Chapter 7
The Future of Mobile Edge Computing

Abstract This chapter first introduces the fundamental principles of blockchain and the integration of blockchain and mobile edge computing (MEC). Blockchain is a distributed ledger technology with a few desirable security characteristics. The integration of blockchain and MEC can improve the security of current MEC systems and provide greater performance benefits in terms of better decentralization, security, privacy, and service efficiency. Then, the convergence of artificial intelligence (AI) and MEC is presented. A federated learning–empowered MEC architecture is introduced. To improve the performance of the proposed scheme, asynchronous federated learning is proposed. The integration of blockchain and federated learning is also presented to enhance the security and privacy of the federated learning–empowered MEC scheme. Finally, more MEC enabled applications are discussed.

7.1 The Integration of Blockchain and Mobile Edge Computing (MEC)

MEC can offer a series of edge services with task processing, data storage, heterogeneity support, and QoS improvement capabilities. In close proximity to devices, MEC can provide instant computing applications with low latency and fast service response. The distributed structure of edge computing also potentially facilitates ubiquitous computing services, scalability, and network efficiency improvement. However, the MEC infrastructure still has unresolved challenges in terms of security and privacy. First, the large amount of heterogeneous data being collected, transferred, stored, and used in dynamic MEC networks can easily suffer serious data leakage. Further, due to the high dynamism and openness of MEC systems, it is very challenging to save the setting and configuration information of the edge servers in a reliable and secure way. Blockchain can enhance the security and privacy of MEC by offering many promising technical properties, such as decentralization, privacy, immutability, traceability, and transparency. The integration of blockchain and MEC can enable secure network orchestration, flexible resource management, and system performance improvements. In this section, we first introduce the structure of

Y. Zhang, *Mobile Edge Computing*, Simula SpringerBriefs on Computing 9,
https://doi.org/10.1007/978-3-030-83944-4_7

blockchain and then present three potential cases of the integration of blockchain and MEC as future research directions.

7.1.1 The Blockchain Structure

Blockchain is an open database that maintains an immutably distributed ledger typically deployed in a peer-to-peer network. The structure of blockchain is shown in Fig. 7.1 and consists of three essential components: transactions, blocks of transaction records, and a consensus algorithm. The transaction information includes node pseudonyms, data types, metadata tags, a complete index history of metadata, an encrypted link to transaction records, and a timestamp of a specific transaction. Each transaction is encrypted and signed with digital signatures to guarantee authenticity. The digitally signed transactions are arbitrarily packed into a cryptographically tamper-evident data block. The blocks are linked in linear chronological order by hash pointers to form the blockchain. To maintain the consistency and order of the blockchain, a consensus algorithm is designed to generate agreement on the order of the blocks and to validate the correctness of the set of transactions constituting the block.

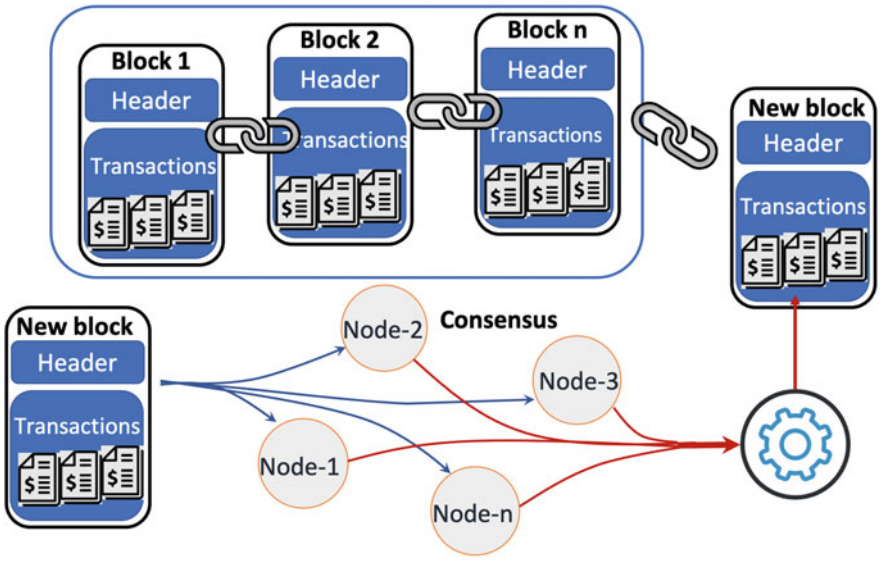

Fig. 7.1 The blockchain structure

7.1.1.1 Transactions

A transaction is the unit data structure of a blockchain, and it is created by a set of users to indicate the transfer of tokens from a specific sender to a receiver. Transactions generally consist of a recipient address, a sender address, and a token value. The input of a transaction is a reference that contains a cryptographic signature. The output of a transaction specifies an amount and an address. Transactions are bundled and broadcast to each node in the form of a block. As new transactions are distributed throughout the network, they are independently verified by each node. To protect the authenticity of a transaction, the functionalities of cryptographic hashing and asymmetric encryption are utilized, as follows.

- *Hash function*: A cryptographic hash function maps an arbitrary-size binary input into a fixed-size binary output. For example, SHA-256 maps an arbitrary-size input to a binary output 256 bits. The binary output is called a hash value. Moreover, the same input will always provide the same hash output. The probability of generating the same output for any two different inputs is negligible; it is thus impossible to reconstruct the input based on a hash output. The hash of a transaction makes it easy to keep track of transactions on the blockchain. In Bitcoin, SHA-256 and RIPEMD160 are utilized as hash function to produce a bitcoin address. In Ethereum, Keccak-256 is utilized as a hash function to produce a public key. In addition, signatures and private keys in blockchain frequently use hash functions to ensure security.
- *Asymmetric encryption*: Asymmetric encryption provides a secure method for authentication and confidentiality. Each node in a blockchain has a pair of keys: a public key and a private key. The public key can be shared with anyone to encrypt a message, whereas the private key should only be known to the key's initiator. In blockchain, the public key is used as the source address of transactions to verify their genuineness. The cryptographic private key is used to sign a transaction, which outputs a fixed-size digital signature for any arbitrary-size input message. The verification result will be true if the digital signature has the correct private key and input message. An elliptic curve digital signature algorithm is a typical algorithm for digital signing transactions. In the elliptic curve digital signature algorithm, when a user (A) wants to sign a transaction, that user first hashes the transaction and then uses his or her private key to encrypt the hashed transaction. The user then broadcasts the encrypted transaction. When another user receives the transaction and wants to verify its correctness, that user can decrypt the signature with user A's public key and hash the received transaction to verify whether the transaction information has been changed.

In blockchain, each transaction is broadcast over the entire network and cross-verified by multiple nodes. The verified transactions are ordered consecutively with linearly ordered timestamps to guarantee correctness.

