Chapter 6
Digital Twin for 6G Networks

Abstract Digital twin is a technology that has the potential to help sixth-generation (6G) networks to realize digitization. In this chapter, we first introduce the combination of digital twin and 6G and then discuss two key use cases in terms of reconfigurable intelligent surfaces and digital twin and digital twins for stochastic offloading.

6.1 Integration of Digital Twin and Sixth-Generation (6G) Networks

To meet the ever-increasing demands of user traffic, fifth-generation (5G) networks integrate several novel network architectures, such as edge computing, software-defined networking, network function virtualization, and ultra-dense heterogeneous networks, to realize performance improvements for peak rates, transmission latency, network energy efficiency, and other indicators. However, the rapid proliferation and breakneck expansion of 5G wireless services also pose new challenges on transmission data rates, ubiquitous coverage, reliability, and network intelligence [67]. These challenges are spurring activities focused on defining the next-generation 6G wireless networks. Compared with 5G, 6G networks are envisioned to achieve the superior performance in the following areas [68, 69].

- *Peak data rate*: The peak data rate is the highest data rate under ideal channel conditions where all available radio resources are completely assigned to a single mobile device. Driven by both user demand and technological advances such as terahertz communications, peak data rates are expected to reach up to 1 Tbps, 10 times that of 5G.
- *Latency*: Latency can be distinguished as the user plane and control plane latency. The minimum latency requirement for the user plane is 1–4 ms. This value is envisioned to be further reduced in 6G to 100 μs or even 10 μs. The minimum

© The Author(s) 2024

Y. Zhang, *Digital Twin*, Simula SpringerBriefs on Computing 16,
https://doi.org/10.1007/978-3-031-51819-5_6

latency for the control plane should be 10 ms in 5G and is also expected to be remarkably improved in 6G.
- *Mobility*: The highest mobility supported by 5G is 500 km/h. In 6G, the maximal speed of 1,000 km/h is targeted to meet the requirements of commercial airline systems.
- *Connection density*: The minimum number of devices with a relaxed quality of service in 5G is $10^6/km^2$. In 6G, the connection density is envisioned to be further improved by 10 times, to $10^7/km^2$.
- *Energy efficiency*: Energy efficiency is an important metric to enable cost-efficient wireless networks for green communications. In 6G, network energy efficiency is expected to increase 10 to 100 times compared to that in 5G.
- *Signal bandwidth*: The requirement for bandwidth in 5G is at least 100 MHz, and 6G will support up to 1 GHz for operations in higher frequency bands, and even higher in terahertz communications.

Beyond imposing new performance metrics, emerging trends that include new services and the recent revolutions in artificial intelligence (AI), computing, and sensing will redefine 6G. Digital twin, as one of the emerging technologies for next-generation network digitalization, can pave the way for the creation of future digital 6G by transforming and precisely mapping physical networks to digital networks with virtual twins. Digital twin will provide three main benefits for 6G. First, digital twin can provide a comprehensive and accurate network analysis for 6G with increasingly accurate and synchronous network updates. Second, digital twin can build a virtual twin layer between the physical entities and user applications. This can establish a bridge between the bottom network and the top application with better cross-layer interaction and timely user experience feedback. Third, digital twin–enabled 6G can utilize AI algorithms to adjust network schedules, such as task offloading, resource allocation, and network management. Thus, digital twin is an essential technique for 6G in terms of supporting network automation and intelligence.

6.2 Potential Use Cases

Several works have explored utilizing digital twins to enhance the performance of next-generation communication networks. In [70], the authors proposed digital twin–enabled 6G to enable network scalability and reliability. The authors in [71] analysed the potential of digital twin for next-generation communication networks in terms of radio access, channel emulation, and network optimization. These works discussed how digital twin could be a powerful tool to fulfil the potential of 6G. Next, we present three detailed use cases of the combination of digital twin and 6G.

- *Reconfigurable intelligent surface (RIS) technology and digital twin*: With the dense deployment of edge servers, there will be increasing data transmission requirements in the next-generation networks, which will aggravate network interference and increase transmission delays. Current massive multiple input, multiple

output and millimetre wave technologies can increase wireless communication data rates, but these can incur high hardware costs and complicated signal processing issues. RIS is a new technology for 6G that can enhance spectral efficiency and suppress interference in wireless communications by adaptively configuring massive low-cost passive reflecting elements. However, to improve wireless transmission rates, RIS requires both the amplitude and phase of passive reflecting elements to be adjusted to facilitate an enhanced signal propagation environment. Since virtual twins can record the real-time states of physical objects, monitor the dynamic changes of wireless networks, and carry out optimization and prediction to improve the performance of the physical system, RIS can utilize digital twin to extract the key features of RIS components, such as the number of RIS elements, the phase and amplitude of the reflecting elements, and the mobile devices served by each RIS element. With the extracted information, digital twin can assist in RIS to adjust the wireless propagation environment to improve the signal-to-noise ratio and decrease the probability of outages.

