RESOURCE ALLOCATION FOR BLOCKCHAIN-BASED MOBILE EDGE COMPUTING WITH DYNAMIC PROGRAMMING

KENING ZHANG

Master of Science in Information Technology

Department of Computing

The Hong Kong Polytechnic University

2022

The Hong Kong Polytechnic University
Department of Computing

Resource Allocation for Blockchain-based Mobile Edge Computing with Dynamic Programming

Kening Zhang

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science

Jul 2022

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Kening Zhang (Name of Student)

Abstract

In recent years, edge computing has been a hot research topic and gaining attraction during information technology development. With the rapidly increasing number of end devices (EDs), the computing paradigm gradually goes from cloud-centric to edge-centric. To this end, the significant attention to edge computing has continuously improved, leading to a decentralized and distributed architecture at the edge of the network. Compared to cloud-centric computing, edge computing can individually and locally manage various resources and services without a centralized third party. However, there are some problems that need addressing in some aspects when EDs interact with edge computing servers (ECSs), such as privacy and security. Due to the above-mentioned reasons, blockchain, an underlying technology of cryptocurrencies, has been widely utilized to solve these issues by taking advantage of properties in security, decentralization, and scalability.

In this dissertation, the detailed description for EDs in the edge-centric condition has been given primarily, and then special features of blockchains are introduced, which fits quite nicely for the edge computing scenarios. Additionally, a general framework is proposed for blockchain-enabled mobile edge computing systems, and it specifically designates the whole workflow progressively among EDs, ECSs, memorizers, verifiers, and smart contracts. Furthermore, how the smart contract in the blockchain assigns the appropriate ECS to process the segments of corresponding ED has been modeled with the method of the semi-Markov decision process (SMDP),

which is solved by a state-of-the-art machine learning algorithm for maximizing the cumulative reward. Then, integrating mobile edge computing and blockchains is suggested to research further to guarantee privacy and security. Finally, the performance of the experiment and simulation is presented.

Acknowledgments

There are full of difficulties during my master degree period. However, the more I set back, the braver I will get. Sometimes I can't avoid feeling sad, but everyone has to experience a lot when pursuing the study career, as far as I am concerned. Although there are several hardships needing to overcome, I still keep my confidence up. The most important reason is that I have received many helps from my friends and people around me.

First and foremost, I would appreciate the supervisor, Prof. Jiannong Cao, who gave the opportunity to attend the research group as a research assistant. Before joining the research group, he recommended his Ph.D. students to me to comprehend deeply for various fields, and allowed me to choose one direction that I'm interested in. It was my great luck to meet him, because not only did he teach me how to research and study, but also asked me to develop my own literary literacy, such as reading books every week. I have become a good thinker and gotten a greater love of life under his influence.

Additionally, I cannot succeed without the help of Dr. Shan Jiang, Dr. Yanni Yang, Dr. Yu Yang, and Dr. Ruosong Yang. They introduced their own areas of expertise to me, especially Dr. Shan Jiang. He supervised me to finish my dissertation during this period. He taught me how to write the academic report, give an academic presentation, help me practice my presentation and so on. Besides, he kept urging me to finish the dissertation in time and developed several deadlines for me, so that

I could take control of my time. Meanwhile, I want to express my sincere thanks to each institution in PolyU, such as Student Hall and SEN support. Without the continuous help of them, I cannot finish my study career completely.

Last but not the least, my family and parents are the most important people in my life, and they always support me unconditionally. My best friends also company with me when I'm lonely. Everyone I met give me a great power and had become the light sprinkling all over my life.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background & Motivation

With the progressive development in the field of Internet of Things (IoT), there are more and more physical devices connected with each other [1]. Due to the increasing of massive data generated by the end devices (EDs) of IoT, bandwidth, computing capability and other resources will be required intensely. Therefore, data analysis performs a statistically significant importance in EDs [2], which needs support and implementation of cloud computing. However, although there are infinite capacities in current computation, the cloud-centric computing expresses several problematic topics which need discussing such as latency, transmission cost and congestions, because these issues have already limited the development of cloud computing [3, 4].

In addition, the fact that centralized computing paradigm in actually shows some negative aspects due to a number of reasons [5, 6, 7]. Primarily, privacy cannot be guaranteed during this architecture, for example, the sensitive and intimate data in EDs, such as coordinates, IP, chatting records, surveillance information, etc., are exposed ordinarily under the services of center-based cloud server, such as location services, identification services, recommended services, etc. Second, the clouds belong

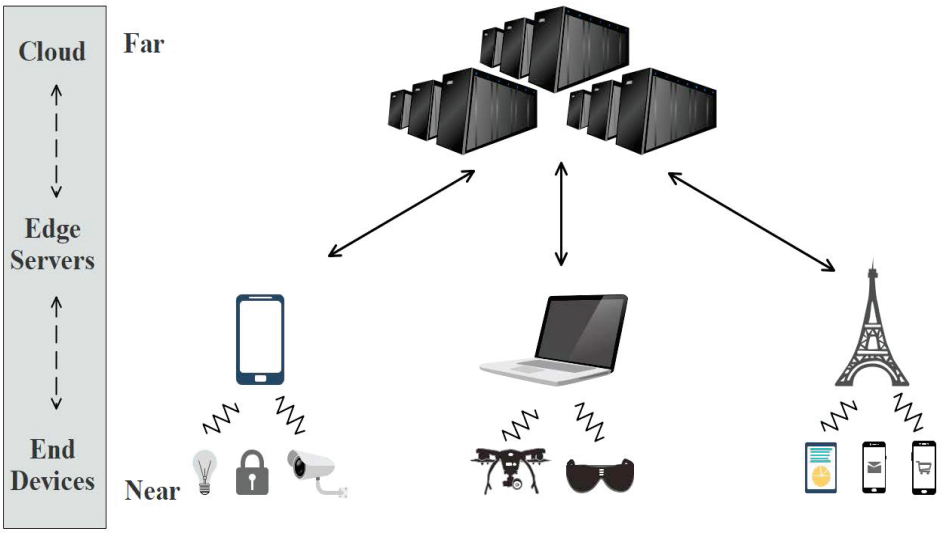

Figure 1.1: Edge centric computing architecture

to the third party, which makes applications completely lose users' control and this has created an untrustworthy situation [8]. Thirdly, with the advent of more advanced individual devices, plenty of computational resources, capability, and storage are not made the most of owing to the centralization of network. At last, single point of failure in the centralized computing results in the poor scalability of the system, so that this structure is not suitable for handling innumerable data service requests sent by an exponentially increasing number of EDs [9].

As a result, the recent trend is towards a decentralized computing paradigm where edge-centric computing, such as mist and fog computing, has been proposed for use in EDs of IoT scenarios [10, 11, 12]. Different from centralized architecture, edge-centric computing converts distinct services, resources, applications and so on to cyber margins that become nearer to the EDs, and the periphery can supply services of short delay, quick reaction and position awareness [13, 14, 15]. Moreover, diverse resources and services of EDs on the edge can be individually and locally regulated without the third-part institutions in terms of decentralized framework [9], henceforth, more and more services and applications can be implemented progressively in

characteristics of innovation, convenience and practicality. For IoT applications in the practical scenarios, short delay, quick reaction and privacy protection services are highly demanded [16], and edge-centric computing meet these requirements preferably well. In this architecture, individual EDs owned by users play quite significant roles in edge computing servers (ECSs), and their resources can be connected and shared with other devices via the network.

However, there are two vital issues for EDs, i.e., privacy and security when the system disposes transactions between EDs and ECSs, since credible and trusted relationships have not formed. To address these problems, blockchains, the state-of-the-art technology, as immutable public ledgers become a viable method currently. Through this technique, transactions can be cryptographically verified during the procedure because the secret key and consensus mechanism in blockchains promote efficiency of collective stewardship in services and resources. In this respect, transactions with blockchain technique are safety due to its irreversibility traceability [17].

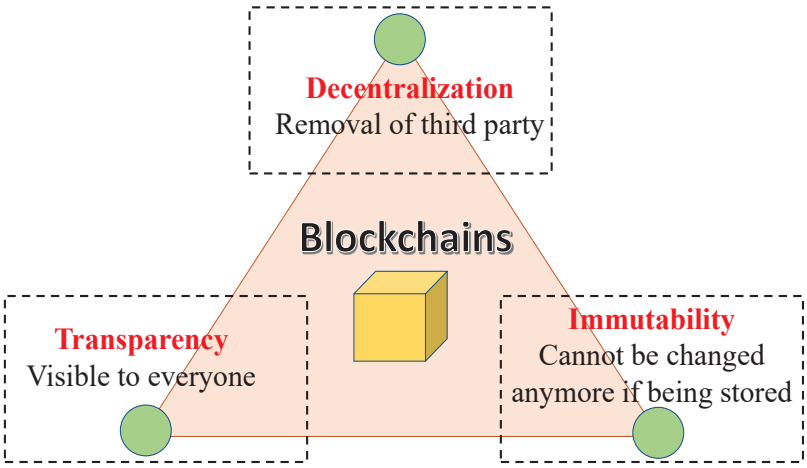

Figure 1.2: Distinctive features of blockchain

Blockchain, as the basic technique applied to the digital cryptocurrency has a significant development during recent years from the tech giants to manufacturers, which has also been utilized in IoT scenarios particularly [18, 19, 20]. Blockchain technology has been widely adopted in education [21], smart warehousing [22], data

sharing [23], etc. For example, there is a prestigious app platform named Ethereum, having already realized Turing-complete language with blockchain, which makes all categories of program codes permitted to executed on blockchains with autonomy and automaticity. To be specific, the autonomy means that there is a smart contract activated by blockchains since the contract can be automatically initiated when meeting specific conditions. For Ethereum, a customizable and arbitrary "Blockchain as a service" is provided by Monax Industry with the beginning of an original Ethereum fork, resulting in generating own rules and protocols created by clients in blockchains [24].

