Chapter 4
Edge Computing for Digital Twin

Abstract Mobile edge computing is a promising solution for analysing and processing a portion of data using the computing, storage, and network resources distributed on the paths between data sources and a cloud computing centre. Mobile edge computing thus provides high efficiency, low latency, and privacy protection to sustain digital twin. In this chapter, we first introduce a hierarchical architecture of digital twin edge networks that consists of a virtual plane and a user/physical plane. We then introduce the key communication and computation technologies in the digital twin edge networks and present two typical cooperative computation modes. Moreover, we present the role of artificial intelligence (AI) for digital twin edge networks, and discuss the unique edge association problem.

4.1 Digital Twin Edge Networks

4.1.1 Digital Twin Edge Network Architecture

In traditional cloud computing–assisted digital twin modelling, the centralized server collects data and constructs twin mappings of the physical components, which leads to large communication loads. In this context, digital twin edge networks, a new paradigm that integrates mobile edge computing (MEC) and digital twin to build digital twin models at the network edge, has emerged as a crucial area. In digital twin edge networks, the edge nodes—for example, base stations (BSs) and access points—can collect running states of physical components and develop their behaviour model along with the dynamic environment. Furthermore, the edge nodes continuously interact with the physical components by monitoring their states, to maintain consistency with their twin mappings. Hence, the networking schemes (i.e. decision making, prediction, scheduling, etc.) can be directly designed and optimized in the constructed digital twin edge networks, which improves the efficiency of networking schemes and reduces costs. To better understand the internal logic

Y. Zhang, *Digital Twin*, Simula SpringerBriefs on Computing 16,
https://doi.org/10.1007/978-3-031-51819-5_4

of digital twin edge networks, we first present a hierarchical architecture of these networks that consists of a virtual plane and a user/physical plane, as shown in Fig. 4.1.

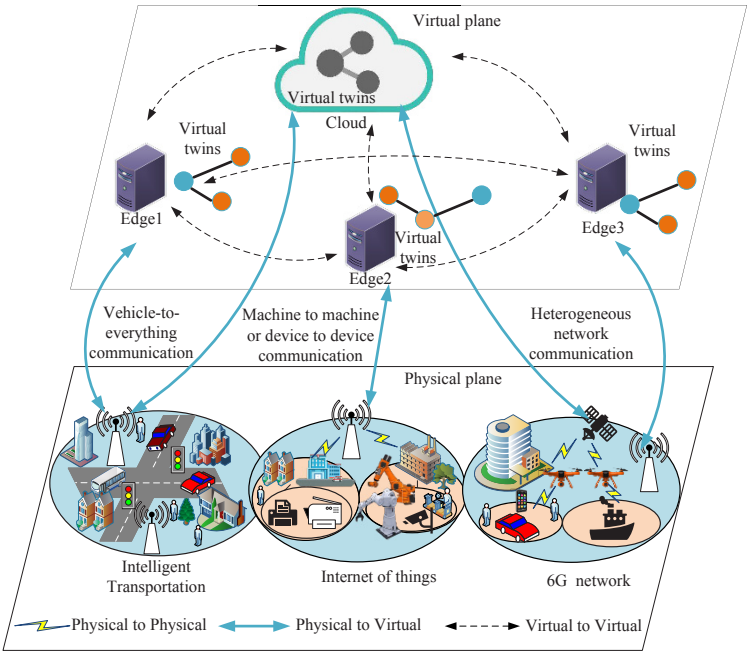

Fig. 4.1 A hierarchical architecture of digital twin edge networks

The user/physical plane is distinguished by typical digital twin application scenarios, such as an intelligent transportation system, the Industrial Internet of Things (IIoT), and sixth-generation (6G) networks. The virtual plane generates and maintains the virtual twins of physical objects by utilizing digital twin technology at the edge and on cloud servers. Specifically, devices in the user/physical plane include vehicles, sensors, smart terminals, and so forth. These devices need to synchronize their data with the corresponding virtual twins in real time through wireless communication technologies. Meanwhile, these devices also accept feedback from their virtual twins for instantaneous control and calibration. Therefore, mobile edge networks are expected to provide communications and computations that satisfy the main requirements of low latency, high reliability, high speed, and privacy and security preservation, to support real-time interactions between physical and virtual planes.

4.1.1.1 Communications

Communications between physical and virtual planes in typical digital twin edge network scenarios can be summarized as follows.

- *Intelligent transportation systems:* In recent years, urban transportation systems have faced such problems as traffic jams and traffic accidents. Digital twin edge networks can provide a virtual vision of the transportation system that can help to manage traffic and optimize public transportation service planning efficiency [46]. For example, traffic accidents can be effectively predicted and avoided by processing the massive amounts of real-time transportation information in the virtual plane [47]. Digital twin edge networks can also offer new opportunities for maintaining transportation facilities. By simulating the usage of transportation facilities in the virtual plane, facility malfunctions can be predicted in advance, which helps managers to schedule appropriate maintenance actions.
  Vehicle-to-everything communications allow vehicles to communicate with other vehicles and their virtual twins via wireless links, which can be realized by dedicated short-range communications and fifth-generation/6G communications [48]. In digital twin edge network–enabled intelligent transportation systems, vehicles' running states and perceived environmental information need to be transmitted to the virtual plane to update the virtual twins. However, it is challenging to guarantee strict data transmission delays, since vehicles move at high speeds. A detailed communications design must be carefully considered for physical plane and virtual plane interactions in such a dynamic network environment.
- *Internet of Things (IoT):* With the increasing scale of the IoT, digital twin is one of the most promising technologies enabling physical components be connected with their virtual twins in digital space by using different sensing, communication, computing, and software analytics technologies, to provide configuration, monitoring, diagnostics, and prognostics for maintaining physical systems [49, 50]. For example, in manufacturing, digital twin edge networks can be utilized for different aspects of manufacturing to improve production efficiency and reduce product life cycles [51, 52]. When designing parts, their full life cycle can be simulated through a virtual model, and design defects can be found in advance to realize accurate parts design. In factory production lines, through a virtual model of the entire production line, the production process can be simulated in advance and problems in the process found, to achieve more efficient production line management and process optimization. Additionally, in the health domain, digital twin edge networks can be utilized to establish twin patients. The twin patients can collect patients' physiological status and life style, medication input data, and data about the patients' emotional changes over time. Thus, twin patients can enable medical experts to provide patients with a full range of medical care and even accurately predict changes.
  Machine-to-machine and device-to-device (D2D) communications are enabling technologies for the digital twin edge network–empowered IoT [53]. Physical components can form clusters and transmit shared status data to the corresponding

