. If the device offloads any of the computation task, the total energy consumption consists of two parts: the energy consumption of the wireless transmission from the device to the edge server and the energy consumption of the computation on the edge server.

2.3.1.1 Minimization of Task Execution Delay

The authors in [18] proposed a one-dimensional search algorithm to minimize execution delay. The proposed algorithm can find an optimal offloading decision policy based on the buffer state, available processing power, and channel information. The offloading decision determines whether to process the application locally or at the MEC server. Another idea aimed at minimizing the execution delay was introduced in [20]. Compared to [18], these authors considered users applying dynamic voltage and frequency scaling and proposed a low-complexity Lyapunov optimization-based dynamic computation offloading algorithm. This algorithm allows users to make an offloading decision in each time slot and simultaneously allocates CPU cycles and transmission power. The proposed method can reduce execution times by up to 64% by offloading the computation task to the edge server. Different from the two works focusing on the design of computation offloading algorithms, the authors in [19] proposed an MEC-assisted offloading architecture that allows for deploying intelligent scheduling logic, namely, a mobile edge scheduler, at the MEC without requiring large computational resources at the eNodeB hardware. The introduced mobile edge scheduler runs on the eNodeB. A two-stage scheduling process was proposed to minimize the delay of general traffic flows in the LTE downlink via the MEC server deployed at the eNodeB.

2.3.1.2 Minimization of Energy Consumption

The computation offloading decision to minimize the energy consumption of devices was proposed in [21]. These authors formulated the optimization problem as a constrained Markov decision process. To solve the optimization problem, two types of resource allocation strategies accounting for both computational and radio resources were introduced. The first strategy is based on online learning, where the network adapts dynamically with respect to the application running on the device. The second, precalculated offline strategy is based on prior knowledge of the application properties and statistical behavior of the radio environment. Numerical experiments showed that the precalculated offline strategy can outperform the online strategy by up to 50% for low and medium arrival rates (loads). Since the offline strategy proposed in [21] showed its merits, the authors in [22] proposed two additional offline dynamic programming approaches to minimize the average energy consumption of devices.

One of the dynamic programming approaches to find the optimal radio scheduling offloading policy is deterministic, while the other is randomized. Numerical experiments showed both offline policies can reduce energy consumption compared to offloading-only and static processing strategies. The authors in [22] further extended the work in [23] from single user to multi-user by jointly optimizing resource allocation and computation offloading to guarantee fairness between users, low energy consumption, and average queuing/delay constraints. Another multi-user offloading decision strategy was proposed in [24] to minimize system energy consumption. This paper determined three multi-user types based on the time and energy cost of the task computing process. The first type of user can compute tasks on the MEC server. The second type of user computes the task on local equipment. The third type of user can decide to either implement tasks locally or offload tasks to the MEC server. Based on the user classification, a joint computation offloading and radio resource allocation algorithm was proposed. The proposed algorithm can decrease energy consumption by up to 15% compared to computation without offloading.

2.3.1.3 Trade-Off Between Energy Consumption and Execution Delay

A computation offloading decision for a multi-user multi-task scenario was proposed in [25] to make the trade-off between energy consumption and execution delay. These authors considered jointly the offloading decisions for all the tasks of each user and the sharing of computational and communication resources among all the users as they compete to offload tasks through a wireless link with limited capacity. The computation offloading problem is formulated as a non-convex quadratically constrained quadratic program. To solve this problem, an efficient three-step algorithm was designed that involves semidefinite relaxation, alternating optimization, and sequential tuning. The numerical results showed the proposed algorithm outperformed purely local processing, purely cloud processing, and hybrid local–cloud processing without an edge server. Another algorithm for the computation offloading decision to trade off energy consumption and execution delay was proposed in [26]. The main difference between the works [25, 26] is that the task in [25] can be also offloaded to a remote centralized cloud if the computational resources of the MEC are insufficient. In [26], the authors proposed a computation offloading decision to minimize both the total task execution latency and the total energy consumption of mobile devices. Two cases of mobile devices were considered: devices with a fixed CPU frequency and those with an elastic CPU frequency. In the fixed CPU scenario, a linear programming relaxation–based algorithm was proposed to determine the optimal task allocation decision. In the elastic CPU scenario, the authors first considered an exhaustive search–based algorithm and then utilized a semidefinite relaxation algorithm to find the near-optimal solution.