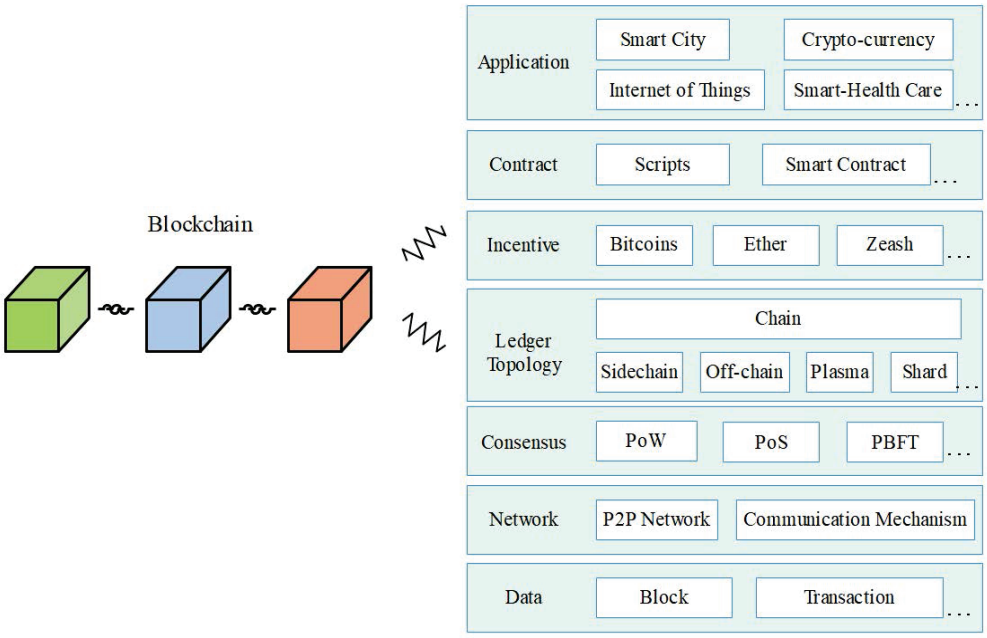

Figure 1.3: Component of Blockchain

To articulate edge computing and blockchain distinctly, a toy example is presented as figure 1.4. Someone loves to eat the "octopus", who is actually a living edge computing master. Octopus own the highest IQ of invertebrates, and they are very dexterous and rapid in movements when hunting. Their wrist and foot are highly cooperative which will be never tangled and knotted, and this is because the huge number of neurons are 60% distributed in the eight legs, and only 40% in the

brain. The structure is "multiple cerebellum + a brain". This unique distributed computational structure gives the octopus a superior IQ among invertebrates. This is actually the concept of edge computing. The data collected is processed close to the edge, rather than being uploaded to a cloud computing center, thereby improving real-time, efficiency and reducing the processing pressure on the center.

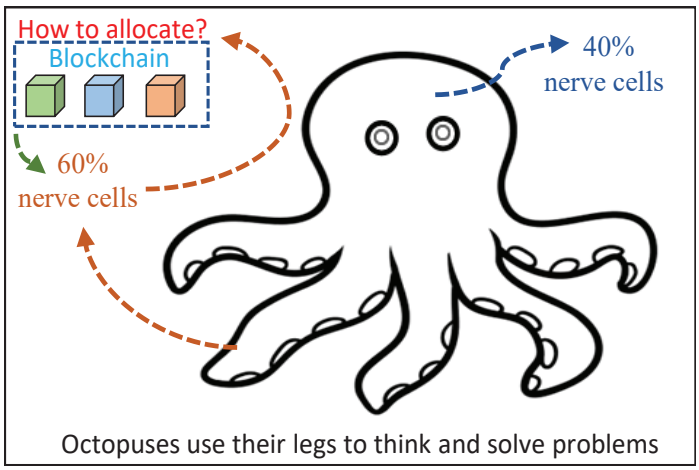

Figure 1.4: A toy example for edge computing and blockchain

Therefore, the relationship between edge computing and blockchain is naturally self-explanatory. Blockchain is also a distributed network technology, not only for storage and community governance, but also for computing, so the technologys of blockchain and edge computing are in common a lot. Naturally, blockchain encompasses a much broader aspect, not just in computing and storage. There are already a number of blockchain projects mentioning that the integration of edge computing and blockchain is quite vital.

It is conceivable that in the future, with the development of the IoT and the exponential growth of smart devices, the edge of the network will generate a huge amount of data. If all these data are processed by a central computing platform, it is difficult to avoid problems in terms of agility, real-time, security and privacy. But with edge computing, massive amounts of data can be processed favorably, and a large

number of devices can work together efficiently, which benefits to solve problems and improve the "intelligence" of computing. To this end, how to allocate and manage such huge distributed computing resources needs ideas of blockchain in consensus mechanism, smart contract, incentive system and so on.

1.2 Research Framework

The main contributions of this dissertation are as follows:

- For edge-centric computing architecture, introduce advantages detailly compared with cloud-centric computing and explain the security and privacy issues, benefits of integrating blockchains and related features in the corresponding network scenarios.

- Propose a blockchain-enabled framework for edge computing scenarios, which designates specifically the whole workflow among EDs, ECSs, memorizer, verifier and smart contract from the data service request created by ED to the service delivery completed by ECS, and the system records transactions on the blockchain finally.

- Design the smart contract in the blockchain assigns the appropriate ECS to process the segments of corresponding ED with method of semi-Markov decision process (SMDP), which is solved by the state-of-the-art machine learning algorithm for maximizing the cumulative reward for computing resources allocation problem. Specifically, diverse Quality of Service (Qos) from various EDs can be distinguished in this proposed scheme, which result in low latency. For the solution, how artificial intelligence technology (AIT) can be combined with blockchains and its merits are expressed clearly.

- Experiment results are explained finally to present the performance of the al-

gorithm in the proposed blockchain-based framework for addressing resource allocation problem.

1.3 Dissertation Organization

The research mainly focuses on resource allocation for blockchain-based mobile edge computing in this dissertation. A blockchain-based mobile edge computing framework is designed integrally, which includes all the steps one transaction will follow in the process. Additionally, a smart contract is proposed to allocate the specific ECS to proceed and compute related data segment sent from EDs. In order to address the proposed issue of the system model, the reinforcement learning method was adopted in this scenario. There are six chapters in this dissertation, and it is organized as follows:

- In chapter 1, cloud-centric, edge-centric computing paradigm, blockchain technology, and the main content of the work related to this dissertation are introduced in general.
- In chapter 2, related works in the academic world are classified into three categories: storage, computation and network, which are discussed detailly.
- In chapter 3, the system model is proposed, which mainly includes system architecture, the framework of blockchain-based edge computing network.
- In chapter 4, the problem is formulated, which includes system state, system action, system dynamics, and policy and reward function are defined detailly. The RL algorithm is adopted to address the problem proposed before, including several formula derivations.
- In Chapter 5, experiment and simulation results are implemented and discussed detailly.

- Chapter 6 summarizes the work in the dissertation, pointing out the shortcomings of the existing work and the outlook for on blockchain and edge computing in the future.

Chapter 2

Related Work

In this chapter, several related works in the academic world are introduced, which is mainly about the integration of blockchain and edge computing. In order to comprehend clearly, there are three categories articulated for the existing work: storage, computation and network.

2.1 Storage

Edge-centric computing makes the data closer to the users and can reduce the stress of central servers, compared with the cloud-centric paradigm [25]. However, there are issues about data integrity, security, etc. in this kind of distributed system because of the isolated storage locations [26]. The characteristics of P2P data storage mechanisms, security and validity of blockchain [27] can be addressed for above-mentioned problems in the integrated system of blockchain and edge computing.

B. Liu et al. [28] proposed a blockchain-based framework, which provides the data consumers and owners the verification to keep the data integrity. In [29], a decentralized storage network has been proposed, which made cloud storage into an algorithmic market with the consensus Proof-of-Replication (PoRep) and Proof-of-

Spacetime (PoSt). G. Zyskind et al. [30] used blockchains as access-control managers in order to keep the data safe and control these by themselves. M. Klems et al. [31] proposed decentralized service marketplaces based on Ethereum blockchain. In [32], scalable blockchain databases have been unitized for edge computing. In [33] [34], a global naming and storage system was proposed with one blockchain layer, peer network layer and data-storage layer. M. S. Ali et al. [35] proposed a modular consortium architecture for data privacy in IoT.

2.2 Computation

For edge computing, one single node needs to allocate tasks to other nodes due to lack of resources and computing capabilities compared with cloud-centric computing. In this kind of decentralized paradigm, the system runs on blockchains for verification and incentive, which is most prospecting method.

In [36], a blockchain-based (sidechain) fully distributed cloud was proposed with the consensus of Proof-of-Contribution (PoC). R. Kumaresan et al. [37] presented a model of incentivizing correct computations for verification, fairness and security. N. C. Luong et al. [38] used an economic approach to manage the resource and combine the mobile blockchain and edge computing. N. C. Luong et al. [39] utilized a deep learning algorithm to allocate edge resources with an optimal auction. Y. Jiao et al. [40] defined a hash power function and maximize the social welfare for mobile blockchain-based edge computing. Z. Xiong et al. [41] jointly considered the profit and miners utilities by proposing a two-stage Stackelberg game for mobile blockchain-based edge computing. M. Andrychowicz et al. [42] proposed timed commitments version of Bitcoin for Multiparty Computation (MPC) to keep fair in certain protocols. A. Kiayias et al. [43] gave a model of MPC by using compensation, which realized the fairness. R. Kumaresan et al. [44] utilized penalties to make the protocols more efficient for MPC. I. Bentov et al. [45] proposed a stateful contracts which

made interaction richer between cryptocurrency and computation. M. von Maltitz et al. [46] flexed secure MPC for IoT environments. L. Zhou et al. [47] proposed a blockchain-based threshold IoT system named BeeKeeper with a threshold secure MPC (TSMPC) protocol. In [48, 49, 50], a P2P cloud computing/edge computing was supported for cloud and highspeed local networks. In [51, 52], an open-source cryptocurrency with a decentralized manner upon the BOINC platform was proposed. L. Luu et al. [53] formalized the security model and consensus in the aspect of latency and accuracy. In [54, 55], a dispute resolution layer and a financial incentive layer was presented for verification in blockchains.