virtual twins by reusing the unoccupied uplink spectrum resources. Machine-to-machine and D2D communications can improve data transmission rates during physical plane and virtual plane interactions. However, privacy and security protection of the information in virtual twin formation is a critical issue, since some core data, such as users' personal information, must be continuously updated for the virtual twins, and malicious attackers could intercept this information through wireless communications. Hence, privacy and security protection mechanisms need to be designed for physical and virtual plane interactions in the IoT.

- *6G networks:* 6G networks aim to realize ultra-high-capacity and ultra-short-distance communications, go beyond best effort and high-precision communications, and converge multiple types of communications [27]. Thus, 6G networks can face challenges in security, spectral efficiency, intelligence, energy efficiency, and affordability. The emergence of digital twin edge networks introduces opportunities to overcome these challenges. Digital twin edge networks provide corresponding virtual twins of 6G network components, which can collect traffic information on the entire network and use data analysis methods to discover network traffic patterns and detect abnormal traffic in advance. 6G networks use the information fed back from the virtual twins to make preparations in advance to improve network performance. In addition, by collecting and analysing the communication data in networks, rules of communication can be discovered to automate demand and provide services on demand. Since communication demand can be predicted in advance, the information can be fed back to the 6G networks to reserve resources, such as spectrum resources.

  The interactions between the physical and virtual planes in digital twin–empowered 6G networks demand high data rates. Small cell communication is an efficient solution for improving spectral efficiency by deploying heterogeneous infrastructures, such as pico and micro BSs [54]. In small cell communication, all BSs are equipped with rich computational resources and are responsible for generating and maintaining the virtual twins of physical objects in the cells. Additionally, intelligent communication infrastructures, such as reconfigurable intelligent surfaces [55] and unmanned aerial vehicles [56, 57], can be leveraged to realize interactions between the physical and virtual planes.

4.1.1.2 Computations for Resource-Intensive Tasks in the Virtual Plane

Beyond communications with low latency and high reliability, the resource-intensive tasks executed by digital twin edge networks require large amounts of computational resources. The virtual plane in the hierarchical architecture of digital twin edge networks consists of multiple distributed edge servers and central cloud servers. Specifically, central cloud servers have strong processing, caching, and computing capabilities. Resource-intensive tasks that focus on computation speed and centralized processing can be deployed on central cloud servers. Through cloud servers, large amounts of data can be processed in a short time (a few seconds), to provide

powerful digital twin services for physical objects. In addition, the cloud architecture facilitates the organization and management of large numbers of connected physical objects and virtual twins, as well as the combination and integration of real-time data and historical experience.

In addition, edge servers have computing (i.e. CPU cycles) and caching resources distributed on the paths between data sources and the cloud computing centre that can analyse and process a portion of the data from both physical objects and virtual twins. Edge servers can be deployed in the network infrastructure, such as at BSs, roadside units, wireless access points, gateways, and routers, or they can be mobile phones, vehicles, and other devices with the necessary processing power and computing and storage capabilities. Considering the proximity of edge servers to physical objects, delay-sensitive tasks can be deployed on edge servers to provide digital twin services for users with high efficiency.

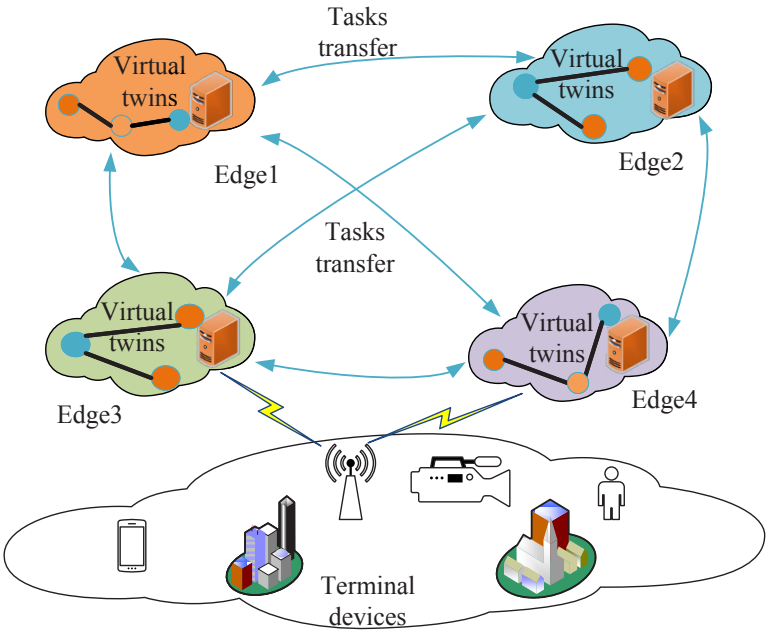

Fig. 4.2 Cooperative edge computing in digital twin edge networks

4.1.2 Computation Offloading in Digital Twin Edge Networks

In digital twin edge network scenarios, data processing and analysis require great amounts of computing resources. Nevertheless, most criteria cannot be met by edge computing, due to the limited capacity of edge servers. For example, when an edge

node has many computing tasks with a long task queue, it can easily create high latency. Cooperative computation is an approach for offloading computing tasks to other nodes that have free computing resources, to reduce task processing latency. According to different cooperation methods among the nodes, the following two cooperative computation modes can be used.