- *Edge association and digital twin*: The huge number of connected devices and the heterogeneous network structure of 6G pose great challenges for constructing digital twins in each network's infrastructure. A possible solution for this issue is to select a subset of base stations as the digital twin servers to maintain the digital twins at reduced time cost and energy consumption, instead of maintaining digital twins at every base station (BS). To achieve this, the edge association problem must be addressed. The objective of edge association is to minimize the average system latency while providing delay-guaranteed service for each user. According to the running phases of digital twins, edge association consists of two subproblems: the digital twin placement problem and the digital twin migration problem. The digital twin placement problem involves how to choose the optimized subset of BSs as digital twin servers. The migration of digital twins problem involves how to allocate network resources to ensure relatively low transmission overhead and communication latency in the process of digital twin migration.
- *Cellular vehicle to everything (C-V2X) and digital twin*: The rapid development of wireless communications and C-V2X has facilitated the wide use of smart vehicles and enriched many intelligent transportation system applications, such as smart navigation, road condition recognition, high-precision real-time mapping, forward collision warning, and driving assistance. However, due to the high mobility of vehicles, it is difficult to test C-V2X functionalities and performance for typical V2X use cases. Digital twin can provide a high-fidelity digital mirror of C-V2X systems throughout their entire life cycle [72]. By using digital twin mapping, the predicted state of automatic driving vehicles can be realized based on a virtual simulation test environment. Based on the prediction information of digital twin, driving behaviours and emergency events can be more actually determined and quickly perceived.

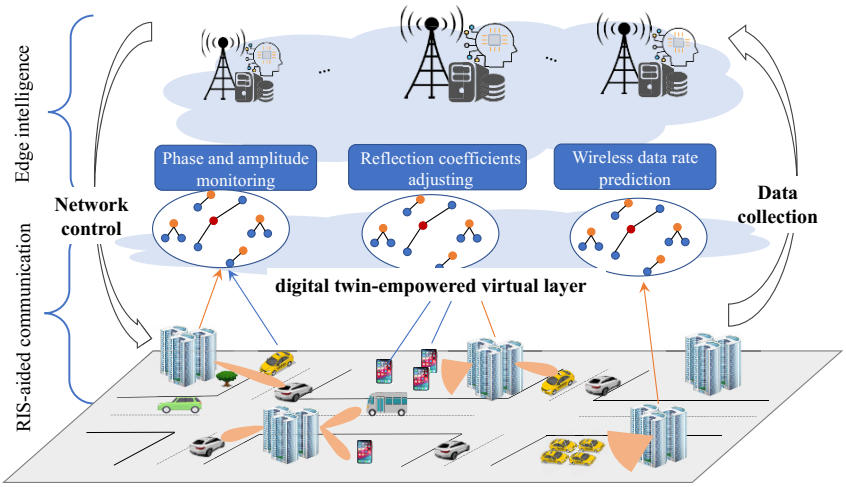

Fig. 6.1 Digital twin–empowered RIS framework

6.3 Digital Twin for RIS

To support emerging applications, 6G networks deploy computation/storage capabilities at BSs to avoid long transmission delays from mobile devices to cloud servers. However, while this shortens the distance and delay to access cloud server resources, it does not improve the wireless propagation environment. The recently proposed RIS technology can enhance spectral efficiency and suppress interference by adjusting both the amplitude and phase of passive reflecting elements. Digital twin can assist in RIS to intelligently adjust passive reflecting elements.

6.3.1 System Model

To clearly illustrate the combination of digital twin and RIS, we present a hierarchical digital twin–empowered RIS framework, as shown in Fig. 6.1. In this framework, edge resources can alleviate the heavy computational pressure of mobile devices, and edge servers can reduce task processing latency due to their proximity to mobile devices. RIS can enhance the quality of wireless communication links in the process of task offloading by intelligently altering the radio propagation environment.

The proposed framework consists of two layers: an RIS-aided communication layer and a digital twin–empowered virtual layer. In the RIS-aided communication layer, RIS elements are distributively installed on the surface of building facades, to improve propagation conditions and increase the quality of wireless communications. The digital twin–empowered virtual layer is constructed by diverse distributed edge servers. With edge resources and AI algorithms, virtual twins can construct a real-

time mirror of the physical network to enable intelligent policy design, quality of service requirements, resource management, and network topology monitoring. This is a general framework that can improve the communication and computational performance in many scenarios, including cellular, vehicular, and unmanned aerial vehicle networks.

6.3.2 Computation Offloading in Digital Twin–Aided RIS

To elaborate on how digital twin assists in RIS coefficient adjustment, in this section, we present a case study that focuses on RIS-aided offloading. We consider a network of digital twin–aided RIS offloading that consists of a physical network entities layer and a digital twin–empowered virtual layer. The physical network entities layer contains three types of physical entities: base stations, RISs, and mobile devices. Since digital twin mirrors a physical entity, the digital twin–empowered virtual layer also contains three types of virtual models. The first type of virtual model involves the BSs. We consider that each physical BS has multiple antennas and an edge server for providing edge computing via wireless communications. The virtual model of a BS with edge intelligence and can thus predict current available communication, computing, and caching resources and monitor current wireless links to construct the current network topology. The second type of virtual model involves RIS, including the number of RIS elements and the phase and amplitude of reflecting elements. The key function of this virtual model is to adjust the RIS coefficients. The third type of virtual model involves mobile devices. This type of virtual model mainly records the size of the collected data, the current locations of the mobile devices, and the latency or computational resource requirements of on-device applications.