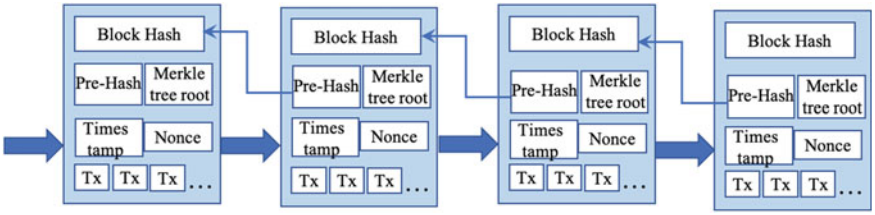

Fig. 7.2 Block structure

7.1.1.2 The Block Structure

A blockchain is a sequence of blocks that holds a complete list of transaction records. A block in a blockchain contains the hash of the current block, the hash of the previous block, a Merkle tree root, a timestamp, a nonce, and transactions, as shown in Fig. 7.2.

- *Block hash*: A block hash is the principal block identifier. It is a cryptographic digest made by hashing the block header twice with the SHA-256 algorithm. It identifies a block uniquely and unambiguously, and it can be independently derived by any node by simply hashing the block header.
- *Previous hash*: The hash of the previous block, which is a 256-bit hash that points to the previous block, is a necessary data field for the block header. Based on the previous block hash, all blocks are linked together to form a chain. If any block is tampered with, this will cause a change in all subsequent block hash pointers. When a block and all previous blocks are downloaded from an untrusted node, block hashing can be used to verify whether any block has been modified.
- *Merkle tree*: A Merkle tree represents a transaction set in the form of a binary tree for quick validation and synchronization. In the tree, the leaf nodes are the lowest tier of nodes, and each leaf node is a hash of a transaction. Each non-leaf node is a hash of the concatenation of two child nodes. The root node of the Merkle tree is known as the Merkle digest or root. Adjacent leaves are concatenated pairwise, and the hash of the concatenation constitutes the node's parent. Parent nodes are concatenated and hashed similarly to generate another level of parent nodes. This process is repeated until a single hash remains, which is the Merkle root. The Merkle tree is useful because it allows users to verify whether a transaction has occurred, based only on the direct branch from the transaction node to the Merkle root path. Moreover, the Merkle root allows tampering of any transaction data to be detected, to ensure their integrity.
- *Timestamp*: The time the block was generated. In blockchain, every block has a timestamp and the timestamp can be referred to as proof of existence. According to Satoshi Nakamoto's white paper [107], a decentralized timestamp service can resolve the double-spending problem. It can also help improve the traceability and transparency of the data stored in the blockchain.
- *Nonce*: A nonce is random number, and it can be used only once. Each node competes to find the nonce first to obtain the correct packing transactions for the

newly generated block. The nonce is difficult to find and is considered a way to weed out less talented miners. Once the nonce is found, it is added to the hashed block. With this number, the hash value of that block will be rehashed, creating a difficult algorithm.

The first block in any blockchain is termed the genesis block. This block is sometimes referred to as block 0. Every block in a blockchain stores a reference to the previous block. However, the genesis block has no previous block for reference.

7.1.1.3 Consensus Algorithms

Blockchain is a distributed decentralized network that provides immutability, privacy, security, and transparency. There is no central authority to validate and verify the transactions, but the transactions are considered secured and verified. This is possible because of consensus. Consensus is a process that allows all nodes to reach a common agreement on the state of a distributed ledger. The consensus problem can be formulated as a Byzantine fault–tolerant problem, that is, how generals can come to a common conclusion in the presence of a small number of traitors and miscommunications. The consensus currently used in most blockchain networks can be split into two categories: probabilistic-finality consensus and absolute-finality consensus. In probabilistic-finality consensus, any block in a blockchain can be reverted with a certain probability; attackers could thus accumulate a large amount of computational power, or stake, to create a long private chain to replace a valid chain. In absolute-finality consensus, a transaction is immediately finalized once it is included in a block. In other words, a new block generated by a leader node is committed by sufficient nodes before submission to the blockchain. We next present several common consensus strategies in blockchain.

- *Proof of work (PoW)*: PoW is a consensus strategy used in Bitcoin, where one node is selected to create a new block in each round of consensus through a computational power competition. In the competition, all participants must solve a cryptographic puzzle by using different nonces until the target is reached. The node that first solves the puzzle has the right to create a new block. Solving a PoW puzzle is costly and time-consuming, but it is easy for other nodes to verify. PoW guarantees security, based on the principle that it is impossible for a malicious attacker or group to collect more than 50% of the network's computational power to control the consensus process. PoW is a probabilistic-finality consensus protocol to guarantee eventual consistency. In PoW, nodes must consume a great deal of energy to solve the cryptographic puzzle. However, this work is useless and the energy consumed is wasteful. To tackle the resource waste problem of PoW, the idea of proof of useful resources was designed. Primecoin proposed a consensus algorithm to turn useless PoW into a meaningful search for special prime numbers when seeking a nonce [108]. Permacoin utilized bitcoin mining resources to distributively store an extremely large data provided by an authoritative file dealer based on proof of retrievability [109]. Instead of wasting energy for PoW, proof

of burn allows miners to burn virtual currency tokens and then grants miners the right to write blocks in proportion to the number of burned coins [110].

- *Proof of stake (PoS)*: PoS is an energy-saving consensus to replace PoW. Instead of consuming large amounts of computational power to solve a PoW puzzle, PoS selects one node to create the next block based on the amount of stake. PoS is a probabilistic-finality consensus protocol, where the chances of being a block creator depends on "wealth". Since the richest node is bound to dominate the network, creator selection based on the amount of stake is quite unfair. Therefore, many researchers have proposed new schemes to decide on the node to forge the next block. Peercoin proposed a metric of coin age to measure the amount of held coins and their holding time [111]. In Peercoin, the node with older and larger sets of coins has a higher probability of creating the next block. Compared with PoW, Peercoin can reduce energy consumption and become more efficient. Ouroboros proposed PoS-based consensus, considering that stakes will shift over time [112]. A secure multiparty coin-flipping protocol was proposed in Ouroboros to guarantee the randomness of the leader election in the block generation process. To combine the benefits of PoW and PoS, proof of activity was proposed [113]. In proof of activity, the leader in each round of consensus is selected based on a standard PoW-based puzzle competition to generate an empty block header, where the stakeholders participating in the block verification receive a reward.
- *Delegated PoS (DPoS)*: The main difference between PoS and DPoS is that PoS involves direct democracy, whereas DPoS involves representative democracy [114]. In DPoS, stakeholders vote to elect delegates. The elected delegates are responsible for block creation and verification. Voting in DPoS is important, since it enables stakeholders to give delegates the right to create blocks, instead of creating blocks themselves; DPoS can thus reduce the computational power consumption of stakeholders to zero. On the other hand, PoW with plenty of nodes participating in the block verification process. In DPoS, only fewer delegates participate in the block verification process, thus the block can be confirmed quickly and the transactions can be confirmed quickly. Compared to PoW and PoS, DPoS is a low-cost, high-efficiency consensus protocol. Additionally, stakeholders do not need to worry about dishonest delegates, because these delegates can be easily voted out. There are also cryptocurrencies that implement DPoS, such as BitShares [115] and EoS. The new version of EoS has extended DPoS to DPoS–Byzantine fault tolerance. [116].
- Practical Byzantine fault tolerance (PBFT): PBFT is a Byzantine fault tolerance protocol with low algorithm complexity and high practicality [117]. Even if some nodes are faulty or malicious, network liveness and safety are guaranteed by PBFT, as long as a minimum percentage of nodes are connected, working properly, and behaving honestly. Hyperledger Fabric [118] utilizes PBFT as its consensus algorithm. In PBFT, a new block is determined in a round. In each round, a primary node is selected as the leader to broadcast the message sent by the client to other nodes. PBFT can be divided into three phases: pre-prepare, prepare, and commit. In each phase, a node enters the next phase if it has received votes from over two-thirds of all nodes. PBFT guarantees the nodes maintain a common state and take

Table 7.1 Main consensus comparison

	Type	Fault tolerance	Power consumption	Scalability
PoW	Probabilistic finality	50%	Large	Good
PoS	Probabilistic finality	50%	Less	Good
DPoS	Probabilistic finality	50%	Less	Good
PBFT	Absolute finality	33%	Negligible	Bad

consistent action in each round of consensus. PBFT achieves strong consistency and is thus an absolute-finality consensus protocol.