4.1.2.1 Cooperative Edge Computing

In cooperative edge computing, as shown in Fig. 4.2, if other edge nodes have free computing resources, they should share in the computing tasks of the overloaded edge nodes. It is very important for multiple edge nodes to maintain workload balance and provide low-latency computing services, particularly when a digital twin edge network provides services for time-sensitive scenarios, as in intelligence transportation systems.

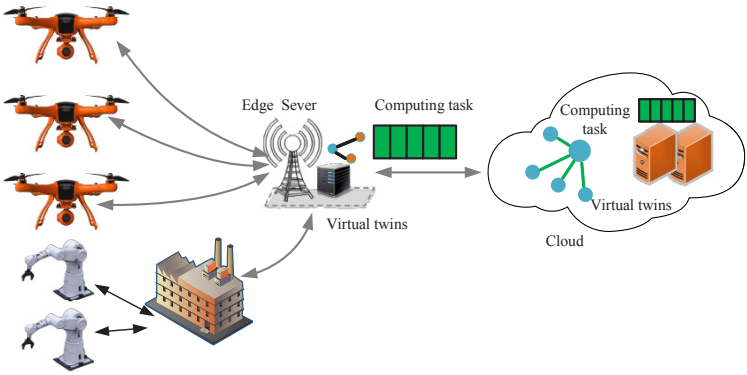

Fig. 4.3 Cooperative cloud-edge-end computing in digital twin edge networks

4.1.2.2 Cooperative Cloud-Edge-End Computing

As shown in Fig. 4.3, cooperative cloud-edge-end computing is necessary to meet the demand for large-scale computations and AI for real-time modelling and simulation in digital twin edge networks. Edge servers process the data that need to be responded to in real time. The cloud server provides strong computing power and the integration of various types of information. The interaction between edge nodes and the cloud in real time can solve the problem of data heterogeneity for the cloud. Cooperative cloud-edge-end computing can provide low-latency computation, communications, and virtual twin continuous updating for digital twin edge networks. In addition, when the storage resources of the edge nodes are insufficient, the cloud can store

part of the data and transmit them to the client through the network when needed, which saves storage resources on the edge servers.

In this section, we first present a hierarchical architecture of digital twin edge networks that consists of a virtual plane and a user/physical plane. Then, we illustrate key communications and computation technologies between the physical and virtual planes in the typical digital twin edge network scenarios and present two cooperative computation modes.

4.2 AI for Digital Twin Edge Networks

The integration of digital twin with AI [58] opens up new possibilities for efficient data processing in applying digital twins in 6G networks. MEC, one of the key enabling technologies for 6G, can considerably reduce system latency by executing computations based on AI algorithms at the edge of the network. AI-empowered MEC has been widely investigated for accomplishing edge intelligence tasks such as computation offloading, content caching, and data sharing. In [59], the authors proposed an AI-empowered MEC scheme in the IIoT framework. In [60], the authors proposed an intelligent content caching scheme based on deep reinforcement learning (DRL) for an edge computing framework. AI can significantly improve the construction efficiency and optimize the running performance of digital twin edge networks. The system model, communication model, and computation model of AI-empowered digital twin edge networks are as follows.

4.2.1 System Model

4.2.1.1 AI-Empowered Network Model

We consider the AI-empowered digital twin edge network shown in Fig. 4.4. Our wireless digital twin network system comprises three layers: a radio access layer (i.e. end layer), a digital twin layer (i.e. edge layer), and a cloud layer. The radio access layer consists of entities such as mobile devices and vehicles that have limited computing and storage resources. Through wireless communications, these entities connect to BSs and request services provided by network operators. In the digital twin layer, some BSs are equipped with MEC servers to execute computation tasks, while other BSs provide wireless communication services to end users. The digital twins of the physical entities are modelled and maintained by the MEC servers. Since the number of entities in the physical layer is much larger than the number of MEC servers in the digital twin layer, an MEC server can maintain multiple digital twins of physical entities. In the cloud layer, cloud servers are equipped with large amounts of computing and storage resources. Tasks that are computation sensitive or require global analysis can be executed in the cloud layer.

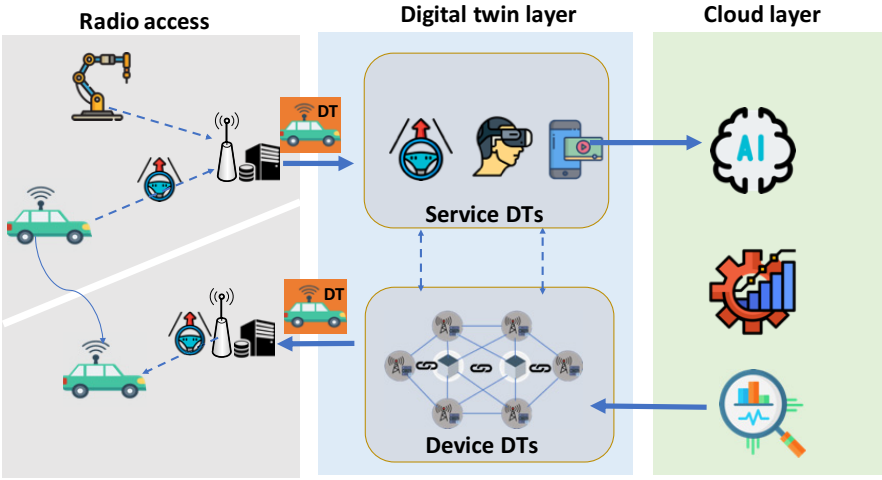

Fig. 4.4 The architecture of wireless digital twin networks

Since digital twins reproduce the running of physical entities, maintaining the digital twins of massive devices consumes a large number of resources, including computing resources, communication resources, and storage resources. To relieve the resource limitation in the edge layer, we model digital twins as one of two types: a device digital twin or a service digital twin. The device digital twin is a full replica of the physical devices, which includes the information of the hardware configuration, the historical running data, and real-time states. The device digital twin for user u_i can be expressed as