Task offloading aims to offload the computation-intensive tasks of mobile devices to nearby distributed BSs for processing. The virtual model of each mobile device records the computation-intensive task as (d_k, c_k), where d_k is the data size of task k and c_k is the required computation resource for the computing unit bit. The virtual model needs to determine what part of the task should be processed locally and how much should be offloaded to the edge server to process. We define this as the offloading ratio (i.e. x_k). RIS offloading utilizes RIS to assist in task offloading for a higher wireless communication rate. Different from traditional wireless transmission links, which only include direct device–BS links, the wireless transmission link in RIS offloading includes both of device–BS links and reflected device–RIS–BS links. For the device–BS link, the virtual model of the BS records its channel vector, that is, $\mathbf{h}_k^d$. The reflected device–RIS–BS link contains three components: the device–RIS link, the RIS reflection with phase shifts, and the RIS–BS link. The virtual RIS model records the channel vectors of the device–RIS link and RIS–BS link as $\mathbf{h}_k^r$ and $\mathbf{h}^H$, respectively. The RIS reflection coefficients are denoted as $\Theta = \text{diag}(\beta_1 e^{j\theta_1}, \beta_2 e^{j\theta_2}, ..., \beta_N e^{j\theta_N})$, where β_n and θ_n are the amplitude and phase shift of the nth RIS element, respectively. The effective channel gain can be expressed as

$$\mathbf{g}_k = \mathbf{h}_k^d + \mathbf{h}^H \Theta \mathbf{h}_k^r. \tag{6.1}$$

Based on channel gain, the maximum achievable wireless transmission data rate can be obtained by

$$R_k = B \log_2(1 + \frac{p_k |\mathbf{w}_k^H \mathbf{g}_k|^2}{\sum_{j=1, j\neq i}^{K} p_j |\mathbf{w}_i^H \mathbf{g}_k|^2 + \sigma^2}), \tag{6.2}$$

where B is the system's bandwidth. The virtual RIS model should properly adjust the reflection coefficients to improve the wireless communication rate.

The task execution latency is determined by the local computation and task offloading. The latency of local computation is mainly related to the computational capability of each mobile device (i.e. f_k^l). The latency of task offloading involves the task transmission time and edge computation time. Since the two parts are executed in parallel, the total task execution latency is equal to the maximal value of the two processes. To minimize the total task execution latency, the RIS configuration offloading ratio and computation resource must be jointly optimized. The RIS offloading problem can be formulated as

$$\begin{aligned}
\min_{x,f,\beta,\theta} \quad & \sum_{k\in\mathcal{K}} \max\{d_k c_k \frac{1-x_k}{f_k^l}, d_k c_k \frac{x_k}{f_k^s} + d_k \frac{x_k}{R_k(\mathbf{g}_k)}\} \\
\text{s.t.} \quad & \sum_{k\in\mathcal{K}} f_k^s \leqslant F^s,\ 0 \leqslant f_k^s \leqslant F^s, k \in \mathcal{K}, && \text{(6.3a)} \\
& x_k, \beta_n \in [0,1], k \in \mathcal{K}, n \in \mathcal{N}, && \text{(6.3b)} \\
& 0 \leqslant \theta_n \leqslant 2\pi, n \in \mathcal{N}, && \text{(6.3c)}
\end{aligned}$$

where f_k^s and F^s are the computation resource that the BS allocates to task k and the total computation resource of the BS. Constraint (6.3a) is the computation resource allocation constraint. Constraints (6.3b) and (6.3c) are the value ranges of the offloading ratio, amplitude and phase shift variables, respectively. Since the digital twin–empowered virtual layer has AI ability, we can use AI, such as deep reinforcement learning (DRL), to solve the complex optimization problem. We first reformulate the above optimization problem as DRL with a system state, action, and reward. The state has five components:

$$s(t) = \{d_k(t), c_k(t), f_k^l(t), F_s, \Theta(t)\}. \tag{6.4}$$

In the environment, the BS assembles the information as a state and sends it to the DRL agent. The action has four parts, which are the variables of the optimization problem:

$$a(t) = \{x_k(t), f_k^s(t), \beta_n(t), \theta_n(t)\}. \tag{6.5}$$

Based on the state and action, the agent can produce a reward $\mathcal{R}^{imm}(s(t), a(t))$ from the environment, where the reward is related to the objective function. In this scenario, the total task execution latency can be regarded as the reward function.