2.3.2 *Partial Offloading*

The literature cited above focused on binary offloading strategies. In a binary offloading problem, the computing task is considered as a whole. However, in practical applications, computing tasks are often divided into multiple parts [27]. According to the divisible nature of computing tasks, devices can offload part of a task, rather than its entirety, to the edge server. There are thus two types of tasks: (1) tasks that can be divided into multiple discrete segments that can all be offloaded to the MEC server for execution and (2) tasks that can be split into two consecutive parts, non-offloadable and offloadable, and only the offloadable part can be offloaded. Next, we introduce works focused on partial offloading.

2.3.2.1 Minimization of Task Execution Delay

The authors in [28] investigated a latency minimization resource allocation problem for a multi-user offloading system with partial offloading. A partial compression offloading was proposed that has three steps. First, each device compresses part of the raw data locally and then transmits the compressed data to the edge server. Second, the device transmits the remaining part of the raw data to the edge server, which compresses the data. Finally, the edge server combines the two parts of compressed data in the cloud center. A weighted sum latency minimization partial compression offloading problem was formulated and an optimal resource allocation algorithm based on the subgradient was designed. More general work on partial offloading was covered in [29]. The authors jointly considered a partial offloading and resource allocation scheme to minimize the total latency for a multi-user offloading system based on orthogonal frequency division multiple access. The proposed scheme first determines the optimal offloading fraction to ensure that the edge computing delay is less than the local execution delay. Then, the proposed scheme determines how to allocate the communication and computational resources. Additionally, users can make full use of multi-channel transmissions to further reduce the transmission delay for tasks with a large data size. The simulation results show that the proposed scheme achieves 17% and 25% better performance than random and complete offloading schemes, respectively.

2.3.2.2 Minimization of Energy Consumption

In [27], the authors investigated partial computation offloading to minimize the energy consumption of devices by jointly optimizing the CPU cycle frequency, the transmission power, and the offloading ratio. They designed an energy-optimal partial computation offloading algorithm that transformed the non-convex energy consumption minimization problem into a convex one based on the variable substitution technique and obtained a globally optimal solution. The authors also analyzed

the conditions under which local execution is optimal. Analyzing the optimality of total offloading, the authors concluded that total offloading cannot be optimal under dynamic voltage scaling of the device. The authors in [30] proposed a joint scheduling and computation offloading algorithm for multi-component applications using an integer programming approach. The optimal offloading decision involves which components need to be offloaded, as well as their scheduling order. The proposed algorithm provides a greater degree of freedom in the solution by moving away from a compiler predetermined scheduling order for the components toward a more wireless-aware scheduling order. For some component dependency graph structures, the proposed algorithm can shorten execution times by the parallel processing of appropriate components on the devices and in the cloud. To minimize the expected energy consumption of the mobile device, an energy-efficient scheduling policy for collaborative task execution between the mobile device and a cloud clone was proposed in [31]. The authors formulated the energy-efficient task scheduling problem as a constrained stochastic shortest path problem on a directed acyclic graph. They also considered three alternative stochastic wireless channel models: the block fading channel, the independent and identically distributed stochastic channel, and the Markovian stochastic channel. To solve the formulated problem, the authors leveraged a one-climb policy and designed a heuristic algorithm to determine the task execution decision.