$$DT^f(u_i) = \Theta(\mathcal{D}_i, S_i(t), \mathcal{M}_i, \Delta S_i(t+1)), \tag{4.1}$$

where $\mathcal{D}_i$ is the historical data of user device i, such as the configuration data and historical running data. The term $S_i(t)$ represents the running state of device i, which consists of r_1 dimensions and varies with time, and it can be denoted as $S(t) = \{s_i^1(t), s_i^2(t), ..., s_i^{r_1}(t)\}$. The term $\mathcal{M}_i$ is the behaviour model set of u_i, which consists of r_2 behaviour dimensions, and $\mathcal{M}_i = \{m_i^1, m_i^2, ..., m_i^{r_2}\}$, and $\Delta S_i(t+1)$ is the state update of $S_i(t)$ in time slot $t+1$. Taking a meteorological IoT device as an example, $S(t)$ can be the temperature, humidity, wind speed, location, and so on. The behaviour models $\mathcal{M}_i$ can consist of the variation models of the temperature, humidity, and wind speed. In this paper, we mainly focus on the scenarios of device digital twin to conduct our study.

Different from a device digital twin, a service digital twin is a lightweight digital replica constructed by extracting the running states of several devices for a specific application. Similar to (4.1), the service digital twin can be expressed as

$$DT(u_i, \zeta) = \Theta(\mathcal{D}_i(\zeta), S_i^\zeta(t), \mathcal{M}_i^\zeta, \Delta S_i^\zeta(t+1)), \tag{4.2}$$

where ζ is the target service, and $\mathcal{D}_i(\zeta)$, $S_i^{\zeta}(t)$, $\mathcal{M}_i^{\zeta}$, and $\Delta S_i^{\zeta}(t+1)$ are the corresponding terms related to the target service ζ. For example, vehicles driving in the same region can be modelled into a specific service digital twin for supporting autonomous driving on a particular stretch of road. In such a case, the service digital twin for autonomous driving collects only the driving information of these vehicles and analyses their driving behaviour to guide them. Depending on the required scale, service digital twins can be constructed on the edge server or the cloud server.

4.2.2 Communication and Computation Model

The communication between end users and edge servers contains the uplink communication for transmitting data from user devices to edge servers and the downlink communication for sending the results from edge servers back to user devices. Note that the size of the results returning to users is much smaller than that of the updated data, so we consider only uplink communication latency in our communication model. The maximum achievable uplink data rate r_{ij} between user i and BS j is given as

$$r_{ij} = Wlog(1 + \frac{p_{ij}h_{ij}}{WN_0}), \tag{4.3}$$

where h_{ij} denotes the channel power gain of user i, p_{ij} denotes the corresponding transmission power for user i, N_0 is the noise power spectral density, and W is the channel bandwidth. The transmission latency for uploading D_i from user i to BS j can be expressed as

$$T_{ij}^{com} = \frac{D_i}{r_{ij}}. \tag{4.4}$$

The wired transmission latency between BSs is highly correlated to the transmission distance. Let ϕ be the latency required for transmitting one unit of data in each unit distance. Then the wired transmission latency can be written as

$$T_{j_1 j_2}^{com} = \phi \cdot D_j \cdot d(j_1, j_2), \tag{4.5}$$

where D_j is the size of the transmitted data and $d(j_1, j_2)$ is the distance between BSs j_1 and j_2.

We denote the total computation resource of edge server j as F_j. The computation resource of edge server j can be allocated to multiple user devices to maintain their digital twins on server j. Let f_{ij} denote the computation resource assigned to the digital twin of user i. Then the time to execute tasks from user i can be expressed as

$$T_{ij}^{cmp} = \frac{D_i}{f_{ij}}, \tag{4.6}$$

where D_i is the size of computation task from user i, $\sum_{i=1}^{N} x_{ij} f_{ij} \leq F_j$, and $x_{ij} = 1$ if $f_{ij} > 0$. Otherwise, $x_{ij} = 0$.

4.2.3 Latency Model

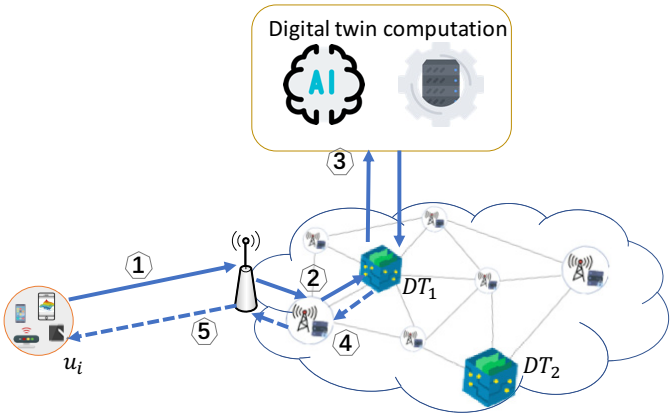

Fig. 4.5 The digital twin construction process

The latency of maintaining a digital twin mainly consists of two parts: the construction delay and the synchronization delay. Figure 4.5 shows the complete process for constructing a digital twin of user u_i. In the beginning, the running data D_i of u_i are transmitted to their nearby BS through wireless communication. Then the nearby BS transmits through wired communication the running data D_i to the digital twin server DT_1 that is responsible for constructing and maintaining the digital twin of u_i. The digital twin server DT_i runs the computation to process and analyse the received data and builds a digital twin model for user u_i, as expressed by Eq. (4.1). During the digital twin computation process, AI-related algorithms are used to extract the data features and to train the digital twin model. Finally, the results of the digital twin model are transmitted back to user u_i through wired and wireless communications. The feedback results provide u_i with insights for improving its service quality or running efficiency for specific applications. The system latency consists of the following items.