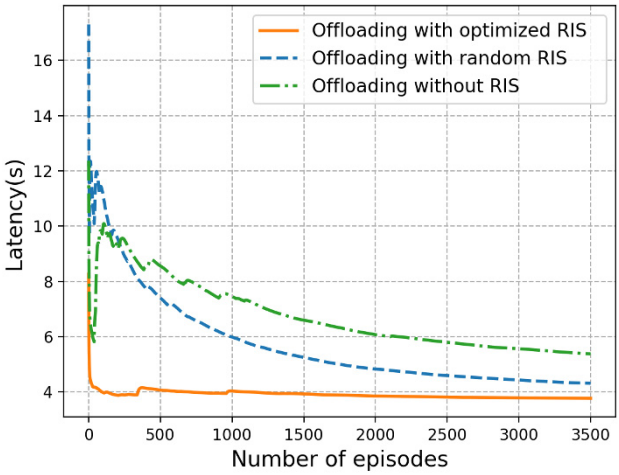

Fig. 6.2 Cumulative task execution latency under different schemes

Based on the state, action, and reward, we exploit asynchronous actor–critic DRL to solve the formulated problem [73]. Asynchronous actor–critic DRL consists of a global agent and several local agents. The global agent accumulates all the parameters of the neural networks from the local agents. Each local agent has an actor neural network and a critic neural network. The actor neural network is for generating actions and the critic neural network is for evaluating the performance of the action generated by the actor neural network. At each training step, the parameter of the actor neural network is updated based on

$$\begin{aligned}\theta_\pi \leftarrow & \theta_\pi + \alpha_\pi \sum_t \nabla_{\theta_\pi} \log \pi(s(t)|\theta_\pi)(\mathcal{R}^{imm}(s(t), a(t)) \\ & + \delta v_{\theta_v}(s(t+1)) - v_{\theta_v}(s(t))),\end{aligned} \tag{6.6}$$

where α_π is the learning rate of the actor network and $\pi(s(t)|\theta_\pi)$ is the output of the actor neural network. The parameter of the critic neural network is updated based on

$$\theta_v \leftarrow \theta_v + \alpha_v \sum_t \nabla_{\theta_v}(\mathcal{R}^{imm}(s(t), a(t)) + \delta v_{\theta_v}(s(t+1)) - v_{\theta_v}(s(t)))^2, \tag{6.7}$$

where α_v is the learning rate of the critic network.

Figure 6.2 shows the total task execution latency of computation offloading under different RIS configuration schemes. First, we can see that the proposed DRL-based computation offloading algorithm converges in all cases and the cumulative task execution latency reduces with the number of episodes. Further, the offloading latency with RIS aid is lower than the latency without RIS aid. The reason is

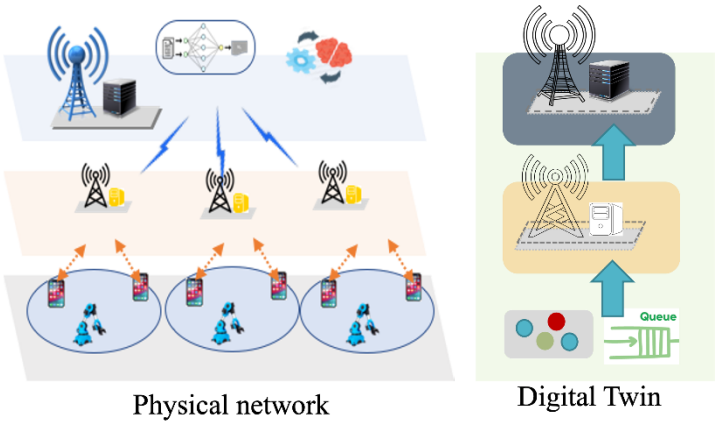

Fig. 6.3 Illustration of a digital twin network

that RIS offloading can achieve a higher transmission data rate, thus resulting in a lower transmission latency. In addition, the offloading latency with optimized RIS configuration is the lowest due the optimal adjustments of the RIS amplitude and phase shift.

6.4 Stochastic Computation Offloading

To improve task processing efficiency and prolong the battery lifetime of mobile devices, computation offloading is a promising approach that can offload the collected data and computation tasks to distributed BSs for processing. However, current research focusing on computation offloading assumes that each device executes a single computation task, without considering the randomness of task arrivals. Such an assumption in the designed policy cannot be applied to a network with a stochastic task arrival model. Since digital twin is a powerful technology that can monitor and analyse the dynamic changes of physical objects, in this section, we utilize digital twin to construct virtual models of the physical objects and solve the stochastic computation offloading problem considering dynamic changes of the task queue.

6.4.1 System Model

We consider a digital twin network consisting of a physical network and its digital twin. As shown in Fig. 6.3, the physical network has three major components: distributed mobile devices, small base stations (SBSs), and a macro base stations (MBS). Each device collects data from sensors and on-device applications, and the

collected data must be analysed in real time. Since data analysis is computation intensive, devices with limited computation capability and battery power might not be able to conduct the data analysis in a timely manner. So, the devices must offload these tasks to edge servers for a high quality of computational experience. Digital twins contain the virtual models of the physical elements. Virtual models not only mirror the characteristics of the physical elements/system, but also make predictions, simulate the system, and can play a crucial role in policy design and resource allocation. In the network, digital twin can be utilized [74] to 1) construct the network topology of the physical network; 2) monitor network parameters and models, that is, dynamic changes of resources and stochastic task arrival processes, and 3) optimize offloading and resource allocation policy.

6.4.2 Stochastic Computation Offloading: Definition and Problem Formulation

Based on digital twin, the digital representation (i.e. virtual models) of the physical network (i.e. virtual world) is created. The virtual models here comprise the wireless network topology, the communication model between the devices and BSs, and the stochastic task queueing model.