In distributed systems, there is no perfect consensus protocol. The consensus protocol should be adopted based on detailed application requirements. We present a simplified comparison of different consensus algorithms in Table 7.1.

7.1.2 Blockchain Classification

Current blockchain systems can be roughly classified into three types: public blockchains, consortium blockchains, and private blockchains. We compare these three types of blockchains from different perspectives.

- *Consensus determination*: In a public blockchain, each node can take part in the consensus process. In a consortium blockchain, only a selected set of nodes is responsible for validating a block. In a private blockchain, one organization fully controls and determines the final consensus.
- *Permission*: All transactions in a public blockchain are visible to the public. In a private or consortium blockchain, permissions depends on the organization or consortium decides whether the stored information is public or restricted.
- *Immutability*: Since transactions are stored in different nodes in the distributed network, it is nearly impossible to tamper with the public blockchain. However, if the majority of the consortium or the dominant organization wants to tamper with the blockchain, the consortium blockchain or private blockchain can be reversed or altered.
- *Efficiency*: It takes time to propagate transactions and blocks, since there are a large number of nodes in a public blockchain network. Taking network safety into consideration, restrictions on a public blockchain are much stricter. Therefore, transaction throughput is limited and latency is high. With fewer validators, consortium and private blockchains can be more efficient.

Table 7.2 Comparison of the different types of blockchains

	Public blockchain	Private blockchain	Consortium blockchain
Energy cost	High	Low	Low
Delay	Long	Short	Short
Security	High	Low	High

- *Centralization*: The main difference between the three types of blockchains is that a public blockchain is decentralized, a consortium blockchain is partially centralized, and a private blockchain is fully centralized, because it is controlled by a single group.
- *The consensus process*: Anyone can join the consensus process of a public blockchain. Different from public blockchains, both consortium and private blockchains are permissioned. A node needs to be certified to join the consensus process in consortium and private blockchains.

We compare the three types of blockchains in terms of energy costs, delay, and security, as shown in Table 7.2. Since a public blockchain often uses PoW to achieve consensus, it incurs high energy costs and long delays. A private blockchain is associated with low energy consumption and short delays to achieve consensus because of centralization. A consortium blockchain utilizes permissioned nodes to create new blocks without a mining process; it also therefore has low energy consumption and can achieve consensus quickly.

7.1.3 Integration of Blockchain and MEC

Many devices in MEC share their resources or content openly, without consideration of personal privacy. The integration of blockchain and MEC can establish a secure and private MEC system.

7.1.3.1 Blockchain for Edge Caching

With the rapid development of the Internet of Things (IoT) and wireless technologies, the huge amounts of data and content are undergoing exponential growth. To support massive content caching while also satisfying the low-latency requirements of content requesters, MEC provides distributed computing and caching resources in close proximity to users. Thus, content can be processed and then cached at the network edge, to alleviate data traffic on backhaul links and reduce content delivery latency. Since state-of-the-art devices are equipped with a certain amount of caching resources, a device with sufficient caching resources can be regarded as a caching

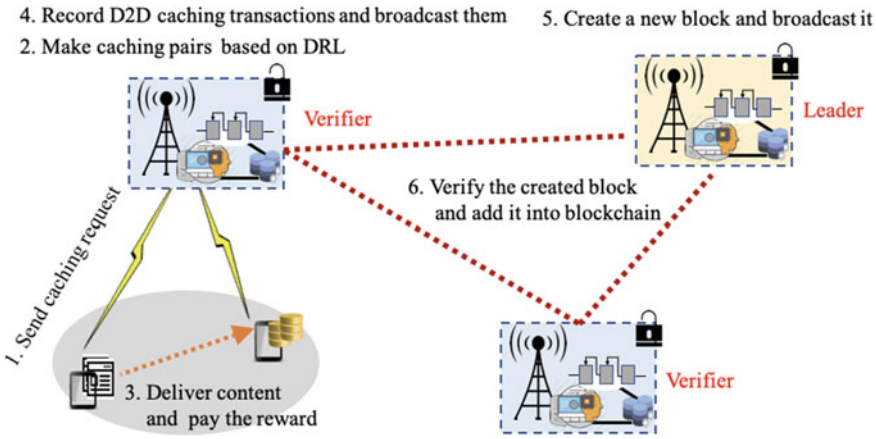

Fig. 7.3 Blockchain-empowered secure content caching

provider, to expand the caching capacity of the network edge. However, content usually involves the generator's sensitive personal information, such that devices might be not willing to store their content with an untrusted caching provider. A secure caching scheme among untrusted devices therefore needs to be built.

Blockchain enables untrusted nodes to interact with each other in a secure manner and provides a promising method for edge caching. We propose a blockchain-empowered distributed and secure content caching framework, as shown in Fig. 7.3 In this content caching system, devices can have two roles: a resource-constrained device with large-scale content is defined as a caching requester, and a device with sufficient caching resources is defined as a caching provider. Base stations are distributed in a specific area to work as edge servers. Specifically, if a content is successfully cached at one caching provider, the caching requester should create a transaction record and send it to the nearest base station. Base stations collect and manage local transaction records. The transaction records are structured into blocks after the consensus process among the base stations is completed and then stored permanently in each base station. The detailed processes are as follows.

- *System initialization*: For privacy protection, each device needs to register a legitimate identity in the system initialization stage. In an edge caching blockchain, an elliptic curve digital signature algorithm and asymmetric cryptography are used for system initialization. A device can obtain a legitimate identity after its identity has been authenticated. The identity includes a public key, a private key, and the corresponding certificate.
- *Roles in edge caching*: Devices choose their roles (i.e., caching requester and caching provider) according to their current caching resource availability state and future plans. Mobile devices with surplus caching resources can become caching providers to provide caching services for caching requesters.

- *Caching transactions*: Caching requesters send the amount of caching resources and expected serving time to the nearest base station. The base station broadcasts all received caching requests to local caching providers. Caching providers provide feedback on the availability of caching resources to the base station and their future plans. Each base station then utilizes a deep reinforcement learning algorithm to match the caching supply and demand pairs among the devices, determines the caching resources that each caching provider can provide, and allocate bandwidth between the base station and the devices.
- *Building blocks in a caching blockchain*: Base stations collect all the transaction records in a certain period and then encrypt and digitally sign them to guarantee their authenticity and accuracy. The transaction records are structured into blocks, and each block contains a cryptographic hash of the prior block in the consortium blockchain. To verify the correctness of a new block, the consensus algorithm (e.g., PBFT) is used. In the consensus process, one of the base station is selected as the leader for creating the new block. Because of broadcasts, each base station has access to the entire transaction record and has the opportunity to be the leader. In a consortium blockchain, the leader is chosen before the block building and does not change before the consensus process is completed.
- *The consensus process*: The leader broadcasts the created block to other base stations for verification and audit. All the base stations audit the correctness of the created block and broadcast their audit results. The leader then analyzes the audit results and, if necessary, sends the block back to the base stations for another audit. Following the audit results and corresponding signatures, compromised base stations will be discovered and held accountable.