1. *Wireless data transmission:* In the construction phase of $DT(u_i)$, the historical running data of user i must be transmitted to its digital twin server through its nearby BS. Let D_i denote the size of the historical data to be transmitted. The wireless communication latency T_{ij}^{com} from user i to its BS j can then be calculated according to Eq. (4.4).
2. *Wired data transmission:* The wired transmission time from the nearby BS of u_i to its digital twin server k is

$$T_{jk}^{com} = \phi \cdot D_i \cdot d(j,k). \tag{4.7}$$

The total communication time for transmitting the historical data of u_i to its digital twin server is thus

$$T_{ik}^{com} = T_{ij}^{com} + T_{jk}^{com}. \tag{4.8}$$

3. *Digital twin data computation:* The computation time at digital twin server k is

$$T_{ik}^{cmp} = \frac{D_i}{f_{ij}}. \tag{4.9}$$

The total latency for constructing the digital twin of user i is

$$T_{ik}^{ini} = T_{ik}^{com} + T_{jk}^{com} + T_{ik}^{cmp}. \tag{4.10}$$

The digital twin of user i, that is, $DT(u_i)$, is constructed on its digital twin server DT_k. Then, $DT(u_i)$ must constantly interact with u_i to remain consistent with the running states of u_i. We denote the size of the updated data as ΔD_i. The latency for one update can then be expressed as

$$T_{ik}^{upd} = \frac{\Delta D_i}{r_{ij}} + \phi \cdot \Delta D_i \cdot d(j,k) + \frac{\Delta D_i}{f_{ij}}. \tag{4.11}$$

The synchronization latency in one unit time slot can be written as

$$T_{ik}^{syn} = \frac{1}{\Delta t} T_{ik}^{upd}, \tag{4.12}$$

where Δt denotes the time gap between every two updates.

4.3 Edge Association for Digital Twin Edge Networks

4.3.1 System Model

Due to the dynamic computing and communication resources available through edge servers, the association of digital twins to corresponding servers is a fundamental problem in digital twin edge networks that needs to be comprehensively explored. Moreover, since the federated learning in digital twin edge networks requires multiple communications for data exchange, the limited communication resources need to be optimally allocated to improve the efficiency of digital twins in the associated edge servers. Thus, in this section, we design a digital twin wireless network (DTWN) model and define the edge association problem for digital twin networks. A permissioned blockchain-empowered federated learning framework for edge association is also proposed.

We consider a blockchain- and federated learning–empowered digital twin network model as depicted in Fig. 4.6. The system consists of N end users, such as

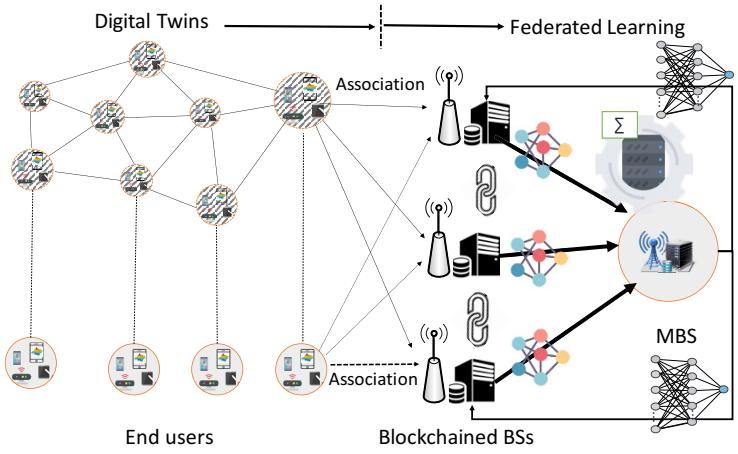

Fig. 4.6 The proposed digital twin wireless network

IoT devices and mobile devices, M BSs, and a macro BS (MBS). The BSs and the MBS are equipped with MEC servers. The end devices generate running data and synchronize their data with the corresponding digital twins that run on the BSs. We use $\mathcal{D}_i = \{(x_{i1}, y_{i1}), ..., (x_{iD_i}, y_{iD_i})\}$ to denote the data of end user i, where D_i is the data size, x_i is the data collected by end users, and y_i is the label of x_i. The digital twin of end user i in the BSs are denoted as DT_i, which is composed of the behaviour model $\mathcal{M}_i$, static running data $\mathcal{D}_i$, and the real-time dynamic state s_t, so that $DT_i = (\mathcal{M}_i, \mathcal{D}_i, s_t)$, where$\mathcal{D}_i$ and s_t are the essential data required to run the digital twin applications. Instead of synchronizing all the raw data to the digital twins, which incurs a huge communication load and the risk of data leakage, we use federated learning to learn model $\mathcal{M}$ from the user data. In various application scenarios, the end users can communicate with other end users to exchange running information and share data, through, for example, D2D communications. Thus, the digital twins also form a network based on the connections of end users. Based on the constructed DTWN, we can obtain the running states of the physical devices and make further decisions to optimize and drive the running of the devices by directly analysing the digital twins.

In our proposed digital twin network model, we use federated learning to execute the training and learning process collaboratively for edge intelligence. Moreover, since the end users lack mutual trust and the digital twins consist of private data, we use permissioned blockchain to enhance the system security and data privacy. The permissioned blockchain records the data from digital twins and manages the participating users through permission control. The blockchain is maintained by the BSs, which are also the clients of the federated learning model. The MBS runs as the server for the federated learning model. In each iteration of federated learning, the MBS distributes the machine learning model parameters to the BSs for training.

The BSs train the model based on the data from the digital twins and returns the model parameters to the MBS.