(1) Network topology in the digital twin network

Digital twin first models the physical network as a graph $G = (\mathcal{U}, \mathcal{B}, \varepsilon)$, where $\mathcal{U} = \{u_1, .., u_N\}$ and $\mathcal{B} = \{b_0, b_1, ..., b_M\}$ are, respectively, the sets of devices and BSs (where b_0 is the index for the MBS, and the other values are the indexes for the SBSs). The term ε is the edge information, that is, for the connection between the devices and BSs.

Then, the digital twin uses a 3-tuple $DT_i(t)$ to characterize devices, that is, $DT_i(t) = \{p_{i,max}(t), l_i(t), f_i^l\}$, where $p_{i,max}(t)$ denotes the maximal transmission power in time slot t, $l_i(t)$ denotes the current location of u_i, and f_i^l denotes the computation resources of the local server. Similarly, the digital twin uses a 3-tuple $DT_j(t)$ to characterize the BSs, that is, $DT_j(t) = \{l_j(t), w_j, f_j^e\}$, where $l_j(t)$ denotes the current location of b_j, w_j denotes the bandwidth of b_j, and f_j^e denotes the computation resource.

The task offloading between the devices and BSs is facilitated through wireless communication. Here, we consider that devices communicate with the nearest BS for offloading. The wireless communication data rate between device u_i and SBS b_j can be expressed as

$$R_{ij}^s(t) = w_{ij}(t) \log(1 + \frac{p_i(t) h_{ij}^s(t) r_{ij}^s(t)^{-\alpha}}{\sigma^2 + I}), \tag{6.8}$$

where $w_{ij}(t)$ ($w_{ij}(t) \leq w_j$) is the bandwidth that SBS b_j allocates to device u_i in time slot t, $h_{ij}^s(t)$ is the current channel gain, α is the path loss exponent, σ^2 is the noise power, $r_{ij}^s(t)$ is calculated based on the locations of $l_i(t)$ and $l_j(t)$, and I is the

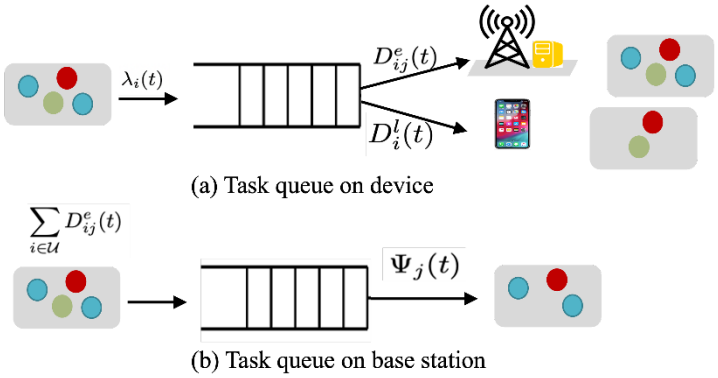

Fig. 6.4 Stochastic computation offloading in digital twin network

interference from other SBSs. With the adoption of orthogonal frequency division multiple access, the interference of different devices in the coverage of the MBS is ignored. The wireless communication data rate between device u_i and the MBS is

$$R^m_{i0}(t) = w_{i0}(t)\log(1 + \frac{p_i(t)h^m_{i0}(t)r^m_{i0}(t)^{-\alpha}}{\sigma^2}), \tag{6.9}$$

where $w_{i0}(t)$ ($w_{i0}(t) \leq w_0$) is the channel bandwidth between device u_i and the MBS in time slot t, $h^m_{i0}(t)$ is the channel gain between device u_i and the MBS, and $r^m_{i0}(t)$ is the distance between device u_i and the MBS.

(2) Stochastic task queueing

At the beginning of time slot t, device u_i inputs the size of the computation task of $\lambda_i(t)$ (bits/slot) into the local dataset. We assume the $\lambda_i(t)$ values in different time slots are independent, and $\mathbb{E}[\lambda_i(t)] = \lambda$. Since device u_i has computation resources, it can execute part of the computation task locally. We consider the size of the computation task that is executed locally as $D^l_i(t)$. The size of the computation task offloaded to BS b_j ($j \in \mathcal{B}$) is $D^e_{ij}(t)$. The rest is stored in a local task buffer, as shown in Fig. 6.4(a). Assume the queue length of the local task buffer is $Q^l_i(t)$ and the queue length is dynamically updated with the following equation:

$$Q^l_i(t+1) = \max\{Q^l_i(t) - \Psi_i(t), 0\} + \lambda_i(t), \tag{6.10}$$

where $\Psi_i(t) = D^l_i(t) + D^e_{ij}(t)$ is the size of the computation task that leaves the task buffer of device u_i during time slot t.