The integration of blockchain and MEC can improve the security of edge networks and extend edge caching and resource sharing among untrusted entities.

7.1.3.2 Blockchain for Energy Trading

Due to harvesting and information communication technologies, distributed renewable energy sources are increasingly being integrated into smart grids, and vehicles not only can charge electricity from a home grid with renewable energy sources, but also can obtain electricity from other vehicles, to shift peak load through energy trading. However, because of privacy concerns, smart vehicles with surplus electricity might not be willing to work as energy suppliers in an energy trading market. To encourage vehicles with surplus electricity to participate in energy trading, the privacy of smart vehicles during the trade must be protected.

Blockchains, with its desirable characteristics of decentralization, immutability, accountability, and trustlessness, can significantly improve network security and save operational costs. Peer-to-peer topology enables electricity trading to be carried out in a decentralized, transparent, and secure market environment. The authors in [121] proposed a secure energy trading system with three types of components: vehicles, edge servers, and smart meters. The vehicles play three roles in electricity trad-

ing, with charging vehicles, discharging vehicles, and idle vehicles. Each vehicle chooses its own role based on its current energy state. Edge servers provide electricity and wireless communication services for the vehicles. Each charging vehicle sends a request about electricity demand to the nearest edge server. The edge server announces the energy demand to other vehicles (plug-in hybrid electric vehicles). Vehicles with surplus electricity submit selling prices to the edge server. After a double auction, two vehicles carry out an electricity trade. Smart meters are utilized to calculate and record the amount of electricity traded. Charging vehicles pay the discharging vehicles, based on the records in the smart meters. The detailed processes of the energy blockchain are similar to those in the caching blockchain, but there is still a very big difference. A caching blockchain utilizes a PBFT consensus algorithm, which requires relatively little energy and time to achieve consensus, because no mining process is involved. The work to achieve consensus is based on PoW. Although more energy and time must be spent for consensus, all the vehicles in a blockchain can participate in the process of verifying transactions, creating blocks, and achieving consensus.

7.2 Edge Intelligence: The Convergence of AI and MEC

The rapid development of AI techniques and applications has provided new possibilities for MEC. The integration of AI algorithms with MEC can considerably improve the intelligence and performance of edge computing. Conventional AI approaches rely on centralized mechanisms that invite serious security and privacy threats and are not suitable for resource-constrained edge networks. Federated learning and transfer learning are two emerging paradigms that shine new light on the convergence of AI and MEC.

7.2.1 Federated Learning in MEC

Increasing concerns of data privacy and security are hindering the wide implementation of AI algorithms to edge networks. Federated learning [122, 123] is proposed as a new learning scheme that enhances data privacy. Users participating in federated learning collaboratively train a global model and preserve their own data locally. Thus, by executing distributed training across users locally, federated learning enhances data privacy and reduces the cost of data transmission. By applying federated learning in MEC systems, the decision making process can be executed on edge devices, which reduces system latency and improves decision efficiency. Federated learning is believed to be one of the strongest enabling paradigms for large-scale MEC systems.

With the benefits of privacy enhancement, decentralization, and collaboration, federated learning has attracted significant attention in wireless networks. For exam-

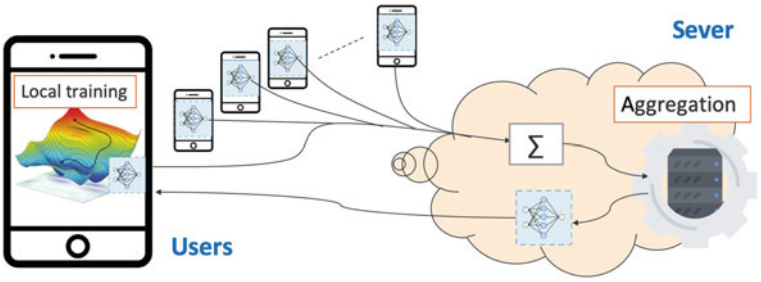

Fig. 7.4 Federated learning–empowered MEC

ple, Google exploited federated learning to train machine learning (ML) models for keyboard prediction [124]. Z. Yu et al. [125] proposed a federated learning–based proactive content caching scheme where the content caching policies are calculated by federated learning algorithms. However, in federated learning, the iterative communication between end users and the server and the local training of machine learning models by end users also consumes a large amount of resources.

To apply federated learning to MEC applications, a good volume of work has explored how to improve the performance of federated learning by optimizing the constrained resources in edge networks. J. H. Mills et al. [126] proposed adapting federated averaging [127] by adopting distributed Adam optimization to reduce the number of communication rounds for convergence. S. Wang et al. [128] proposed a control scheme to determine the optimal execution trade-off between local training and global aggregation within a given resource budget. In [129], C. Dinh et al. optimally allocated computation and communication resources in the network to improve the performance of federated learning deployed in wireless networks.

7.2.1.1 A Federated Learning–Empowered MEC Model

The architecture of federated learning–empowered MEC systems is depicted in Fig. 7.4. The end users in the system are the clients of federated learning, and the edge servers are the aggregation server of federated learning. For end user u_i with dataset D_i, the loss function for local training is defined as

$$F_i(w) = \frac{1}{|D_i|} \sum_{j \in D_i} f_j(w, x_j, y_j) \tag{7.1}$$

where $f_j(w, x_j, y_j)$ is the loss function on data sample (x_j, y_j) with parameter vector w, and $|D_i|$ is the size of the data samples in D_i. The loss function $f_j(w, x_j, y_j)$ is determined according to the specific learning algorithms, such as the mean squared error and the mean absolute error. The global loss function in federated learning is defined as

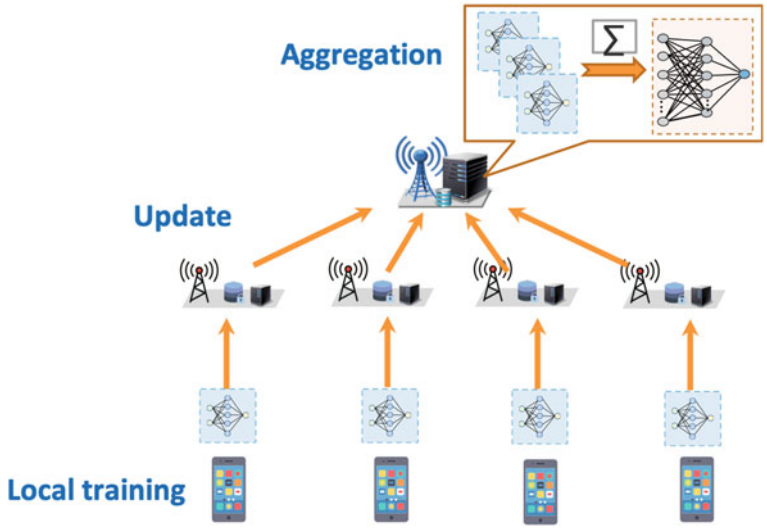

Fig. 7.5 Processes of federated learning–empowered MEC

$$F(w) = \frac{1}{|D|} \sum_{j \in D} f_j(h(w, x), y) = \frac{1}{|D|} \sum_{i} |D_i| \cdot F_i(w) \tag{7.2}$$

where $|D|$ is the size of the total training data $|D| = \sum_i |D_i|$. The objective of federated learning is to find the parameter vector w that minimizes the global loss function $F(w)$, that is,

$$Q(w, t) = \arg\min_{i \in N, t \leq T} F(w) \tag{7.3}$$

$$\text{such that} \quad \forall u_i \in U, i \in \{1, 2, \ldots, N\} \tag{7.4}$$

where $u_i \in U$ denotes the user participating in the federated learning training process.