We use orthogonal frequency division multiple access for wireless transmission in our system. To upload trained local models, all the BSs share C subchannels to transmit their parameters. The achievable uplink data rate from BS i to the MBS is

$$R_i^U = \sum_{c=1}^{C} \tau_{i,c} W^U log_2(1 + \frac{P_{i,c}^U h_{i,c}^U r_{i,m}^{-\alpha}}{\sum_{j \in \mathcal{N}'} P_{j,c}^U h_{j,c}^U r_{i,m}^{-\alpha} + N_0}), \tag{4.13}$$

where C is the total number of subchannels, $\tau_{i,c}$ is the time fraction allocated to BS i on subchannel c, and W^U is the bandwidth of each subchannel, which is a constant value. The transmission power is $P_{i,c}^U$ and the uplink channel gain on subchannel c is $h_{i,c}^U$; $r_{i,m}^{-\alpha}$ is the path loss fading of the channel between BS i and the MBS; $r_{i,m}$ is the distance between BS i and the MBS; α is the path loss exponent; N_0 is the noise power; and $\sum_{j \in \mathcal{N}'} P_{j,c}^U h_{j,c}^U r_{i,m}^{-\alpha}$ is the interference caused by other BSs using the same subchannel. In the download phase, the MBS broadcasts the global model with the rate

$$R_i^D = \sum_{c=1}^{C} W^D log_2(1 + \frac{P_{i,c}^D h_{i,c}^D}{\sum_{j \in \mathcal{N}''} P_{j,c}^D h_{j,c}^D r_{i,m}^{-\alpha} + N_0}), \tag{4.14}$$

where $P_{i,c}^D$ is the downlink power of BS i, $h_{i,c}^D$ is the channel gain between BS i and the MBS, and $\sum_{j \in \mathcal{N}''} P_{j,c}^D h_{j,c}^D r_{i,m}^{-\alpha}$ is the downlink inference.

4.3.2 Edge Association: Definition and Problem Formulation

The end devices or users are mapped to the digital twins in the BSs in the DTWN. The maintenance of digital twins consumes a large amount of computing and communication resources for synchronizing real-time data and building corresponding models. However, the computation and communication resources in wireless networks are very limited and should be optimally used to improve resource utility. Thus, the association of various IoT devices with different BSs according to their computation capabilities and states of the communication channel is a key problem in DTWNs. As depicted in Fig. 4.6, the digital twins of IoT devices are constructed and maintained by their associated BSs. The training data and the computation tasks for training are distributed to various BSs based on the association between the digital twins and the

Definition (Edge Association) Consider a DTWN with N IoT users and M BSs. For any user u_i, $i \in \mathcal{N}$, the goal of edge association is to choose the target BS $j \in \mathcal{M}$ to construct the digital twin DT_i of user i. The association $\langle DT_i, BS_j \rangle$ is denoted as $\Phi(i, j)$. If DT_i is associated with BS j, then $\Phi(i, j) = D_i$, where D_i is the size of the data used to construct DT_i. Otherwise, $\Phi(i, j) = 0$. □

A BS can be associated with multiple digital twins, whereas a digital twin can only be associated with at most one BS; that is, $\sum_{j=1}^{M} \Phi(i,j) = D_i$. We perform edge association according to the datasets D_i of IoT users, the computation capability of the BSs f_j, and the transmission rate $R_{i,j}$ between u_i and BS_j, denoted as

$$\Phi(i,j) = f(D_i, f_j, R_{i,j}). \tag{4.15}$$

The objective of the edge association problem is to improve the utility of resources and the efficiency of running digital twins in the DTWN.

We use the weight matrix $A = [a_{ik}]$ to represent the association relations between the user devices and the digital twin servers, where $a_{ik} = 1$ if the digital twin of user i is maintained by digital twin server k. Otherwise, $a_{ik} = 0$. For example, in Fig. 4.5, since the digital twin of u_i is maintained by DT_1, we have $a_{i1} = 1$ and $a_{i2} = 0$. The association matrix takes the form

$$\begin{bmatrix} a_{11} & a_{12} & a_{1k} & \dots & a_{1M} \\ a_{21} & a_{22} & a_{2k} & \dots & a_{2M} \\ a_{21} & a_{22} & a_{2k} & \dots & a_{2M} \\ . & . & . & \dots & . \\ . & . & . & \dots & . \\ . & . & . & \dots & . \\ a_{N1} & a_{N2} & a_{Nk} & \dots & a_{NM} \end{bmatrix}.$$

Now we start to derive the formulation of the edge association problem. We consider that the gradient $\nabla f(w)$ of $f(w)$ is L-Lipschitz smooth; that is,

$$||\nabla f(w_{t+1}) - \nabla f(w_t)|| \leq L||w_{t+1} - w_t||, \tag{4.16}$$

where L is a positive constant and $||w_{t+1} - w_t||$ is the norm of $w_{t+1} - w_t$. We consider that the loss function $f(w)$ is strongly convex; that is,

$$f(w_{t+1}) \geq f(w_t) + \langle \nabla f(w_t), w_{t+1} - w_t \rangle + \frac{1}{2}||w_{t+1} - w_t||^2. \tag{4.17}$$

Many loss functions for federated learning can satisfy the above assumptions, for example, logic loss functions. If (4.16) and (4.17) are satisfied, the upper bound of the global iterations can be obtained as

$$\mathcal{T}(\theta_L, \theta_G) = \frac{O(log(1/\theta_L))}{1 - \theta_G}, \tag{4.18}$$

where θ_L is the local accuracy $\frac{||\nabla f(w_{t+1})||}{||\nabla f(w_t)||} \leq \theta_L$, θ_G is the global accuracy, and $0 \leq \theta_L, \theta_G \leq 1$. As in [94], we consider θ_L a fixed value, so that the upper bound $\mathcal{T}(\theta_L, \theta_G)$ can be simplified to $\mathcal{T}(\theta_G) = \frac{1}{1-\theta_G}$. If we denote the time of one local training iteration by T_{cmp}, then the computation time in one global iteration is $log(1/\theta)T_{cmp}$, and the upper bound of total learning time is $\mathcal{T}(\theta_G)T_{glob}$.