2.3 Network

For blockchain-based edge computing network, the most important function is to keep secure in the communication process. In this architecture, two devices are connected and communicated with each other via the central nodes in the edge layers, where the blockchain communication is also set [56]. Consequently, the integration of blockchain and edge computing will keep the network more reliable and efficient.

P. K. Sharma et al. [57] used blockchain-based flow rule to update the network architecture based SDN. M. Samaniego et al. [58] proposed a Multichain blockchain cluster in a Fog network to reduce the response time. M. A. Salahuddin. et al. [59] embraced fog and cloud computing, blockchain and message brokers in a softwarized infrastructure. In [60], a blockchain-based distributed cloud architecture was proposed to reduce the delay and response time. C. Li et al. [56] proposed a blockchain-enabled multi-layer IoT model to secure the network. M. Samaniego et al. [61] reduced the communication delay with Cloud-hosted permission-based blockchains. In [62, 63], a blockchain protocol for autonomy was proposed to support P2P messaging and distributed file sharing. A. Stanciu et al. [64] proposed a blockchain-based distributed control system for edge computing deployed by docker containers. A. Dorri et al. [65]

proposed a blockchain-based smart home framework to prevent DDOS attack and linking attack, and in [66] also formulated a hierarchical blockchain-based architecture for IoT. A. Lei et al. [67] proposed a blockchain-based distributed key management for the efficiency and robustness. C. Tselios et al. [68] presented an overview of security issues of SDN based IoT clouds and a blockchain-based layer. M. Samaniego et al. [69] utilized virtual resources and a Multichain blockchain to provision IoT services on edge hosts. N. Herbaut et al. [70] provided the content sessions quickly on the blockchain. D. B. Rawat et al. [71] leveraged the blockchain protocol to avoid the double spending of same wireless resources.

Table 2.1: A Survey of Integrated Blockchain and Edge Computing Systems

Category	**Ref.**	**Contributions**	**Applications**
Storage	[28]	A blockchain-based framework	IoT Network
	[29]	A decentralized storage network	Internet
	[30]	A blockchain-based system	Storage Network
	[31]	Blockchain-based marketplaces	Service System
	[32]	A scalable blockchain database	Datebase
	[33, 34]	A global naming system	Internet
	[35]	IoT data privacy	IoT Network
Computation	[36]	A distributed cloud infrastructure	Internet
	[37]	Incentivize correct computations	Internet
	[38]	Mobile blockchains	IoT Network
	[39]	Blockchain-based edge computing	IoT Network
	[40]	Mobile blockchains	IoT Network
	[41]	Mobile blockchains	IoT Network
	[42]	Construct timed commitments	Internet

Continued on next page

Table 2.1 – continued from previous page

Category	Ref.	Contributions	Applications
	[43]	A formal model of secure MPC	Internet
	[44]	Secure MPC	Internet
	[45]	Amortized secure MPC	Internet
	[46]	Flex secure MPC	Internet
	[47]	A system: BeeKeeper	Internet
	[48, 49, 50]	Combine BOINC with networks	Internet
	[51, 52]	An open source cryptocurrency	Internet
	[53]	Formalize the security model	Internet
	[54, 55]	A verification method	Internet
Network	[57]	Detect attacks in IoT networks	IoT Network
	[58]	Reduce the response time	IoT Network
	[59]	A softwarized infrastructure	Smart Healthcare
	[60]	Reduce delay time	IoT Network
	[56]	Secure IoT network	IoT Network
	[61]	Reduce delay time	IoT Network
	[62, 63]	A blockchain protocol	IoT Network
	[64]	A Blockchain-based system	Control Systems
	[65]	Prevent DDOS attack	Smart Home
	[66]	A hierarchical architecture	Smart Home
	[67]	A dynamic scheme	Control Systems
	[68]	Prevent the attacks in SDN	IoT Network
	[69]	Provide IoT services	IoT Network
	[70]	Provision the content sessions	Internet
	[71]	Wireless network virtualization	IoT Network

Chapter 3

System Model

3.1 System Architecture

In this dissertation, the system architecture is presented in figure 3.1, which includes two layers: 1) the end device layer; and 2) the edge computing server layer.

- The ED layer: The end devices, such as numerous cellphones, sensors, VRs, laptops, smart cars, etc., detect and collect different data from the external environments. When each ED handles and sends the interrelated data to the edge computing layer, the smart contract will assign specific ECS for computing. If required, the output data after being finished by ECS can be given back to the corresponding ED or transmitted to the cloud for advanced data processing or prolonged storage. For this procedure, it's only considered that the analysis results should be returned to the EDs in a timely fashion, i.e., real-time interaction.

- The ECS layer: There are multiple ECSs which can provide computing services for data. In distributed computing system, how to schedule tasks are the extremely significant standard in view of the fact that the efficiency is essentially

a representation of the capabilities of the applications [60]. Therefore, when there is a data computing requested by an ED, the algorithm will automatically assign a most appropriate ECS for the computing service on the basis of the current system state (e.g., workload of each ECS) and the QoS requirement (e.g., the service delay of each ED). Accidentally, ECSs will offload data to the cloud servers at the price of increasing spare consumption and related latency, if there are not enough computing resources.

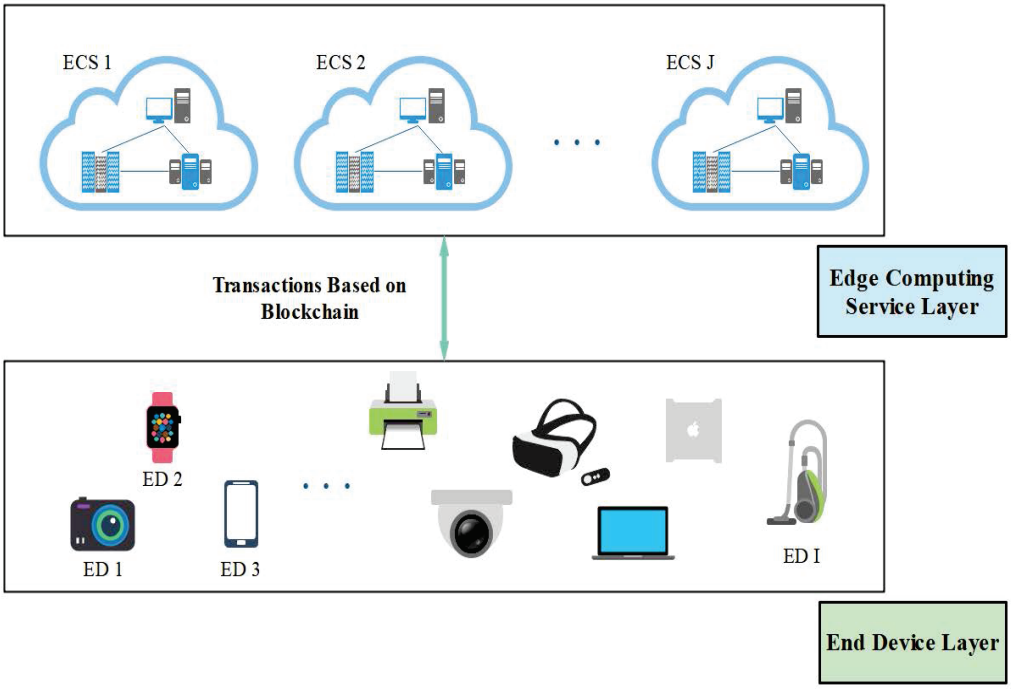

Figure 3.1: Overview of The System Architecture

3.2 Framework of Blockchain-based Edge Computing network

In this part, the whole framework of blockchain-enabled edge computing network is proposed in detail, which includes six stages on how the transactions are processed

step by step from the EDs to the ECSs based on blockchain technique.

There is a particular kind of crypto token supposed in the architecture for circulating in its entirety. Requesting computing data from EDs needs to give payment for the crypto tokens, correspondingly, ECSs can make tokens if they use resources, such as bandwidth, to make a contribution for processing tasks. The crypto token during the framework will get confirmation if it has already been transferred, which has the same security as the underlying PoS or PoW in blockchains. In addition, the underlying blockchain platform offers consensus mechanisms in the architecture, such as Consensus mechanisms. To encourage taking part in verification and computation, dispensing newly minted tokens has been a supernumerary measure. However, the main research of this architecture focuses on allocation of ECS, so above content and relevant issues are beyond the scope of this dissertation.

For data transmission, the core cell of EDs steaming data is regarded as a segment, which means steaming data will be split into several segments after being received from EDs. Then, every segment is assigned a series of sequences for recognizing them in the correct order [72]. Specifically, the integrity of related data can be confirmed through hash of each segment. In the proposed framework, this type of transmission has been adopted for payment and computation of the blockchain.

The complete process of each step has been demonstrated in below figure, where the computing requirement is sent to the blockchain by ED i. The procedure is explained detailly as figure 3.2.