2.3.2.3 Trade-Off Between Energy Consumption and Execution Delay

Partial offloading decision considering a trade-off between energy consumption and execution delay was described in [32]. The offloading decision considered four parameters: (1) the total number of bits to be processed, (2) the CPU cycles of the device and of the MEC server, (3) the channel state between the device and the serving femtocell access points, and (4) the device's energy consumption. The joint communication and computational resource allocation problem was formulated as a convex optimization problem. The simulation results indicated that partial offloading could reduce the energy consumption of devices, compared to the case of full offloading, when all the computation tasks are forced to be carried out on either the device or at the femtocell access point. The study in [33] provided a more in-depth theoretical analysis on the trade-off between energy consumption and the latency of the offloaded applications preliminarily handled in [32]. To carry out partial offloading, the authors considered data partition-oriented applications and focused on three parameters of an application: (1) the size of the data, (2) the completion deadline, and (3) the output data size. Then, a joint optimization of the radio and computational resource problem was formulated, and a simple one-dimensional convex numerical optimization technique was utilized to solve it. The authors further demonstrated that the probability of computation offloading is higher when given good channel quality. The authors in [34] considered the trade-off between power consumption and execution delay for a multi-user scenario. The authors formulated a power consumption minimization problem with an application buffer stability constraint. An

online algorithm based on Lyapunov optimization was proposed that decides the optimal CPU frequency of the device for local execution and allocates the transmission power and bandwidth when offloading the application to an edge server. The numerical results demonstrated that computation offloading can reduce power consumption up to roughly 90% and reduce execution delays by approximately by 98%.

2.4 Challenges and Future Directions

A wide variety of research challenges and opportunities exists for future research on computation offloading. However, the MEC research is still in its infancy, and many critical factors have been overlooked for simplicity. In this section, we point out several open challenges and shed light on possible future research directions.

- *Multi-server scheduling*: The collaboration of multiple MEC servers allows for their resources to be jointly managed in serving a large number of mobile devices simultaneously. Server cooperation not only can improve resource utilization but also can provide mobile users with more resources to enhance user experience. However, the increase in network size hinders practical MEC server scheduling. Too many offloading users will cause severe inter-user communication interference and the system will need to make large numbers of offloading decisions. More comprehensive research is required for multi-server scheduling.
- *Multi-resource optimization*: The architecture of mobile edge networks involves various resources: computing, caching, and communication resources. The efficient integration of these resources to achieve optimal performance for all users and applications is quite challenging. Efficient resource management requires the design of distributed low-complexity resource optimization algorithms, considering radio and computational resource constraints and computation overhead.
- *User mobility*: User mobility is a key challenge in mobile edge networks. Since the movement and trajectory of users provide location and personal preference information for edge servers, the contact times between users and MEC servers is dynamic, which will impact the offloading strategy. Moreover, the frequent mobility of users causes frequent handovers among edge servers, which will increase computation latency and thus deteriorate user experience. Therefore, mobility management techniques from both horizontal and vertical perspectives should be implemented to allow users seamless access to edge servers.
- *Security*: Security is one of the main concerns of technology advisers in securing MEC deployments. The deployment of edge cloud servers is creating novel security challenges due to the exploitation of mobile device information. The growing rate of the evolution of security solutions cannot keep up with the pace of new security challenges. Many existing security protocols assume full connectivity, which is not realistic in mobile edge networks, since many links are intermittent by default. On the other hand, in MEC, user data are offloaded to an MEC server

that gives access control to other mobile users. This introduces challenges, such as data integrity and authorization. For example, offloaded data can be modified or accessed by malicious users. Moreover, data owners and data servers possess dissimilar identities and business interests that make the scenario more vulnerable. Therefore, a comprehensive scientistic research study is required to avoid any security issues that can damage MEC systems.

This chapter first introduced the hierarchical mobile edge computing architecture with a cloud plane, an edge plane, and a user plane. Then, three types of computation models were discussed in detail for the typical computation offloading problem in MEC. In terms of the offloading decision, current research on computation offloading was surveyed, as were the binary offloading and partial offloading problems. Finally, several open challenges and future directions were discussed.