The time cost in our proposed scheme mainly consists of the following.

1. *Local training on digital twins:* The time cost for the local training of BS i is determined by the computing capability and the data size of its digital twins. The time cost is
$$T_i^{cmp} = \frac{\sum_{j=1}^{K_i} b_j D_{DT_j}}{f_i^C} f^C, \tag{4.19}$$
where f^C is the number of CPU cycles required to train one sample of data, f_i^C is the CPU frequency of BS i, and b_j is the training batch size of digital twin DT_j.
2. *Model aggregation on the BSs:* The BSs aggregate their local models from various digital twins. The computing time for local aggregation is
$$T_i^{la} = \frac{\sum_{j=1}^{K_i} |w_j|}{f_i^C} f_b^C, \tag{4.20}$$
where $|w_j|$ is the size of the local models and f_b^C is the number of CPU cycles required to aggregate one unit of data. Since all the clients share the same global model, $|w_1| = |w_2| = ... = |w_j| = |w_g|$. Thus the time cost for local aggregation is
$$T_i^{la} = \frac{K_i |w_g|}{f_i^C} f_b^C. \tag{4.21}$$
3. *Transmission of the model parameters:* The local models aggregated by BS i are then broadcast to other BSs as transactions. The time cost is related to the number of blockchain nodes and the transmission efficiency. Since other BSs also help to transmit the transaction in the broadcast process, the time function is related to $log_2 M$, where M is the size of the BS network. The required time cost is
$$T_i^{pt} = \xi log_2 M \frac{K_i |w_g|}{R_i^U}, \tag{4.22}$$
where ξ is a factor of the transmission time cost that can be obtained from the historical running records of the transmission process.
4. *Block validation:* The block producer BS collects the transactions and packs them into a block. The block is then broadcast to other producer BSs and validated by them. Thus, the time cost is
$$T_{bp}^{bv} = \xi log_2 M_p \frac{S_B}{R_i^D} + \max_i \frac{S_B f^v}{f_i s}, \tag{4.23}$$
where M_p is the number of block producers and S_B is the size of a block.

Note that, in the aggregation phase, the size of the model parameters $|w_g|$ is small and the computing capability f_i is high. Thus, compared to other phases, the time for aggregation is very short, such that it can be neglected. Based on the above analysis, the time cost for one iteration is denoted as

$$T = \max_i \left\{ \frac{\sum_{j=1}^{K_i} b_j D_{DT_j}}{f_i^C} f^C \right\} + \max_i \left\{ \xi log_2 M \frac{K_i |w_g|}{R_i^U} \right\} + \xi log_2 M_p \frac{S_B}{R_i^D} + \max_i \frac{S_B f^v}{f_i s}. \tag{4.24}$$

In the 6G network, the growth of the user scale, the ultra-low latency requirement of communication, and the dynamic network status make the reduction of the time cost of model training an important issue in various applications. Since accuracy and latency are the two main metrics for evaluating the decision-making abilities of digital twins in our proposed scheme, we consider the edge association problem to find the trade-off between learning accuracy and the time cost of the learning process. Due to the dynamic computing and communication capabilities of various BSs, the edge association of digital twins—that is, how to allocate the digital twins of different end users to various BSs for training—is a key issue to be solved to minimize the total time cost. Moreover, increasing the training batchsize b_n of each digital twin DT_n can improve the learning accuracy. However, this will also increase the learning time cost to execute more computations. In addition, how to allocate the bandwidth resources to improve communication efficiency should be considered. In our edge association problem, we should carefully design these policies to minimize the total time cost of the proposed scheme. Thus, we formulate the optimization problem as the minimization of the time cost of federated learning for a given learning accuracy. To solve the problem, the association of digital twins, the batchsize of their training data, and the bandwidth allocation should be jointly considered according to the computing capability f_i^C and the channel state $h_{i,c}$. The optimization problem can be formulated as

$$\min_{\boldsymbol{K_i}, \boldsymbol{b_n}, \boldsymbol{\tau_{i,c}}} \frac{1}{1-\theta_G} T \tag{4.25}$$

$$\text{s.t.} \quad \theta_G \geq \theta_{th}, \theta_G, \theta_{th} \in (0,1), \tag{4.25a}$$

$$\sum_{i=1}^{M} K_i = D, K_i \in \mathcal{N}, \tag{4.25b}$$

$$\sum_{i=1}^{M} \tau_{i,c} \leq 1, c \in C, \tag{4.25c}$$

$$b^{min} \leq b_n \leq b_n^{max}, \forall n \in \mathcal{N}. \tag{4.25d}$$

Constraint (4.25b) ensures that the sum of the number of associated digital twins does not exceed the size of the total dataset. Constraint (4.25c) guarantees that each subchannel can only be allocated to at most one BS. Constraint (4.25d) ensures the range of the training batchsize for each digital twin. Problem (4.25) is a combinational problem. Since there are several products of variables in the objective function and the time cost of each BS is also affected by the resource states of other BSs, problem (4.25) is challenging to solve.

4.3.3 Multi-Agent DRL for Edge Association

Since the system states are only determined by network states in the current iteration and the allocation policies in the last iteration, we regard the problem as a Markov decision process and use a multi-agent DRL-based algorithm to solve it.

The proposed multi-agent reinforcement learning framework is depicted in Fig. 4.7. In our proposed system, each BS is regarded as a DRL agent. The environment consists of BSs and the digital twins of the end users. Our multi-agent DRL framework consists of multiple agents, a common environment, the system state $\mathcal{S}$, the action $\mathcal{A}$, and the reward function $\mathcal{R}$, which are described below.