Each edge server also has a task buffer to store the offloaded but not yet executed task. As shown in Fig. 6.4(b), the queue length is dynamically updated by

$$Q^e_j(t+1) = \max\{Q^e_j(t) - \Psi_j(t), 0\} + \sum_{i\in\mathcal{U}} D^e_{ij}(t), \tag{6.11}$$

where $\sum_{i\in\mathcal{U}} D_{ij}^{e}(t)$ is the amount of tasks offloaded from all the devices connected to BS j, and $\Psi_j(t)$ is the size of the computation tasks leaving the edge task buffer. According to the definition of stability in [75], the task queue is stable if all the computation tasks satisfy the following constraints:

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in\mathcal{U}}\mathbb{E}\{Q_i^l(t)\}<\infty, \tag{6.12a}$$

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j\in\mathcal{B}}\mathbb{E}\{Q_j^e(t)\}<\infty. \tag{6.12b}$$

(3) Task offloading in the digital twin network

Let $f_i^l(t)$ be the computation resource of device u_i during time slot t and let c denote the required computation resource for executing one bit of a computation task. Thus, the size of computation tasks executed locally will be

$$D_i^l(t)=\frac{\tau f_i^l(t)}{c}, \tag{6.13}$$

where τ is the duration of the time slot. The energy consumption of a unit of computation resource is $\varsigma(f_i^l)^2$, where ς is the effective switched capacitance, depending on the chip architecture. The local energy consumption for computing task $D_i^l(t)$ can be defined as

$$E_i^l(t)=\varsigma\tau f_i^l(t)^3. \tag{6.14}$$

Devices offload their tasks to BSs via wireless communication. Since the devices are associated with different BSs, the offloaded tasks of device u_i during time slot t can be expressed as

$$D_{ij}^{e}(t)=\begin{cases}R_{ij}^{s}(t)\tau & j\in\mathcal{B}/\{b_0\},\\ R_{i0}^{m}(t)\tau & j=b_0.\end{cases} \tag{6.15}$$

The energy consumption in this case has three parts: the energy consumption for uplink offloading, the energy consumption for computation, and the energy consumption for downlink feedback. The third quantity is generally ignored due to its small data size. Thus, the energy consumption for executing task $D_{ij}^{e}(t)$ on BS b_j can be expressed as

$$E_{ij}^{e}(t)=p_i(t)\tau+\frac{D_{ij}^{e}(t)*c}{f_{ij}^{e}(t)}*\varepsilon, \tag{6.16}$$

where $f_{ij}^{e}(t)$ is the computation resource that b_j allocates to device u_i in time slot t, and ε is the energy consumption for unit computation on edge servers.

The total energy consumption is the combination of local energy consumption, edge server energy consumption, and the transmission energy consumption for computation offloading. Therefore, the total energy consumption can be expressed as

$$E^{tol}(t) = \sum_{i\in\mathcal{U}} E_i^l(t) + \sum_{i\in\mathcal{U}}\sum_{j\in\mathcal{B}} E_{ij}^e(t). \tag{6.17}$$

(4) Stochastic offloading problem

Based on the total energy consumption, we can define network efficiency as

$$\eta_{EE} = \frac{\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\{E^{tol}(t)\}}{\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in\mathcal{U}}\sum_{j\in\mathcal{B}}\mathbb{E}\{D_i^t(t)+D_{ij}^e(t)\}}. \tag{6.18}$$

This is the ratio of long-term total energy consumption to the corresponding long-term aggregate of accomplished computation tasks.

We define $\mathbf{a}(t) = [\mathbf{w}(t), \mathbf{p}(t), \Psi(t), \mathbf{f^l}(t), \mathbf{f^e}(t)]$ as the system action in time slot t, where $\mathbf{w}(t)$ is the bandwidth allocation vector, $\mathbf{p}(t)$ is the transmission power vector, $\mathbf{\Psi}(t)$ is the vector associated with the computation task leaving the edge servers, and $\mathbf{f^l}(t)$ and $\mathbf{f^e}(t)$ are the vectors of computation resources that edge servers allocate to the devices. Taking the network stability constraint into account, the stochastic offloading problem for minimizing η_{EE} can be formulated as

$$\begin{aligned}
\text{P1}: \quad & \min_{\mathbf{a}(t)} \ \eta_{EE} \\
\text{s.t.} \quad & \sum_{i\in\mathcal{U}} \frac{w_{ij}(t)}{w_j} \le 1, \ \ w_{ij}(t) \ge 0, && \text{(6.19a)}\\
& 0 \leqslant p_i(t) \leqslant p_{i,max}(t), && \text{(6.19b)}\\
& 0 \leqslant f_i^l(t) \leqslant f_i^l, && \text{(6.19c)}\\
& \sum_{i\in\mathcal{U}} f_{ij}^e(t) \le f_j^e, \ \ f_{ij}^e(t) \ge 0, && \text{(6.19d)}\\
& \Psi_j(t) * c \le f_j^e \tau, \ \ \Psi_j(t) \ge 0, && \text{(6.19e)}\\
& (6.12a) - (6.12b).
\end{aligned}$$

Constraint (6.19a) is the bandwidth allocation constraint. Constraints (6.19b) and (6.19c) denote the transmission power and computation resource constraints, respectively. Constraint (6.19d) is the computation resource allocation constraint. Constraint (6.19e) implies that the amount of computation resource for processing task Ψ_j cannot exceed the available computation resources.