The general architecture of the federated learning–empowered MEC system consists of two planes: the end user plane and the edge server plane. As shown in Fig. 7.5, local training is executed in the user plane, while global aggregation is executed in the edge server plane. The federated learning–empowered MEC system involves three main steps: local training, parameter updating, and global aggregation. The MEC server plays the role of global server, and the end users, with mobile phones, smart vehicles, and IoT devices, and so on, are clients of federated learning. The three steps are repeated in the system to train the global machine learning model. Computation tasks are executed by running the federated learning algorithms in the MEC system.

- *Local training in the user plane:* The aggregation server distributes the ML model $\mathcal{M}$ to end users in the initialization phase. Each of the end users then trains the shared model $\mathcal{M}$ based on their local datasets. Gradient descent approaches are

widely used in the training process. The model parameters $w_i(t)$ of iteration t are updated as

$$w_i(t) = w_i(t-1) - \eta \cdot \nabla F_i(w_i(t-1)), \tag{7.5}$$

where η is the learning rate, and $\nabla F_i(w_i(t-1))$ is the gradient of the loss function with parameters $w_i(t-1)$. The users then transmit the trained parameters $w(t)$ to the server for aggregation.

- *Global aggregation in the edge plane:* As denoted in Fig. 7.5, the MEC server collects all the parameters $\sum_i w_i(t)$ and calculates the aggregated model. The average aggregation is widely adopted to obtain the global model, as

$$w(t) = \frac{1}{\sum_{i=1}^{N} |D_i|} \sum_{i=1}^{N} |D_i| \cdot w_i(t) \tag{7.6}$$

The MEC server then transmits the aggregated global model to the end users to start a new training iteration. The learning process continues until the trained model reaches a predefined accuracy threshold or the execution time runs out.

7.2.1.2 Performance-Improved Asynchronous Federated Learning in MEC

Federated learning–empowered MEC systems can enlarge the scale of the training data and protect the data privacy of end users. However, new challenges have also arisen in the deployment of federated learning in MEC systems. First, the iterative update process of federated learning increases the transmission burden in communication resource–constrained edge networks. Second, the heterogeneous communication and computing capabilities of end users hinder the fast convergence of the learning process. Third, the risk of fake parameters from malicious participants also exists. To address these issues, a primary approach is to reduce the execution delay of federated learning. Thus, asynchronous federated learning is proposed.

In conventional federated learning, a synchronous mechanism is maintained by the clients and the global server to update the trained parameters and aggregate the global model. All the users participate in the global aggregation in each round. The training times of different end users varies greatly, because of their heterogeneous computing capabilities and dynamic communication states. In such a case, the execution time of each iteration is determined by the slowest clients, which incurs a high waiting cost for others, due to the heterogeneous runtimes. Asynchronous federated learning optimally selects a portion of the users to participate in global aggregation, while others continue with local training. Different optimizing approaches can be used as the node selection algorithm to decide on the participating nodes based on their capabilities. An overview of the asynchronous federated learning–empowered MEC scheme is shown in Fig. 7.6.

The asynchronous federated learning scheme comprises the following phases.

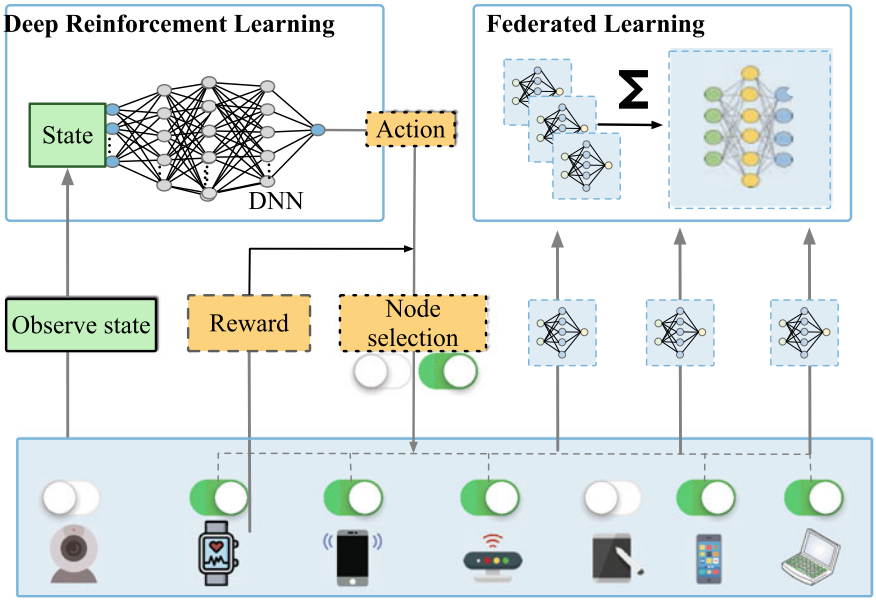

Fig. 7.6 Asynchronous federated learning–empowered MEC

- *Node selection:* Participating nodes are selected from all the end users through a node selection algorithm, according to their communication states and available computing resources. End users with sufficient resources are prone to being selected as participating nodes.
- *Local training and aggregation:* The participating nodes train their local models $m_i(t)$ according to their local data and obtain the parameters $w_i(t)$ for the trained model $m_i(t)$. User i also executes local aggregation by retrieving parameters $w_j(t)$ from nearby end users through device-to-device communication.
- *Global aggregation:* The MEC server carries out global aggregation based on local model parameters it has collected from participating end users, following Eq. (7.6). The global model $\mathcal{M}(t)$ is then broadcast to the end users to start a new learning iteration.

Deep reinforcement learning can be widely exploited as the node selection algorithm, deployed at the MEC server. The deep reinforcement learning algorithm learns the optimal node selection policy by using deep neural networks to approximate the policy gradient. Other techniques, such as convex optimization and game theory, can also be used in the node selection process.

7.2.1.3 Security-Enhanced AI in MEC: Integrating Blockchain with Federated Learning

In federated learning–empowered MEC systems, the parameters transmitted between end users and the MEC server are subject to serious security and privacy issues. The risk of data leakage increases, since an attacker can infer information on the original training data from these parameters. Moreover, malicious participants can upload fake parameters or use poisoned data to train their local models, which can cause the failure of the entire federated learning process. In addition, as the global aggregator, MEC servers also raise the risk of a single point of failure or malicious attacks. Building a trust mechanism among untrusted end users and MEC servers is therefore essential.