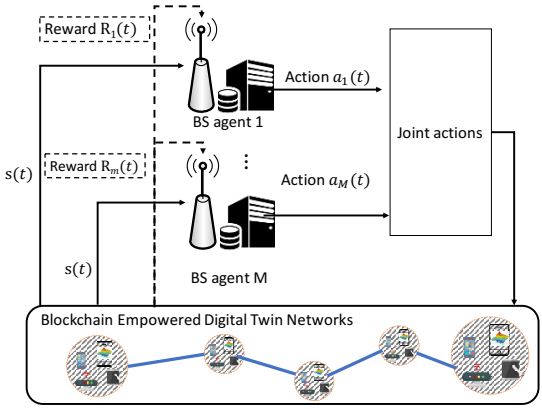

Fig. 4.7 Multi-agent DRL for edge association

- *State space:* The state of the environment is composed of the computing capabilities f^C of the BSs, the number of digital twins K_i on each BS i, the training data size of each digital twin D_n, and the channel state $h_{i,c}$. The states the of multiple agents are denoted as $s(t) = (\boldsymbol{f}^C, \boldsymbol{K}, \boldsymbol{D}, \boldsymbol{h})$, where each dimension is a state vector that contains the states for all the agents.
- *Action space:* The actions of BS i in our system consist of the digital twin allocation K_i, the training data batchsizes for its digital twins $\boldsymbol{b}_i$, and the bandwidth allocation $\boldsymbol{\tau}_i$. Thus, the actions are denoted as $\boldsymbol{a}_i(t) = (K_i, \boldsymbol{b}_i, \boldsymbol{\tau}_i)$. BS agent i makes new action decisions $\boldsymbol{a}_i(t)$ at the beginning of iteration t based on system state $s(t)$. The system action is $\boldsymbol{a}(t) = (\boldsymbol{a}_1, ..., \boldsymbol{a}_i, ..., \boldsymbol{a}_m)$.
- *Reward:* We define the reward function of BS i according to its time cost T_i based on Eq. (4.24):

$$\mathcal{R}_i(s(t), \boldsymbol{a}_i(t)) = -T_i(t). \tag{4.26}$$

  The reward vector of all the agents is $\boldsymbol{R} = (\mathcal{R}_1, ..., \mathcal{R}_m)$. According to Eq. (4.25), the total time cost T is decided by the maximum time cost of the agents $\max\{T_1, T_2, ..., T_m\}$. Each DRL agent in our scheme thus shares the same reward

function. In the training process, the BS agents adjust their actions to maximize the reward function, that is, to minimize the system time cost in each iteration.

The learning process of BS i is to find the best policy that maps its states to its actions, denoted as $\boldsymbol{a}_i = \pi_i(s)$, where $\boldsymbol{a}_i$ is the action to be taken by BS i for the whole system state s. The objective is to maximize the expected reward, that is,

$$\mathcal{R}_t = \sum_i \gamma \mathcal{R}_i(s(t), \boldsymbol{a}_i(t)), \tag{4.27}$$

where γ is the discount rate, $0 \leq \gamma \leq 1$. In the conventional DRL framework, it is hard for an agent to obtain the states of others. In our DTWN, the states of the digital twins and BSs are recorded in the blockchain. A BS can retrieve records from the blockchain to obtain the system states and actions of other agents in the training process. We use $\pi = [\pi_1, \pi_2, ..., \pi_n]$ to denote the policies of the n agents, whose parameters are denoted as $\theta = [\theta_1, \theta_2, ..., \theta_n]$. Thus we have the following policy gradient for agent i:

$$\begin{aligned} \nabla_{\theta_i} J(\pi_i) = E_{\{\theta_i\}, a\sim D}[\nabla_{\theta_i} \pi(a_i|o_i) \cdot \\ \nabla_{a_i} Q_i^{\pi}(\{\theta_i\}, a_1, ..., a_n)|_{a_i=\pi_i(o_i)}], \end{aligned} \tag{4.28}$$

where $\{\theta_i\}$ is the observation of agent i, that is, the state of each agent. In our scheme, since the placement of digital twins requires global coordination, we consider that all the agents share the same system state through information exchange between the servers. Agent i determines its action $\boldsymbol{a}_i$ through its actor deep neural network (DNN) $\pi(s_t|\theta_\pi)$, denoted as

$$a_i(t) = \pi_i(s_t|\theta_{\pi_i}) + \mathfrak{N}, \tag{4.29}$$

where $\mathfrak{N}$ is the random noise for generating a new action. The actor DNN is trained as

$$\theta_\pi = \theta_\pi + \alpha_\pi \cdot \mathbb{E}[\nabla_{a_i} Q(\boldsymbol{s_t}, \boldsymbol{a}_1, ..., \boldsymbol{a}_i|\theta_Q)|_{\boldsymbol{a_i}=\pi(\boldsymbol{s_t}|\theta_\pi)} \cdot \nabla_{\theta_\pi} \pi(\boldsymbol{s_t})], \tag{4.30}$$

where α_π is the learning rate of the actor DNN.

The critic DNN of agent i is trained as

$$\theta_{Q_i} = \theta_{Q_i} + \alpha_{Q_i} \cdot \mathbb{E}[2(y_t - Q(\boldsymbol{s_t}, \boldsymbol{a_i}|\theta_{Q_i})) \cdot \nabla Q(\boldsymbol{s}_t, \boldsymbol{a}_1, ..., \boldsymbol{a}_i)], \tag{4.31}$$

where α_{Q_i} is the learning rate, y_t is the target value, and $(\boldsymbol{a}_1, ..., \boldsymbol{a}_i)$ constitutes the actions of the agents in our system.

In the proposed algorithm, all the actor networks and critic networks are initialized randomly as the initial training parameters. Then the replay memory is initialized to store the experiential samples in the training process. In each episode, the agent selects its action towards its current observation state and obtains the reward for its current action. Then the new observation of the system state is obtained. The experience tuple (s_t, a_i, r_t, s_{t+1}) is then stored in the replay buffer. Finally, the

agents train their critic network and actor network by sampling records from the replay buffer.