- ED i claims a request for computing data segment on the blockchain and several crypto tokens are deposited in escrow.
- After collecting the request, there is a concrete ECS j which will be allocated to ED i via the smart contract. In blockchain technology, there are plenty of criteria for selecting the specific computing server in various scenarios. For instance, PoS is delegated to select a computing node for executing transcoding

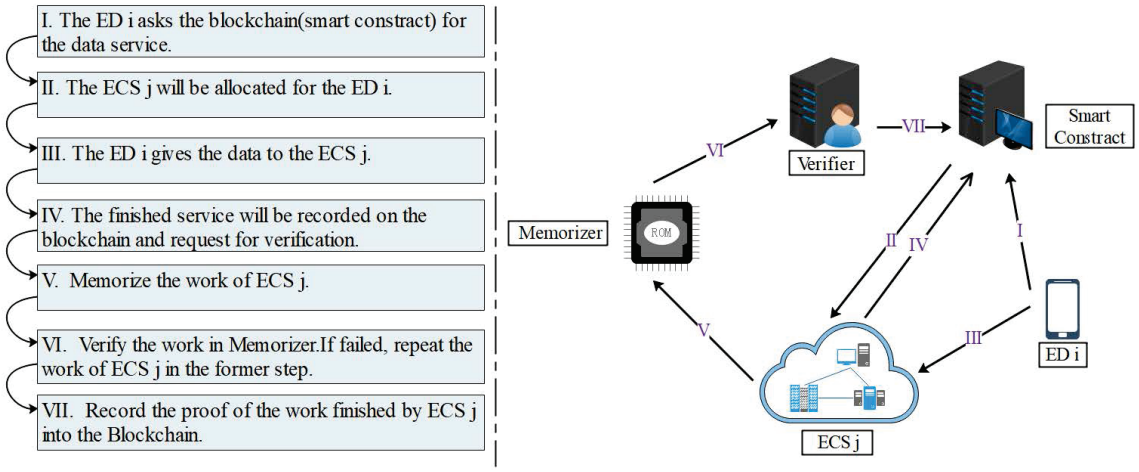

Figure 3.2: Procedure of A Transaction

work in [73]. PoS means proof of stake, in which the node having the most cumulative stakes is chosen as the computing role. In this dissertation, the deep reinforcement learning algorithm is utilized to choose the most appropriate ECS.

- Upon selecting the specific ECS j, it will claim an acceptance of permission on network, and afterwards ED i can send the segments to ECS j with the signature of data.

- When the computation is finally completed, the ECS j transmits the notification to the blockchain, and there is a transaction including the relative receipt which will be added into the Merkle tree in one block simultaneously. For receipt, it contains the hash of the input and output segment, the signature of input data from ED i and output data from ECS j, and segment identification. Then, ED i verifies process and result of its computation.

- In the meanwhile, ECS j needs to send the receipt of segment data payload to the memorizer for being further verified, and the data keep being stocked from the beginning of process to the end.

- There is a verifier working as a certain protocol during the procedure, which randomly selects partial data from the memorizer for validating to guarantee the rightness of computation given by ECSs. When the receipt is proved to be correct, a declaration will be transmitted to the smart contract from verifier. Correspondingly, several tokens will be forwarded from the ED's account to the ECS after completing the task. On the contrary, the ECS is punished and deduct a few tokens once the verification is improper, and then the process will be re-executed again by other ECSs.

- Finally, the validated transaction of a block will be logged in the blockchain.

In the following, the smart contract, i.e., the second step is discussed detailly because it's actually the key process of this dissertation, which is mainly about how to allocate the most suitable ECS to compute tasks.

3.3 Queuing Theory

Queuing theory is the mathematical theory, which studies the phenomenon of stochastic aggregation and the working process of stochastic service systems. It is also known as the theory of stochastic service systems, and is a branch of operations research. Queuing theory is the basic theory for the study of the amount of communication information in computer communication networks and computer systems. The quantitative study of problems in information system communication will often be discussed under the help of queuing theory.

The queuing system is also known as the service system. A service system consists of a service organisation and a service recipient (customer). The moment of arrival of the service recipient and the time when it is served (i.e. the time occupied by the service system) are random.

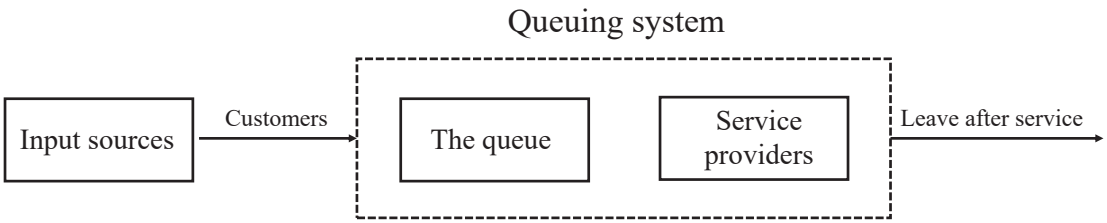

Figure 3.3: Task procedure

3.3.1 Input Process

The input process examines the pattern of customer arrivals in a service system. It can be described in terms of the number of customer arrivals within a certain time period or the interval between the successive arrivals of two customers, which can be generally divided into deterministic type and random type. For example, the arrival of parts on a production line at specified intervals, regular shuttle buses and flights are all deterministic inputs. A random input is one in which the number of arrivals $n(t)$ at time t follows a certain random distribution. Generally, the Poisson distribution is obeyed in this proposed system model, and the probability of n customers (edge nodes) arriving at time t is

$$P_n(t) = \frac{e^{-\lambda t}(\lambda t)^n}{n!}, n = 1, 2, ... \tag{1}$$

The interval T between successive arrivals follows a negative exponential distribution, i.e. $P(T \leq t) = 1 - e^{-\lambda t}$, where λ is the expectation of the number of arrivals per unit time, called the average arrival rate, and $1/\lambda$ is the average interval.

3.3.2 Queuing Systems

There are three types of queuing systems: waiting system, loss system and mixed system. The waiting system means that a new task arrives and waits in line when all service establishments are occupied. In the waiting system, the order in which

services are performed for tasks can be first come first served (FCFS), last come first served (LCFS), randomized service, service with priority (e.g. hospitals receiving emergency patients). If a task arrives and leaves immediately upon seeing that the service establishment is not available, this is named the loss system. When there is limited space left for tasks to wait in queues, extra tasks must leave the system, and this is called the mixed system.

In this proposed system, the Poisson distribution is adopted for the arrival of tasks from edge nodes, and the service rule in edge computing servers is FCFS.

3.4 System Delay Model

Assume that there are I EDs given as d_i where $i \in \{1,\ 2, \ldots,\ I\}$ and J ECSs credited as h_j where $j \in \{1,\ 2, \ldots,\ J\}$ respectively. Although various EDs have their own principles for computation, which can be generally evaluated with the service delay time (SDT). For instance, several EDs may adopt the less cost of resources compared with the computation time, while other EDs prefer minimizing SDT even if there is more cost. In order to meet the requirements of the rule of service for ED d_i, there is a threshold labeled as δ_i^{lm} restricting the SDT a certain range. To be specific, $T_{i,j}$ presents the total delay time for processing one segment from ED d_i to ECS h_j, which keep meeting the inequality ($T_{i,j}\ \leq\ \delta_i^{lm}$).

In this dissertation, the total delay time for serving one segment from ED d_i to ECS h_j is composed of two elements below.

$$T_{i,j} = T_{i,j}^{comp} + T_{i,j}^{net} \tag{2}$$

In the above formula, $T_{i,j}^{comp}$ represents the computation delay, which means the time spent on computing one segment in ECS h_j. With regard to $T_{i,j}^{net}$, the total transmission delay is included from the ED to the designated ECS and return time.

For the above two components of SDT, $T_{i,j}^{comp}$ and $T_{i,j}^{net}$ are expressed respectively in the following parts.

3.4.1 Edge Computing Server Delay

Suppose that the arriving segments of data from ED d_i to ECS h_j satisfy the Poisson distribution individually at the rate of φ_i, $i = 1, 2, \ldots, I$. Similarly, every ECS has its own computing capability, here v_j is used to represent the service rate of ECS h_j.

After smart contract allocates a specific ECS h_j to compute the data segment from ED d_i, the delay $T_{i,j}^{comp}$ can be further divided into two components: 1) the computation time in the ECS h_j, labeled as D_c^j and 2) the time needing to wait in the queue, which is denoted as D_w^j. So that the computation delay $T_{i,j}^{comp}$ can be regarded as

$$T_{i,j}^{comp} = D_c^j + D_w^j \tag{3}$$

For computing time D_c^j in the server, it is assumed that the time of computing one data segment in ECS h_j can be calculated by the equation $1/v_j$. Therefore, D_c^j can be represented as

$$D_c^j = \frac{1}{v_j} \tag{4}$$

For the waiting time D_w^j in the queue, there are also two parts in it: 1) the total data segments, labeled as D_{wq}^j, which are not computed in the queue and 2) the remaining process time of the present task, denoted as D_{wr}^j which is computing. Then, D_w^j can be written as

$$D_w^j = D_{wq}^j + D_{wr}^j \tag{5}$$

Suppose that there are n_j data segments in the current ECS h_j in total, which is practically dynamic during the whole process. In this mechanism, first-come, first-served discipline is adopted, which indicates that the ECS h_j will extract and process one segment at a time from the queue. After finishing computation, the number of data segments n_j in the ECS h_j will be reduced by one, so that the time D_{wq}^j can be updated as

$$D_{wq}^j = \frac{n_j}{v_j} \tag{6}$$

THE REMAINING PROCESS TIME

For the remaining process time of the present task D_{wr}^j, the derivation procedure can be expressed as follows:

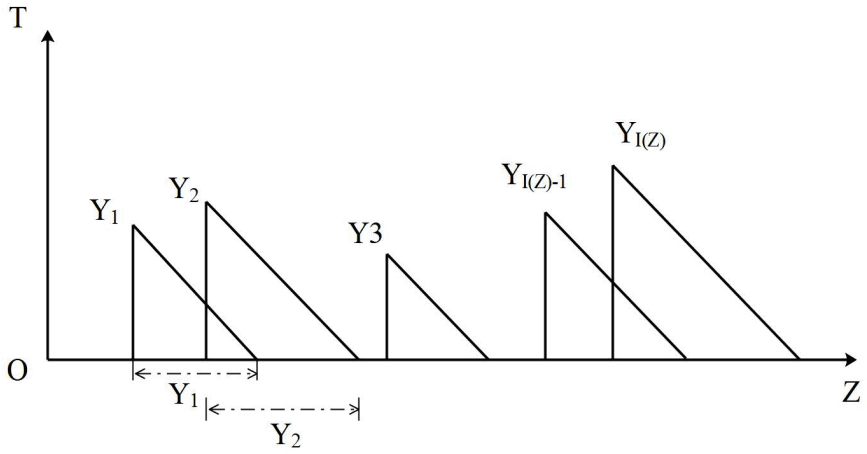

Figure 3.4: Task procedure

Suppose that there is a period of time that belongs to $[0,\ Z]$, and denote $T(z),\ z \in [0,\ Z]$. There are $I(Z)$ computing tasks arrived during $[0,\ Z]$, and Y_i is process time of each $I(Z),\ \ i \in 1, 2, \ldots, I(Z)$. This is demonstrated in the figure

5, and then, the remaining time can be defined as

$$\begin{aligned} D_{wr}^{j} &= \frac{1}{Z}\int_{0}^{Z} T\left(z\right) dz \\ &= \frac{1}{Z}\sum_{i=1}^{I(Z)} \frac{1}{2}\, Y_i^2 \\ &= \frac{1}{Z}\frac{1}{2}\frac{I\left(Z\right)}{I\left(Z\right)}\sum_{i=1}^{I(Z)} Y_i^2 \\ &= \frac{I\left(Z\right)}{2Z}\frac{1}{I\left(Z\right)}\sum_{i=1}^{I(Z)} Y_i^2 \end{aligned} \tag{7}$$

In this formula, $I\left(Z\right)/Z$ means the average arrival rate of tasks sent by EDs, which can be represented by $\bar{\varphi}$, and $\bar{\varphi} = (\varphi_1 + \varphi_2 \ldots + \varphi_I)/I$. Y_i is process time and can be regarded as $1/v_j$. So, the equation will be upgraded as [74]

$$D_{wr}^{j} = \frac{1}{2}\bar{\varphi}\frac{1}{v_j^2} \tag{8}$$

Therefore, the complete computing delay time can be updated as [33]

$$\begin{aligned} T_{i,j}^{comp} &= D_{c}^{j} + D_{w}^{j} \\ &= D_{c}^{j} + D_{wq}^{j} + D_{wr}^{j} \\ &= \frac{1}{v_j} + \frac{n_j}{v_j} + \frac{1}{2}\bar{\varphi}\frac{1}{v_j^2} \\ &= \frac{n_j + 1}{v_j} + \frac{1}{2}\bar{\varphi}\frac{1}{v_j^2} \end{aligned} \tag{9}$$

3.4.2 Network Delay

As a usual, the delay of data transmitted from ECSs back to EDs is ignored in the model due to the small size of data in general. Therefore, only the network delay of sending data segments from EDs to ECSs is considerate. However, there are plenty of factors which can influence the delay time, such as the channel conditions, the

distance, degree of crowdedness and so on. Practically, the network delay will usually be estimated by unitizing training data transmitted from EDs to ECSs [41]. Then, in [75], finite-state Markov channels (FSMCs) are utilized to describe the realistic time-varying channels in this situation. There is a parameter r_i^j proposed for representing the received signal-to-noise ratio (SNR). In the meanwhile, suppose two parameters b_i and s_i represent the bandwidth and one segment size respectively. Then, following the Shannon equation, $T_{i,j}^{net}$ can be updated as

$$T_{i,j}^{net} = \frac{s_i}{b_i \log\left(1 + r_i^j\right)} \tag{10}$$

Finally, the total delay time of the framework from ED d_i to ECS h_j and the corresponding inequality that needs satisfying can be regarded as

$$\begin{aligned} T_{i,j} &= T_{i,j}^{comp} + T_{i,j}^{net} \\ &= \frac{n_j + 1}{v_j} + \frac{1}{2}\bar{\varphi}\frac{1}{v_j^2} \ + \frac{s_i}{b_i \log\left(1 + r_i^j\right)} \leq \delta_i^{lm} \end{aligned} \tag{11}$$

In particular, there is no verification time during the total delay time. The computation results will be transmitted at once to the ECS h_j or the cloud servers if the task is finished due to the characteristic of real-time interactive edge computing. At the same time, the memorizer will accept the computation results for further verification later. After that, ECS h_j will get tokens escrowed on the blockchain by ED d_i before, and the logging is written onto the blockchain. However, ECS h_j will not earn corresponding crypto tokens and a certain percentage of punishment is given to it, if the result is incorrect after verification. In fact, it's usually not possible to fail the verification because there is no benefit at all for the ECSs to have malicious behaviors.

Chapter 4

Solution for the Proposed Model

This chapter is mainly focused on how the system allocates the most appropriate ECS h_j to compute the segment sent from ED d_i, which means the ECS h_j should not only meet the time delay inequality but also maximize the total income. A deep reinforcement learning method [76, 77] is utilized not a value-based but a policy gradient method, which takes advantage of the typical actor-critic algorithm. By getting the optimal policy forthrightly, the system can respond more quickly, especially in intricate and complicated, which is certified as a better, simpler and more robust solution.

4.1 Problem Formulation

To accurately describe the problem, a continuous-time Markov decision process is utilized to address this. There are two typical types of the continuous-time Markov decision processes: 1) continuous-time jump Markov decision processes and 2) actions can be chosen continuously in time. For the first type, when the decisions are allowed to made at a certain time, the process can be regarded as semi-Markov or even discrete-time Markov decision [78].

Similarly, a toy example is given for articulating the SMDP clearly and detailly in figure 4.1. There are frogs jumping on lotus leaves. Different states are represented on each lotus leaf, and jumping from one lotus leaf to another represents a state transition. The transfer process depends only on the lotus leaf where the frog is now located and it is independent of the lotus leaf where the frog has previously stayed. If one considers only the sequence of moments when the frog jumps (i.e., it is only the moment, not for a while), the process is a discrete-time Markov process. If the frog is considered to have spent some time on the lotus leaf and this time is exponentially distributed, then the process is a continuous time Markov process. If the staying time is non-exponentially distributed, then the process is SMDP.

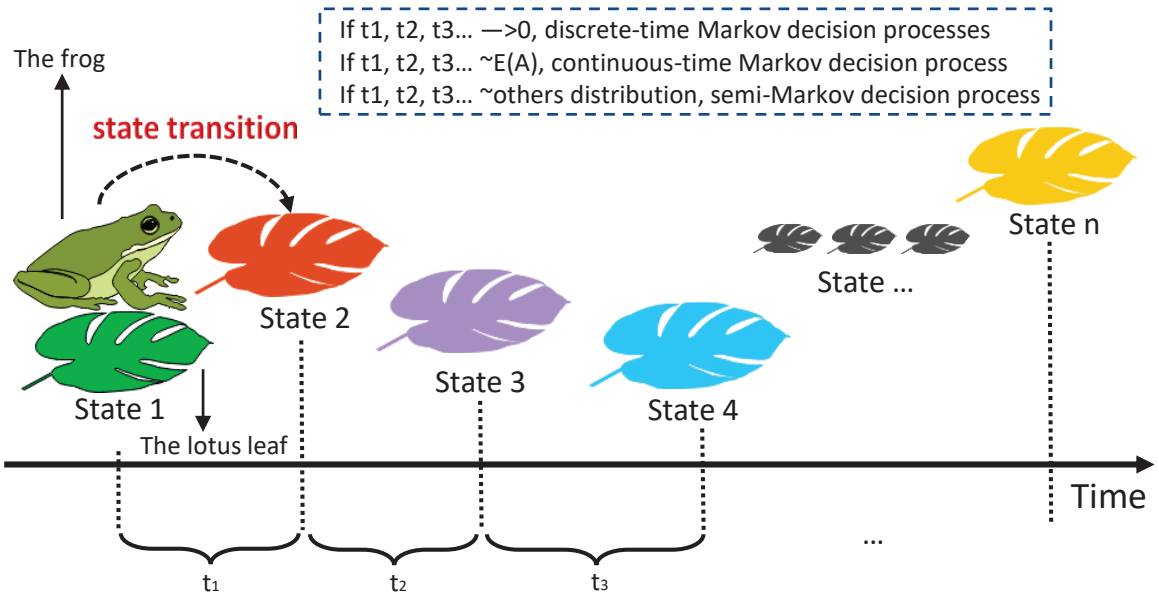

Figure 4.1: A toy example for SMDP

In the dissertation, SMDP is utilized for this problem, during which every decision is regarded as the action uniformly, and every epoch means the moment that an action is determined.

4.1.1 System State

Suppose there are totally J ECSs in the network, then the system state can be defined as a J-dimensional vector.

$$\mathbf{x}(t) = [n_1(t) \ , n_2(t), \ldots, n_j(t), \ldots, n_J(t)] \in \mathbf{S} \tag{12}$$

where $n_j(t)$ in $\mathbf{x}(t)$ indicates the number of segments currently in the ECS h_j at decision epoch t, and the state space is $\mathbf{S}$. Suppose that there is no limit of the buffer size in every ECS, so infinite segments can be stored in the queue. The system state changes as two approaches during the process: 1) at the decision epoch t, if the ECS h_j is selected from all the ECSs to compute the data segment, the system state $\mathbf{x}$ will be updated as $\mathbf{x} + \mathbf{e_j}$ and 2) the system state $\mathbf{x}$ will be updated as $\mathbf{x} - \mathbf{e_j}$ once the ECS h_j finishes one task and releases its resource. Here, $\mathbf{e_j}$ is a unit J-dimensional vector, in which the jth digital position is 1 and others are 0.

4.1.2 System Action

The smart contract will make a decision to allocate the corresponding ECS for computing if there is a fresh data segment. In the general SMDP, the decision epoch of the system is the arriving moment of every data segment. However, the system state changes in two situations, when there is a new segment being allocated to the ECS and the ECS finishes the assigned computing task. Therefore, drawing on the idea of [79], the arrival and release time instants are both defined as the decision epoch t.