Problem P1 is a stochastic optimization problem. The complex coupling among optimization variables and mixed combinatorials make P1 difficult to solve. Further, the stochastic task arrival, dynamic channel state information, and dynamic task buffer make it challenging to design an efficient resource management policy for the devices and edge servers. We therefore exploit Lyapunov optimization to transform the original stochastic optimization problem into a deterministic per-time block problem and propose a stochastic computation offloading algorithm to solve P1.

6.4.3 Lyapunov Optimization for Stochastic Computation Offloading

We define the quadratic Lyapunov function as the sum of the squared queue backlogs,

$$L(\Theta(t)) = \frac{1}{2}\{\sum_{i\in\mathcal{U}}[Q_i^l(t) - \beta_i]^2 + \sum_{j\in\mathcal{B}} Q_j^e(t)^2\}, \tag{6.20}$$

where $\Theta(t) = [Q^l(t), Q^e(t)]$ represents the current task queue lengths of the devices and edge servers, and β is a perturbation vector. Further, we define the Lyapunov drift-plus-penalty function as

$$\triangle_V L(\Theta(t)) = \triangle L(\Theta(t)) + V\mathbb{E}[\eta_{EE}(t)|\Theta(t)], \tag{6.21}$$

where $\triangle L(\Theta(t)) = \mathbb{E}[L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)]$ is the conditional drift, and V is a non-negative weight parameter. By minimizing $\triangle_V L(\Theta(t))$, we can ensure network stability and simultaneously minimize network efficiency. The upper bound of $\triangle_V L(\Theta(t))$ can be derived as

$$\begin{aligned}\triangle_V\, L(\Theta(t)) \le C - \sum_{i\in\mathcal{U}}[Q_i^l(t) - \beta_i]\mathbb{E}[\Psi_i(t) - \lambda_i(t)|\Theta(t)] \\ - \sum_{j\in\mathcal{B}} Q_j^e(t)\mathbb{E}[\Psi_j(t) - \sum_{i\in\mathcal{U}} D_{ij}^e(t)|\Theta(t)]\} + V\mathbb{E}[\eta_{EE}(t)|\Theta(t)],\end{aligned} \tag{6.22}$$

where $C = \frac{1}{2}\{\sum_{i\in\mathcal{U}}[\Psi_{i,max}^2 + \lambda_{i,max}^2] + \sum_{j\in\mathcal{B}}[\Psi_{j,max}^2 + (\sum_{i\in\mathcal{U}} D_{ij,max}^e)^2]\}$, and $\Psi_{i,max}, \lambda_{i,max}, \Psi_{j,max}$, and $D_{ij,max}^e$ are the upper bounds of $\Psi_i(t), \lambda_i(t), \Psi_j(t)$, and $D_{ij}^e(t)$, respectively. Based on Lyapunov optimization theory, we can minimize the right side of the inequality in (6.22) to obtain the optimal solution of P1. Specifically, instead of solving P1, we can observe $\Theta(t)$ and $\lambda_i(t)$ to determine $a(t)$ by solving the following problem in each time slot:

$$\begin{aligned}\text{P2}: \min_{\mathbf{a}(t)}\ & V[E^{tol}(t) - \eta_{EE}(t)\sum_{i\in\mathcal{U}}\sum_{j\in\mathcal{B}}(D_i^t(t) + D_{ij}^e(t))] + \sum_{j\in\mathcal{B}} \\ & \{Q_j^e(t)[\sum_{i\in\mathcal{U}} D_{ij}^e(t) - \Psi_j(t)]\} - \sum_{i\in\mathcal{U}}[Q_i^l(t) - \beta_i][\Psi_i(t) - \lambda_i(t)] \\ & \text{s.t.}\quad (6.12a) - (6.12b), (6.19a) - (6.19e).\end{aligned} \tag{6.23}$$

Problem P2 needs to minimize the system cost per time slot. Here, we use DRL to solve P2, because it is efficient for finding a near-optimal solution in real time.

To solve P2, the system first constructs a Markov decision process, that is, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, and then uses a DRL algorithm to explore the actions. From Fig. 6.5, the network state $s(t)$ is constructed by digital twin and output to the DRL agent. To gather network information, digital twin needs to predict the locations, energy, and the generated task flow of the devices and BSs. The locations can be predicted by the K-nearest neighbours classification method in [76]. To prolong the battery

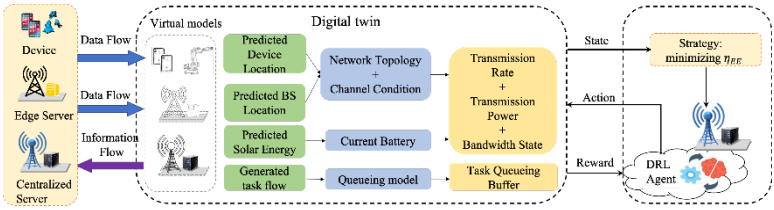

Fig. 6.5 Digital twin–enabled DRL

life of the devices, some of them are equipped with energy-harvesting chips, such as solar panels. Digital twin thus needs to support solar energy prediction here. The generated task flow is based on the application running on each device. Digital twins are used to first predict and gather the information on location, energy, and task flow. Then, based on the gathered information, digital twin updates the network topology, channel condition, and task queueing models. Finally, digital twin generates the current state and transmits it to the DRL agent.