Blockchain has achieved great success in providing secure collaboration mechanisms among untrusted users. We propose integrating blockchain with federated learning to provide trust, security, and intelligence in MEC systems.

- *Blockchain for federated learning:* Blockchain provides a trusted collaboration mechanism for all participants (users) of federated learning. Through the authorization mechanism and identity management of the blockchain, especially a permissioned blockchain, users lacking mutual trust can be united to establish a secure and trusted cooperation mechanism. In addition, the model parameters of federal learning can be stored in the blockchain to ensure their safety and reliability.
- *Federated learning for blockchain:* The contradiction between the limited storage capacity of blockchain nodes and the larger storage demands of blockchains has always been a bottleneck in blockchain development. By processing the original data through federated learning, blockchains can store only the computation results, reducing storage cost and communication overhead. In addition, based on federated learning, the authentication calculation and transmission scheduling of blockchain transactions are optimized, which can considerably improve blockchain performance.

Based on the above analysis, we propose integrating blockchain with federated learning to build a trusted, secure, and intelligent MEC system. The integrated architecture is illustrated in Fig. 7.7. The architecture can be divided into the end user layer and the edge service layer. Users mainly consist of smart devices, such as IoT devices and mobile phones. The servers are represented by base stations equipped with MEC servers with certain storage and computing capabilities.

The integrated scheme consists of two main modules: federated learning and a permissioned blockchain. The federated learning learns the model parameters through local training on the user side, while the blockchain runs on the MEC server to collect and store the parameters of the federated learning. The parameters are verified by the consensus protocol. The detailed processes are as follows.

- *Local training:* Based on their local data, participating users train the model parameters through a gradient descent algorithm to minimize the loss function.

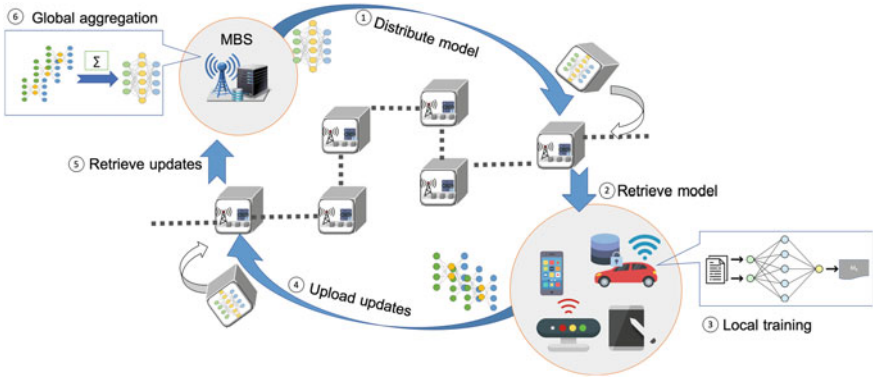

Fig. 7.7 The integration of blockchain and federated learning

- *Parameter transmission:* The trained local parameters are transmitted to the base station in the edge service layer through wireless links. The parameters of each user are collected and stored in blockchain nodes in the form of transactions.
- *Block generation:* Each blockchain node collects the transactions (model parameters) from the user layer and packages them into blocks using encryption and signatures. The block generator is determined by the consensus mechanism. The blockchain node that obtains the right to generate blocks broadcasts the block to the entire blockchain network and adds the block to the blockchain after verification.
- *Global aggregation:* The aggregator, that is, the MEC server, in the edge service layer aggregates model parameters according to the records in the blockchain and updates them into the global model. Furthermore, the global model is distributed to all participating users to start a new round of training.

The integration of blockchain and federated learning combines the security and trust of blockchains with the distributed intelligence of federated learning, which improves the security and data privacy of the MEC system.

7.2.2 Transfer Learning in MEC

7.2.2.1 Applying Transfer Learning in MEC

Transfer learning, as one of the machine learning methods, aims to transfer knowledge from existing domains to a new domain by learning across domains with non-independent and identically distributed data. Specifically, in transfer learning, a model developed for a task can be used as the original model for a related task. The basic idea of transfer learning is learning to learn, that is, to retain and reuse previously learned knowledge in the machine learning process. Different from traditional

machine learning techniques, the source task and the target task are not the same, but related. The definition of transfer learning is as follows [130].

Definition 7.1 Given a source domain $\mathcal{D}_S$, a learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$, and a target learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using knowledge learned in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$.

To apply transfer learning to an MEC system, the following three main transfer learning research issues need to be addressed.

- *What to transfer:* Some knowledge can be specific to individual domains or tasks, while some knowledge can be common to both the source and target domains. It is therefore essential to determine which part of the knowledge can be transferred from the source domain to the target domain. The transferred knowledge helps to improve the performance of target tasks in the target domain.
- *How to transfer:* After determining what knowledge to transfer, learning algorithms or models need to be developed to transfer the knowledge from the source domain or source tasks to the target domain or target tasks.
- *When to transfer:* There are various applications and services in an MEC system. In some cases, the transfer of knowledge can improve system performance, while, in other cases, the transfer can decrease the quality of services or applications. Therefore, whether to transfer from the source domain to the target domain or not needs to be carefully analyzed.

In MEC systems, stochastic task models, heterogeneous MEC servers, and dynamic source data and user capabilities hinder the cooperation between the MEC servers, as well as the deployment of joint MEC tasks across different servers. To mitigate these challenges, transfer learning is believed to be a promising technique for deploying an MEC system across heterogeneous servers. Transfer learning–enabled MEC can be applied in the following scenarios.

- *Multiple computation tasks:* There can be multiple computation tasks and heterogeneous servers in MEC systems. Using transfer learning in the MEC systems can preserve the knowledge learned by some tasks and reuse it in related tasks. Servers in MEC systems can cooperate with each other by sharing and transferring the knowledge they learned from the local network within their coverage. Thus the utility of resources is improved and computation latency is reduced.
- *Computation offloading:* In computation offloading applications, optimal offloading strategies can be determined by learning a policy model with AI algorithms. By using transfer learning, the learned policy model can be used by other MEC servers as the starting point model. The model can be retrained for new MEC systems for a small computation cost. Thus the efficiency of computation offloading can be considerably improved, and energy consumptions can be further reduced.
- *Content caching:* In the content caching scenario, heterogeneous data types and dynamic caching capabilities among different MEC servers are the main issues in caching content across different MEC servers. The popularity of content in

different MEC systems can vary greatly due to the different types of users. Transfer learning can be exploited in heterogeneous MEC systems to mitigate the inaccuracy of caching models. The performance of the caching policy models and computation efficiency is thus improved in transfer learning–enabled MEC systems.

7.2.2.2 Federated Transfer Learning in MEC

Federated learning can connect isolated data and perform joint analyses on the data in way that preserves privacy. However, the issue of model personalization remains unresolved, since all users in federated learning share the same general model. In some cases, the general model might be not applicable to particular users. Moreover, the heterogeneous data distribution of users exacerbates the effective deployment of federated learning. To mitigate these issues, the concept of federated transfer learning emerges as a possible solution. The integration of federated learning and transfer learning broadens the application scope of federal learning. Applications with a small amounts of data or low-quality data can also obtain good machine learning models with the assistance of federated transfer learning.