Denote that $t_0 = 0$ represents the initial time instant, the decision epoch t mentioned above can be defined as $t_m, m = 0, 1, 2, \ldots$ At each epoch, the specific ECS is assigned to compute the data segment from EDs or ECSs finish tasks and release resources. Particularly, $\mathbf{a}(t_m)$ can be utilized to represent the system action

at one decision epoch t_m as follows

$$\mathbf{a}(t_m) = \left[\mathbf{a}_1(t_m), \mathbf{a}_2(t_m), \ldots, \mathbf{a}_j(t_m), \ldots, \mathbf{a}_J(t_m)\right] \tag{13}$$

where the jth component $\mathbf{a}_j(t_m)$ can be extended as

$$\mathbf{a}_j(t_m) = \left[u_{j,1}(t_m), u_{j,2}(t_m), \ldots, u_{j,I}(t_m)\right], \ \forall j \in \{1, 2, \ldots, J\} \tag{14}$$

where $u_{j,i}(t_m) \in \{0, 1\}, \forall i \in \{1, 2, \ldots, I\}, \forall j \in \{1, 2, \ldots, J\}$, means the action of the computation processed by jth ECS for the ith ED at the decision epoch t_m. To be specific, if $u_{j,i}(t_m) = 0$, the new-coming data segment will not be computed by the jth ECS. On the contrary, if $u_{j,i}(t_m) = 1$, the jth ECS will serve for the segment sent by ith ED. If ECSs only release resources and there is no segment being sent by EDs at the t_m epoch, the state should still change while the action is $\mathbf{a}(t_m) = [\mathbf{0}, \mathbf{0}, \ldots \mathbf{0}]$.

In order to meet the delay requirement δ_i^{lm} mentioned before, the total delay $T_{i,j}$, including the computing time $T_{i,j}^{comp}$ and transmission time $T_{i,j}^{net}$, should be less than or equal to this threshold. For instance, the inequation below can described for this situation, which means the jth ECS can accept the data segment from the ED d_i:

$$u_{j,i}(t_m) \neq 1, if \frac{n_j + 1}{v_j} + \frac{1}{2}\bar{\varphi}\frac{1}{v_j^2} + \frac{s_i}{b_i \log\left(1 + r_i^j\right)} > \delta_i^{lm} \tag{15}$$

Here, there is a restricted condition that $\sum_{j=1}^{J} u_{j,i}(t_m) \leq 1, \forall i \in \{1, 2, \ldots, I\}, \forall j \in \{1, 2, \ldots, J\}$ for the system action $u_{j,i}(t_m)$, that is to say, it's not permitted more than one ECS to serve the same data segment from the ED d_i at the decision epoch t_m. Naturally, if there is no ECS which can give service for the segment from ith ED in the edge computing layer, the parameter will only be $u_{j,i}(t_m) = 0, \forall j \in \{1, 2, \ldots, J\}$. In particular, the action $\mathbf{a}(t_m)$ cannot be $\mathbf{a}(t_m) = [\mathbf{0}, \mathbf{0}, \ldots, \mathbf{0}]$ when the state of the system $\mathbf{x}$ is $\mathbf{x} = [0, 0, \ldots, 0]$ at the epoch t_m, because $\mathbf{x}$ will not evolve in this situation without any task being assigned and released simultaneously.

Consequently, when the system state is $\mathbf{x}$, the action space $\mathbf{A}_\mathbf{x}$ can be defined as

$$\mathbf{A_x} = \mathbf{a}\left(t_m\right) : u_{j,i}\left(t_m\right) \neq 1, if \quad \frac{n_j + 1}{v_j} + \frac{1}{2}\bar{\varphi}\frac{1}{v_j^2} + \frac{s_i}{b_i \log\left(1 + r_i^j\right)} > \delta_i^{lm}$$
$$\sum_{j=1}^{J} u_{j,i}\left(t_m\right) \leq 1$$
$$and \quad \mathbf{a}\left(t_m\right) \neq [\mathbf{0}, \mathbf{0}, \ldots, \mathbf{0}], if\ \mathbf{x} = [0, 0, \ldots, 0] \tag{16}$$
$$\forall i \in 1, 2, \ldots, I$$
$$\forall j \in 1, 2, \ldots, J$$
$$m = 1, 2, \ldots$$

4.1.3 State Dynamics

The state transition probabilities can be utilized to describe the state dynamics. To be specific, there is a jump process in the Markov chain, which means continuous-time Markov chains have associated embedded Markov chains, the transition probability can be found through this property. For the transition probability, it can be defined as

$$p_{\mathbf{xy}}\left(\mathbf{a}\right) \triangleq \mathbf{P}\left(\mathbf{x}\left(t_{k+1}\right) = \mathbf{y} | \mathbf{x}\left(t_k\right) = \mathbf{x}, \mathbf{a}\left(t_k\right) = \mathbf{a}\right) \tag{17}$$

which stands for the probability that the system state will be converted from $\mathbf{x}$ at present epoch t_k to $\mathbf{y}$ at next epoch t_{k+1} under the condition of the $\mathbf{x}$ system state and the $\mathbf{a}$ system action at epoch t_k. All kinds of $p_{\mathbf{xy}}$ $(\forall \mathbf{x}, \mathbf{y} \in \mathbf{S})$ can collectively constitute a transition matrix under the state space. However, the transition matrix can only describe the embedded Markov chain while not suit for the continuous-time Markov chain perfectly, due to lack of the transition rates. Therefore, the transition rate matrix is proposed to perfect the process.

For this matrix, all the elements $q_{\mathbf{xy}}$ $(\forall \mathbf{x}, \mathbf{y} \in \mathbf{S})$ are greater than or equal to 0, i.e., non-negative, represents the transfer process in the system. If the element $q_{\mathbf{xy}}$

is $\mathbf{x} = \mathbf{y}$, the system state keeps static and does not have any jump. Particularly, $q_{\mathbf{xy}}$ ($\forall \mathbf{x},\mathbf{y} \in \mathbf{S}$) is the transmission rate not probability, so it can exceed the value 1.

Then, the transition probability can be formed by the transition rate.

The time that system spends on state $\mathbf{x}$ is obeyed exponential distribution with rate $\sum_{\mathbf{x}\neq\mathbf{y}} q_{\mathbf{xy}}$, after that the system will turn into state $\mathbf{y}$. The probability of that can be defined as

$$p_{\mathbf{xy}} = \frac{q_{\mathbf{xy}}}{\sum_{\mathbf{x}\neq\mathbf{y}} q_{\mathbf{xy}}}, \mathbf{x} \neq \mathbf{y} \tag{18}$$

The above formular suggests that the probability from state $\mathbf{x}$ to $\mathbf{y}$ is the ratio of specific transition rate and total cumulative transition rate. For the proposed framework in this dissertation, the transition probability can be completely redefined as

$$p_{xy}(a(t_m)) = \begin{cases} \frac{\sum_{i=1}^{I} a_{j,i}(t_m)\varphi_i}{\sum_{j=1}^{J}\left[\sum_{i=1}^{I} a_{j,i}(t_m)\varphi_i+\mu_i\right]}, if\ \mathbf{y} = \mathbf{x} + \mathbf{e_j} \\ \\ \frac{\mu_j}{\sum_{j=1}^{J}\left[\sum_{i=1}^{I} a_{j,i}(t_m)\varphi_i+\mu_i\right]}, if\ \mathbf{y} = \mathbf{x} + \mathbf{e_j} \\ \\ \quad where\ \forall j \in \{1,\ 2, \ldots,\ J\} \end{cases} \tag{19}$$

4.2 Policy and Reward Function

In this part, the policy and reward are defined explicitly. For an action $a \in \mathbf{A_x}$, it should be changed with a policy $\pi_{\mathbf{x}}$, which belongs to a set Π. This can be described as

$$\Pi = \{\pi : \mathbf{x} \rightarrow \mathbf{a} | \pi_{\mathbf{x}} \in \mathbf{A_x} \qquad \forall \mathbf{x} \in \mathbf{S}\} \tag{20}$$

As mentioned in former chapter, crypto tokens are regarded as the revenue. Suppose the ECS h_j can earn a few tokens from the ED d_i if having finished the computation

successfully, while it will be deducted several tokens if the verification is not passed. However, whether ECS h_j succeeds or fails the relative task is uncertain, so the mathematical expectation is utilized to solve this issue.

Table 4.1: Probability Distribution of Revenue on ECS h_j

Revenue of ECS	Succeeded	Failed
Probability	p_f^j	$1 - p_f^j$
Tokens	N_i	$-\rho_h Y_0$

In the table above, p_f^j represents the probability that ECS h_j can finish the task and pass the verification successfully with N_i tokens as income. Conversely, $1 - p_f^j$ represents the probability that ECS h_j fails to pass the verification, and it will be deducted $\rho_h Y_0$ tokens, of which Y_0 is the guarantee deposit previously delivered, and parameter ρ_h is the punishment percentage that can control the extent of penalty. The minus sign means the reduction of tokens. Based on this, the expectation of revenue R_{h_j} can be formulized as $\mathbb{E}\left[Revenue\right]$

$$R_{h_j} = \mathbb{E}\left[Revenue\right] = p_f^j N_i - (1 - p_f^j)\rho_h Y_0 \tag{21}$$

On the other hand, there is some cost if the ECS h_j serves for the ED d_i, including the protocol fees of the blockchain and consume energy of servers. Suppose that the server will consume P_c^j energy per second, and the computing time is $1/\mu_j$, so the total energy can be regarded as P_c^j/μ_j. For this expression, the units of time and energy are s and J respectively. Furthermore, assume that the price of energy is e_c^j per Joule, and the protocol fee is Y_h in the system. Then, the total cost can be defined as

$$C_{h_j} = \frac{e_c^j P_c^j}{\mu_j} + Y_h \tag{22}$$

On the basis of these definitions, the reward can be written as

$$
\begin{aligned}
r\left(t_m; \mathbf{x}, \mathbf{a}\right) &= \sum_{i=1}^{I} \sum_{j=1}^{J} u_{j,i}\left(t_m\right) \left[R_{h_j} - C_{h_j}\right] \\
&= \sum_{i=1}^{I} \sum_{j=1}^{J} u_{j,i}\left(t_m\right) \left[p_f^j N_i - \left(1 - p_f^j\right) \rho_h Y_0 - \frac{e_c^j P_c^j}{\mu_j} - Y_h\right]
\end{aligned}
\tag{23}
$$

In reinforcement learning, $r\left(t_m; \mathbf{x}, \mathbf{a}\right)$ represents the immediate reward at the epoch t_m. However, the cumulative reward is more significant in the whole process and should be considerate for getting the policy π, which can be described as

$$
U_\pi = \sum_{m=0}^{T} \gamma^m r\left(t_m\right) \tag{24}
$$

where the parameter γ is the discount factor ($0 \leq \gamma \leq 1$), and it will decrease in an exponential manner with the time getting farther from the present.