The DRL agent constructs the system state as $s(t) = \{\mathbf{R}(t), \mathbf{F}, \mathbf{p}_{max}(t), \mathbf{w}, \Theta(t)\}$ with wireless data rate, computation resource, transmission power, and task queueing information. Action $a(t) = [\mathbf{w}(t), \mathbf{p}(t), \Psi(t), \mathbf{f}^{\mathbf{l}}(t), \mathbf{f}^{\mathbf{e}}(t)]$ is constructed with the bandwidth allocation, the transmission power, the executed computation task, and the computation resource allocation. It is worth noting that all the variables in action $\mathbf{a}(t)$ are continuous. Thus, we will utilize a policy gradient–based DRL algorithm to explore policy. After executing action $\mathbf{a}(t)$, digital twin updates the system state and estimates the immediate reward $\mathcal{R}^{imm}(s(t), a(t))$. Because the distribution of transition probabilities is often unknown in DRL, the DRL agent utilizes a deep neural network to approximate it. We define the immediate reward function $R^{imm}(s(t), a(t))$ as the objective of P2 problem. After computing the immediate reward, the system updates its state from $s(t)$ to $s(t+1)$ based on action $a(t)$.

We use an online and asynchronous DRL algorithm to explore policy. The online DRL consists of a global agent and multiple learning agents. The detailed policy is explored by the learning agent in each SBS. The policy learned by the learning agent is $a(t) = \pi(s(t)|\theta_\pi)$, where $\pi(s(t)|\theta_\pi)$ is the explored offloading and resource allocation policy produced by a deep neural network. According to $s(t)$ and $a(t)$, the DRL agent can produce the reward and the next state. To estimate the performance of the proposed DRL algorithm, we consider a network topology with one MBS, $M = 3$ SBSs, and $N = 20$ devices. Each learning agent has an actor network and a critic network. The actor network has three fully connected hidden layers, each with 128 neurons, and an output layer with eight neurons using the softmax function as the activation function. The critic network has three fully connected hidden layers, each with 128 neurons, and one linear neuron output layer.

Figure 6.6 depicts the system costs with respect to training episodes under different schemes. The green curve is the benchmark of the joint optimization of computation offloading, the bandwidth, and the transmission power, but without computation resource allocation. The orange curve is the benchmark of the joint optimization of

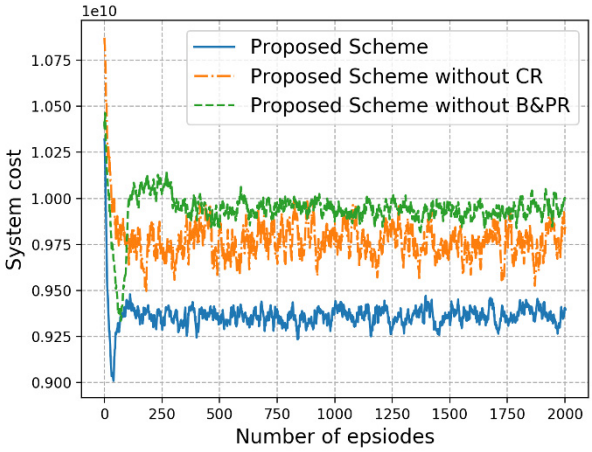

Fig. 6.6 System costs under different schemes

the computation offloading and computation resource allocation. Figure 6.6 shows that the performance of the proposed scheme outperforms the two benchmarks, since it can concurrently optimize computation offloading, the bandwidth, the transmission power, and the computation resource allocation. In addition, the system cost of the orange curve is lower than that of the green curve. This means that, compared with the optimization of computation resources, the joint optimization of the bandwidth and transmission power has a greater influence on performance.

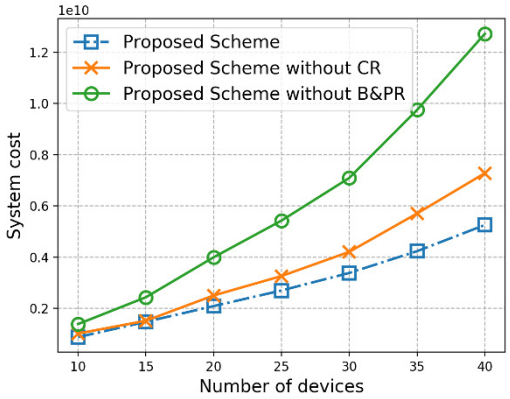

Fig. 6.7 System costs with respect to the number of devices under different schemes

Figure 6.7 compares the system costs with respect to the number of devices under different schemes. The number of devices ranges from 10 to 40. From Fig. 6.7, we

can make two observations. First, for each of the three schemes, the system cost increases with the number of devices. The reason is that the increase of devices leads to more offloading requests, resulting in the consumption of more communication and computation resources. Second, the performance of the proposed algorithm outperforms two benchmarks by jointly optimizing computation offloading, bandwidth, transmission power, and computation resource allocation.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.