Federated transfer learning differs from conventional transfer learning in the following aspects.

- *The training architectures are different.* Federated transfer learning is performed on distributed datasets from various users, and the original data are never transmitted to other users. Conventional transfer learning, however, can transmit the original data to a centralized server for training.
- *The machine learning models are trained in different places.* In federated transfer learning, the machine learning models are trained by all distributed users with their local computing resources and datasets. In conventional transfer learning, however, the training of ML models is usually completed by centralized servers.
- *The requirements for data security and privacy are different.* Federated transfer learning aims to protect the security and privacy of user data. In conventional transfer learning, the data face severe risks of leakage.

Research on federated transfer learning is still in its early stage. Y. Liu et al. [131] introduced a new framework, known as federated transfer learning, to improve the performance of machine learning models under a data federation. In [132], H. Yang et al. applied federated transfer learning to image steganalysis and proposed a framework named FedSteg to train a secure personalized distributed model on distributed user data. These limited works explored the integration of federated learning with transfer learning in various areas and provided rough frameworks of federated transfer learning. Federated transfer learning has huge potential in MEC in future networks. MEC can be enabled by federated transfer learning in the following areas.

- *Personalized services:* For future MEC systems, the provision of personalized services for different users is a crucial challenge. Enabled by federated transfer learning, knowledge of, for example, user behaviors and user preferences can be

transferred among different users, based on the trained machine learning models. The quality of service in MEC systems can be considerably improved by the use of federated transfer learning techniques.

- *Super IoT:* The limited data storage capabilities and resources of IoT devices are major obstacles in deploying MEC systems in IoT networks. Federated transfer learning can mitigate the requirement for large amounts of data to train machine learning models. IoT devices can also train ML models with small amounts of data. Moreover, latency in training can be further reduced. The performance of IoT networks and applications can thus be improved.
- *Green communications:* With the increase in numbers of connected devices and applications, the energy costs of MEC systems are becoming a major concern that need to be addressed in future networks. Since machine learning models can be trained with small amounts of data, federated transfer learning decreases the computations for training and the communication overhead for data transmission. The energy cost is reduced and the efficiency is improved, leading to greener communications in future networks.

7.3 MEC in Other Applications

Along with the advancement of beyond 5G technology and the pervasive IoT, MEC techniques have been applied in diverse scenarios to meet intensive computing, processing, and analysis demands for disease prevention, industrial production, and emergency response.

7.3.1 MEC in Pandemics

A pandemic is the spread of an infectious disease across large regions, and it poses serious health risks to huge numbers of people. For instance, the recent COVID-19 pandemic has infected more than 20 million people worldwide and has severely affected the global economy. Early estimates demonstrated that most major economies lost at least 2.4% of their gross domestic product during 2020.

Since pandemic outbreaks are always a surprise and people are largely unprepared to address them, in the early stage, a virus has an extraordinary capacity to spread. It is therefore imperative to establish a pandemic prediction system that can provide valuable and comprehensive information for pre-judging the time, location, scale, and other key characteristics of virus outbreaks.

The efficient operation and precise judgment of the pandemic prediction system are based on comprehensive data collection and complex data analysis. The data are captured by sensing devices that are widely placed in crowded areas, such as bus stations, shopping malls, and schools, and are characterized by large sizes, diverse types, and heterogeneous elements. Processing the captured data with feature extrac-

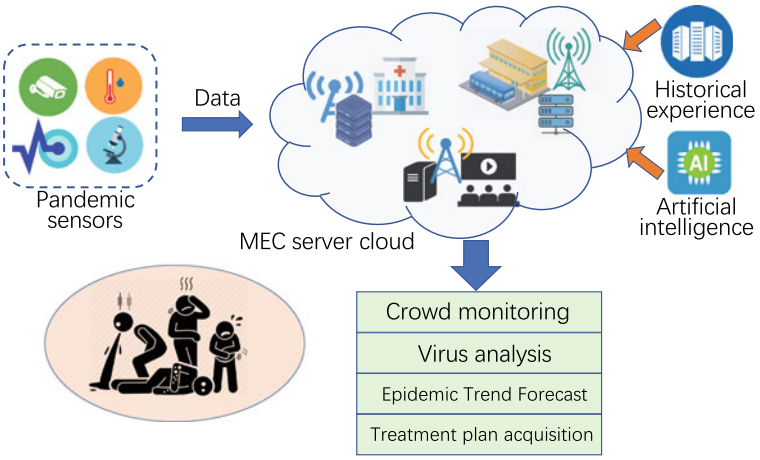

Fig. 7.8 MEC in pandemics

tion and correlation analysis always requires large amounts of computing resources. A traditional approach to meet this demand is to upload the data to remote cloud servers for processing. However, a great deal of communication costs involve yields in this transmission, especially in wireless remote access scenarios. Moreover, the cloud-based processing approach can incur long time delays, which fails to cater to the fast response requirements of applications such as personnel monitoring at a station entrance.

MEC is an appealing paradigm to address this problem. It helps pandemic prediction systems process sensing data in proximity to virus monitoring areas and provides important epidemiological reports about virus spread trends in a short time. MEC servers are equipped on cellular network base stations, Wi-Fi access points, and other nodes that facilitate data transmission and have a stable energy supply. With the continuous enhancement of AI technology, which is capable of revealing hidden issues and correlations from big data, the incorporation of AI and MEC has emerged as a promising approach. Thus, machine learning modules should be deployed to MEC servers to track diseases, predict their growth, and propose coping strategies. For instance, an MEC-empowered deep learning model is suitable for disease classification, while MEC-aided multi-agent learning can be used to predict infection trends in large areas. Moreover, since pandemic prediction and prevention require joint analyses and actions between different departments in multiple regions, a multi-level collaborative MEC service system must be created that shares virus information among MEC servers to better understand and address the pandemic crisis. Figure 7.8 shows a typical framework of MEC techniques applied to pandemic prediction.

In applying MEC in pandemics, open questions still exist. The first involves the privacy protection of the MEC service. In pandemic prediction, the monitoring target is human activities and their physical characteristics, through which malicious users could obtain individuals' private information, such as personal daily activity trajecto-

ries and health status. Recent research has indicated that nontraditional data sources, including social media, Internet search histories, and smartphone data, which are closely related to privacy, are helpful in forecasting pandemic. Consequently, MEC-empowered pandemic management with strict privacy protection is imperative. Furthermore, for the flexible and dynamic detection of pandemics in multiple locations, the pandemic monitoring devices should be lightweight and portable. If the MEC service is integrated into a battery-powered mobile pandemic detection device, the energy efficiency of the data processing will become a key issue for consideration.

7.3.2 MEC in the Industrial IoT (IIoT)

Enabled by IoT technology, industry has witnessed substantial changes in operational efficiency, product quality, and management mechanisms in recent years, and it is continuously evolving toward the IIoT. From the perspective of manufacturers, the proliferation of the IIoT will provide interconnections between large-scale distributed industrial equipment, enable a comprehensive awareness of production environments, and help realize full industrial automation.