4.3 Policy Gradient Method

This part is mainly about how to address the SMDP problem and find the policy of proposed architecture for maximum revenues. To be specific, there is a state-of-the-art reinforcement learning algorithm named A3C algorithm [80] for the acquirement of optimal policy, which was proposed by Google Mind and had gotten success in many respects for ANNs [81]. Compared with the deep Q-learning algorithm (DQL) [82], there is a difference that A3C algorithm is not a value-based but a policy gradient method, which takes advantage of the typical actor-critic algorithm. By getting the optimal policy forthrightly, the system can respond more quickly, especially in intricate and complicated, which is certified as a better, simpler and more robust solution. In addition to this, both discrete and continuous action spaces can be utilized in this algorithm.

In the actor-critic algorithm, there are two neural networks needing to be trained: 1) the actor parameter and 2) the critic network. The parameters of these two

networks are θ_π and θ_v respectively. The critic network plays the role of referees and umpires, which gives a score to the actor so that the actor can change parameters according to this feedback. By proceeding in this manner, the actor and critic network can interact with each other in order to acquire the optimal policy. Essentially, the whole process of SMDP can be described by the policy $\pi\left(\mathbf{a}|\mathbf{x};\theta_\pi\right)$.

For the policy $\pi\left(\mathbf{a}|\mathbf{x};\theta_\pi\right)$,there are three parameters deciding the process of movement in the system. Specifically, initially, the system state is $\mathbf{x_0}$, and then an action $\mathbf{a_0}$ is selected to apply for the system, after that, the system will get a reward r_1 and turn into the next state $\mathbf{x_1}$, back and forth. The process can be described as a sequence: $\mathbf{x_0}$, $\mathbf{a_0}$, r_1, $\mathbf{x_1}$, $\mathbf{a_1}$, r_2,..., $\mathbf{x_{T-1}}$, $\mathbf{a_{T-1}}$, r_T. Here, in order to obtain the optimal policy, the cumulative reward with the discount as mentioned before is utilized for training. The cumulative reward is defined as $R_\tau = r_1 + \gamma\, r_2 + \gamma^2\, r_3 + \ldots + \gamma^{T-1} r_T$. However, there is the indeterminacy throughout the whole process, so only the math expectation can be used to estimate the cumulative reward, which is written as $\mathbb{E}\left[R_\tau|\pi;\theta_\pi\right]$

In the following content, the expectation of the cumulative reward $\mathbb{E}\left[R_\tau|\pi;\theta_\pi\right]$ is derived step by step, which is under the condition of policy π and actor neural network parameter θ_π:

$$\begin{aligned}
\nabla_{\theta_\pi}\mathbb{E}\left[R_\tau|\pi;\theta_\pi\right] &= \nabla_{\theta_\pi}\sum \pi\left(\mathbf{a}|\mathbf{x};\theta_\pi\right) R_\tau\left(\mathbf{x},\mathbf{a}\right) \\
&= \sum \nabla_{\theta_\pi}\pi\left(\mathbf{a}|\mathbf{x};\theta_\pi\right) R_\tau\left(\mathbf{x},\mathbf{a}\right) \\
&= \sum \pi\left(\mathbf{a}|\mathbf{x};\theta_\pi\right) \frac{\nabla_{\theta_\pi}\pi\left(\mathbf{a}|\mathbf{x};\theta_\pi\right)}{\pi\left(\mathbf{a}|\mathbf{x};\theta_\pi\right)} R_\tau\left(\mathbf{x},\mathbf{a}\right) \\
&= \sum \pi\left(\mathbf{a}|\mathbf{x};\theta_\pi\right) \nabla_{\theta_\pi} log\pi\left(\mathbf{a}|\mathbf{x};\theta_\pi\right) R_\tau\left(\mathbf{x},\mathbf{a}\right) \\
&= \mathbb{E}\left[R_\tau\left(\mathbf{x},\mathbf{a}\right) \nabla_{\theta_\pi} log\pi\left(\mathbf{a}|\mathbf{x};\theta_\pi\right)\right] \\
&= \sum_{t=0}^{T-1} R_t(\mathbf{x}_t,\mathbf{a}_t)\nabla_{\theta_\pi} log\pi\left(\mathbf{a}_t|\mathbf{x}_t;\theta_\pi\right)
\end{aligned} \tag{25}$$

where the derivation steps follow the chain rule and the formula $\nabla_\theta log\, y = (1/y)\nabla_\theta(y)$.

Then, the parameter θ_π can be updated as

$$\theta_\pi \leftarrow \theta_\pi + \eta_\pi \sum_{t=0}^{T-1} R_t\left(\mathbf{x}_t, \mathbf{a}_t\right) \nabla_{\theta_\pi} log\pi\left(\mathbf{a}_t|\mathbf{x}_t; \theta_\pi\right) \tag{26}$$

where η_π is the learning rate, and satisfies $0 \leq \eta_\pi \leq 1$. In addition to R_τ and θ_π, there is a state-value function defined as $V_\pi\left(\mathbf{x}; \theta_v\right)$ in the critic network, which means the summary of rewards under the condition of state $\mathbf{x}$ and the policy π

$$V_\pi\left(\mathbf{x}; \theta_v\right) = \mathbb{E}\left[R_\tau | x_0 = x; \pi\right] \tag{27}$$

For the state-action value function, it is defined as $Q_\pi\left(\mathbf{x}, \mathbf{a}; \theta_v\right)$, which indicates the summary of rewards under the condition of state $\mathbf{x}$, the action $\mathbf{a}$ and the policy π:

$$Q_\pi\left(\mathbf{x}, \mathbf{a}; \theta_v\right) = \mathbb{E}\left[R_\tau | \mathbf{x}_0 = \mathbf{x}; \mathbf{a}_0 = \mathbf{a}; \pi\right] \tag{28}$$

The advantage function $A_\pi\left(\mathbf{x}, \mathbf{a}; \theta_v\right)$ can be written as

$$A_\pi\left(\mathbf{x}, \mathbf{a}; \theta_v\right) = Q_\pi\left(\mathbf{x}, \mathbf{a}; \theta_v\right) - V_\pi\left(\mathbf{x}; \theta_v\right) \tag{29}$$

where the advantage function can take the place of the reward $R_t(\mathbf{x}_t, \mathbf{a}_t)$, so θ_π will be updated as

$$\theta_\pi \leftarrow \theta_\pi + \eta_\pi \sum_{t=0}^{T-1} A_\pi\left(\mathbf{x}_t, \mathbf{a}_t; \theta_v\right) \nabla_{\theta_\pi} log\pi\left(\mathbf{a}_t|\mathbf{x}_t; \theta_\pi\right) \tag{30}$$

However, there is a more advanced method for calculate the advantage function $A_\pi\left(\mathbf{x}_t, \mathbf{a}_t; \theta_v\right)$ in [72, 73], which is

$$A_\pi\left(\mathbf{x}_t, \mathbf{a}_t; \theta_v\right) = \gamma_t + \gamma V_\pi\left(x_{t+1}; \theta_v\right) - V_\pi\left(x_t; \theta_v\right) \tag{31}$$

Consequently, the parameter θ_v of the critic network can be revised as

$$\theta_v \leftarrow \theta_v + \delta_v \sum_{t=0}^{T-1} \left(\gamma_t + \gamma V_\pi\left(\mathbf{x}_{t+1}; \theta_v\right) - V_\pi\left(\mathbf{x}_t; \theta_v\right)\right)^2 \tag{32}$$

where δ_v is the learning rate, and satisfies $0 \leq \eta_\pi \leq 1$.

For the process of updating θ_π, there is actually the entropy which benefits the convergence by exploring more while training. The entropy of the policy can be represented as $H\left(\pi\left(\mathbf{a}_t|\mathbf{x}_t;\theta_\pi\right)\right)$, which changes following the action probability. The new training formula can be revised as

$$\theta_\pi \leftarrow \theta_\pi + \eta_\pi \sum_{t=0}^{T-1} A_\pi\left(\mathbf{x}_t, \mathbf{a}_t; \theta_v\right) \nabla_{\theta_\pi} log\pi\left(\mathbf{a}_t|\mathbf{x}_t;\theta_\pi\right) + \xi \nabla_{\theta_\pi} H\left(\pi\left(\mathbf{a}_t|\mathbf{x}_t;\theta_\pi\right)\right) \tag{33}$$

where the parameter ξ is a relatively large value at the beginning while it will decrease over time during the training process.

4.4 Feasibility of Other Methods

Reinforcement learning is similar to supervised learning, and the goal is to perform a certain action $\mathbf{a}$ in the current state $\mathbf{s}$ [83, 84]. After that, the system proceeds to the next state and receives a reward value. The process is performed repeatedly to achieve the goal. The algorithm needs to determine a function called a policy function that implements a mapping from states to actions. The algorithm needs to ensure that the cumulative reward obtained is maximized in all states according to a certain policy, hence there is the definition of the state value function and the action value function.