Along with the evolution of the IIoT, large amounts of data regarding factory environment status detection, robot device control, and product quality monitoring are being generated and processed. Since modern industrial production is an assembly line operation, any instruction error or behavior lag in the production process will seriously affect the overall manufacturing efficiency. Consequently, the demand for data processing services of high reliability and low latency has increased.

MEC technology, which can facilitate data processing closer to industrial facilities, thereby enabling production managers and equipment controllers to speed up their decision making, has been widely recognized as a promising approach to cater to the demands mentioned. Figure 7.9 shows typical scenarios of MEC application in industrial automation control, logistics transportation management, product quality assurance, and energy scheduling.

To boost production efficiency, remain profitable, and replace expensive human labor with ever-cheaper machines, various manufacturing robots are being widely used in industrial factories. During the operation of the robots, MEC servers work as information processors and control systems that analyze the robot monitoring data from sensors and actuators, while generating control instructions for robotic arm behavior and coping with problems in automated production lines.

Smart logistics have become a key attribute of modern industry. They incorporate autonomous transport vehicles, sensor-driven cargo tracking tools, and online automated sales platforms throughout the whole supply and sale chain. With the aid of MEC technology, unmanned vehicles can achieve more precise and real-time driving control, the transportation status of cargo can be tracked throughout the process, and sales strategies can be optimized in time.

Product quality is the core element of industrial production, and there are many quality inspection methods. With the development of AI, machine learning

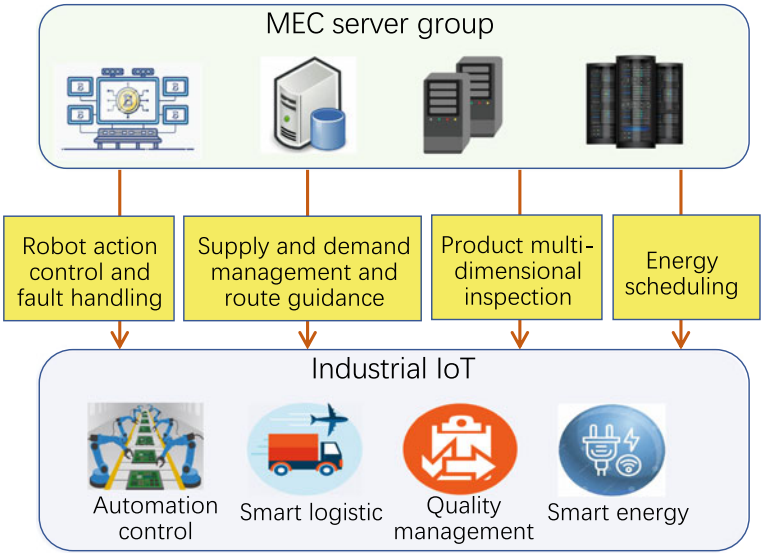

Fig. 7.9 MEC in the IIoT

approaches have been introduced to identify the characteristics of products' dimensions, performance, and stability. The learning process always requires intense data processing and complex model construction. MEC servers that provide sufficient computing capabilities at the site of quality inspection facilities are crucial elements to cater to this requirement.

Industrial manufacturing relies heavily on energy consumption. Among the diversified energy types, electrical energy has been proven to have the most important effects on factory production capacities and costs. With the rise of smart grids, the matching of electricity supply and demand has become flexible, but has also created calculation demands, such as for grid state analysis and user demand trend prediction. MEC is an appealing approach to address this additional demand. Furthermore, besides traditional energy types that harm the environment, renewable energy sources, such as solar, wind, and tidal energy, are beginning to be widely used in industrial production. The time-varying and unstable supply characteristics of renewable energy also require MEC's analytical monitoring and adaptive scheduling. Although MEC technology provides many benefits to the IIoT, some challenges of industrial MEC remain unresolved. For instance, the MEC-empowered IIoT is vulnerable to malicious attacks. Since wireless has been pervasively used in IIoT device-to-device communication, task offloading data can be easily eavesdropped and forged, resulting in the leakage of commercial secrets or production interruptions. In addition, industrial logistic vehicles move throughout large geographical areas and can therefore access heterogeneous MEC servers. The coordination and integration of MEC services is also an unexplored issue.

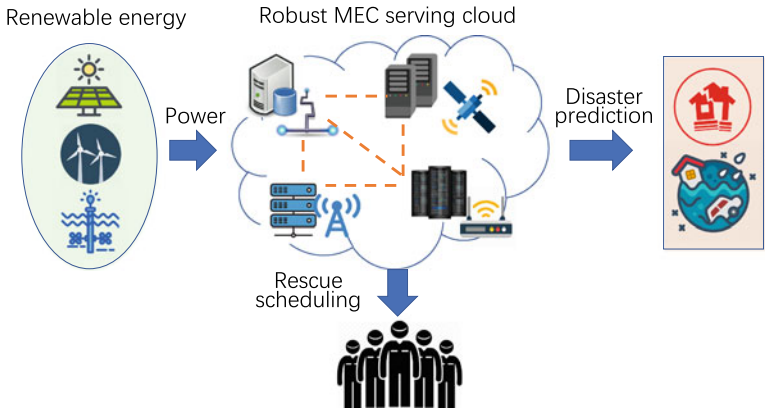

Fig. 7.10 MEC in disaster management

7.3.3 MEC in Disaster Management

Sudden disasters cause serious and widespread human, economic, or environmental losses. To address this issue, disaster management has been proposed for taking some countermeasures and scheduling relief supplies to protect human lives and infrastructures.

To ensure the effective operation of a disaster management system, a large amount of information needs to be processed, which is mainly reflected in two aspects. The first involves the comprehensive analysis of collected environmental data, including meteorological, geological, and hydrological data, to accurately predict possible disasters. On the other hand, after the occurrence of a disaster, progress monitoring and estimations of the status of the disaster relief and supply of materials are required to facilitate the scheduling of rescuers and resources. To meet these information processing demands, servers with powerful computing capabilities should be equipped in the disaster areas. Due to possible damage to communication network facilities and lines caused by the disaster, core cloud servers and remote task offloading are not suitable for providing computing services. MEC's proximity computing service can effectively make up for these shortcomings. However, a single MEC server can also be damaged in a severe disaster; therefore, a group of distributed MEC servers empowered with robustness and survivability is a feasible solution.

Figure 7.10 illustrates the framework of an MEC-empowered disaster management system, where each MEC server pair is connected through several redundant backup communication lines. These links can be wired connections or wireless connections through cellular networks, Wi-Fi, or even satellite networks. The dual backup capability of the servers and communication links greatly improves the robustness and survivability of the entire MEC system. To cope with a potentially unstable power supply in the disaster area, MEC servers can leverage renewable energy and use energy batteries as storage devices to adapt to the time-varying char-

acteristics of wind and solar power. In addition, MEC servers are evolving toward miniaturization and lightweight configurations to meet the portability requirements of a disaster relief operation carried out at multiple locations.

Despite the advantages that edge computing has provided disaster management, key issues remain unexplored in MEC service deployment. A typical problem involves the energy efficiency of MEC servers. Due to the lack of energy supply in disaster areas and the constrained battery power of portable servers, providing powerful computing capabilities at a low energy cost is a critical challenge. Moreover, the effective integration of diversified disaster environment detection networks and heterogeneous rescue systems with MEC services urgently requires further investigation.