After the objective function has been constructed, finding the optimal policy π can be achieved by the optimization algorithm. In addition to deep learning, typical solution algorithms include Monte Carlo algorithms, time-series difference algorithms, etc.

4.4.1 Monte Carlo Algorithm

The Monte Carlo algorithm, also known as the statistical simulation method, is generally divided into three steps, including the process of constructing random probabilities, sampling from the constructed random probability distribution and solving for the estimates.

- Constructing random probabilistic processes: For problems that are inherently stochastic in nature, this probabilistic process has to be correctly described and modelled. For deterministic problems that are not inherently stochastic in nature, a probabilistic process needs to be constructed beforehand to transform them into stochastic problems.

- Sampling from a known probability distribution: Since various probabilistic models can be viewed as consisting of a variety of probability distributions, generating random variables with known probability distributions becomes a fundamental means of implementing Monte Carlo methods for simulating experiments.

- Solving for the estimate: After implementing the simulation experiment, a random variable is determined as the solution to the required problem, i.e. an unbiased estimate. Establishing the estimator is equivalent to examining the results of the experiment to obtain a solution to the problem.

4.4.2 Time-series Difference Algorithm

The time-series difference algorithm is a model-free reinforcement learning algorithm, which is also based on the principle of experimentation. The similarity between time-series difference methods and Monte Carlo is that they can learn from sample data and do not require prior knowledge of the environment. At each sampling step

when interacting with the environment using a policy, the time-series difference algorithm can be used to update the state value estimate. The time-series difference algorithm utilize estimates of the value function and can be shown to converge to the value function of the strategy π eventually.

4.4.3 Intelligence Algorithm

Intelligent algorithms are relatively new algorithms or theories that are often encountered in engineering practice, such as simulated annealing, genetic algorithms, sparrow search algorithms, etc. All these algorithms or theories share some common properties, such as simulating natural processes. Simulated annealing, genetic algorithms, forbidden search, neural networks, etc. have been improved in different perspectives and strategies in order to achieve a better "global minimum solution".

Chapter 5

Simulation Results and Discussion

In this chapter, experiment and simulation results are implemented and discussed detailly. First, the number of EDs in end device layer and ECSs in edge computing layer are both set up to three. Then the segments from EDs obey the Poisson distribution with various arrival rate φ_i. The ECSs work also with distinct computing rate v_j. Suppose that it's a Markov process between EDs in the below network layer and ECSs in the above layer, referring to [75]. All the parameters needing to be set in this system are as follows.

For simulation, the CPU and GPU are Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz and NVIDIA GeForce RTX 2060 2.59 GHz respectively, and RAM is 16.0 GB. Software environment is on Windows 11.

Based on the above parameters, there are four schemes utilized for comparison with each other.

- Scheme I: The proposed original scheme referring parameters in the table 5.1.

- Scheme II: All the EDs need to meet strict delay constraints in this proposed system.

Table 5.1: Parameter Settings

Parameters	**Values**
Total training steps	100000
Segment sizes	50,50
Buffer sizes	5000,5000
Discount factor γ	0.89
Exploration probability ξ	0.02
Arrival rate (from EDs) d_i	0.6,1.4,3.6
Computing rate (ECSs) h_j	1.0,7.4,6.6
Delay requirement δ_i^{lm}	0.2,7.8,10
Successful probability of verification p_f^j	0.69,0.67,0.63
Energy cost per second P_c^j	0.16,0.2,0.13
Energy price per Joule e_c^j	0.21,0.23,0.21

- Scheme III: There is no delay constraint in this proposed system.
- Scheme IV: The channels among EDs and ECSs are not dynamic but static in this proposed system.
- Scheme V (baseline schemes): All the EDs need to meet the same delay constraints and all the ECSs have the same computing rates under another existing system.

5.1 Convergence Performance

The convergence performance of this network architecture has been demonstrated in Figure 5.1. Through the figure, it can be noted that the loss function converges to a very small value while the reward keeps increasing in general. In the early episode, the reward is relatively low and grows rapidly. After around 500 episodes, the value has been becoming relatively stable since then. For the loss function, it decreases

during the process, which represents the algorithm has a decent convergence on the whole.

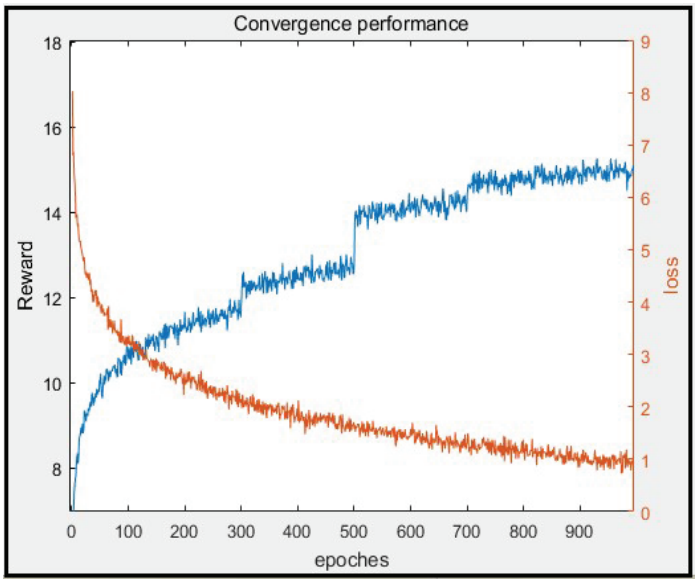

Figure 5.1: Convergence Performance

5.2 Performance with Distinct Arrival Rates

In this part, the average reward, the total delay and the task drop rate are compared among five mentioned schemes with the change of arrival rates φ_i in the system. The simulation results are demonstrated in figures 5.2, 5.3, and 5.4.

Figure 5.2 demonstrates the relationship between reward and arrival rate by adopting five various schemes. From the figure, it's shown that all the schemes increase gradually during the whole process, but tend to be stable after the arrival rate approximating to 8. The reason is that the ECSs have sufficient computing resources when the arrival rate is low. All the resources of servers have already been utilized for tasks if the arrival rate is high enough, so the average reward basically remains unchanged although the rate increases. For the scheme II, the reward is relatively lower on account of the strict delay constraints. In particular, although the performance of

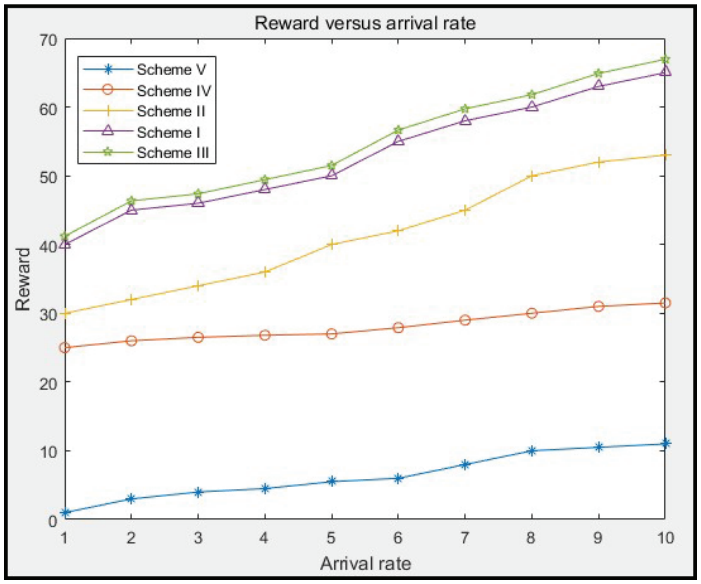

Figure 5.2: Reward versus arrival rate

scheme III shows the best result, scheme I and scheme III are getting close to each other with the increasing of arrival rates. The reason is that servers in two schemes can be both capable of computing the specific size of segments under the high arrival rate. Compared with scheme V, the first four schemes (I, II, III, IV) performs quite better in this experiment.

Figure 5.3 shows the average total delay in five various schemes, which includes computing time and transmission time in the whole process. For the scheme II, it reached the maximum delay among five schemes. The reason is that the algorithm makes servers consider maximizing the reward only if there is no delay constraint, which results in high reward and long latency. On the contrary, the delay will be larger in other schemes since the smart contract needs to balance more reward and low delay simultaneously. Particularly, it's easy to tell that the scheme IV has the smallest delay due to the static channels, so there is no transmission time in the network.

Figure 5.4 illustrates task drop rates in five schemes. In the figure, the drop

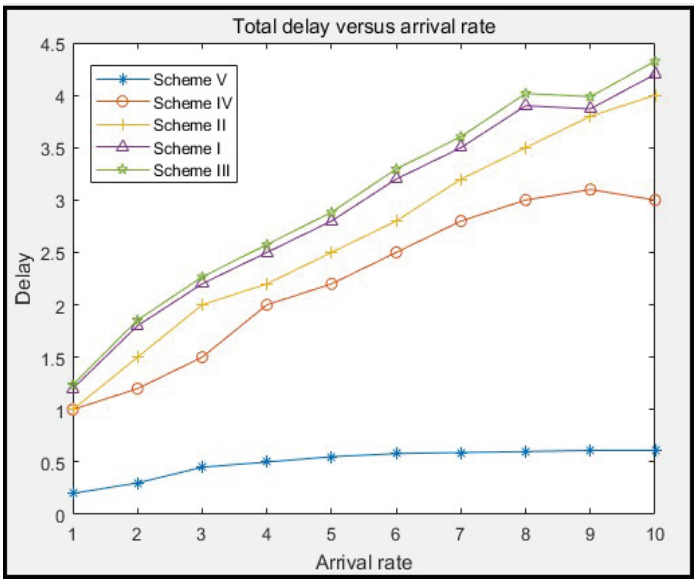

Figure 5.3: Total delay versus arrival rate

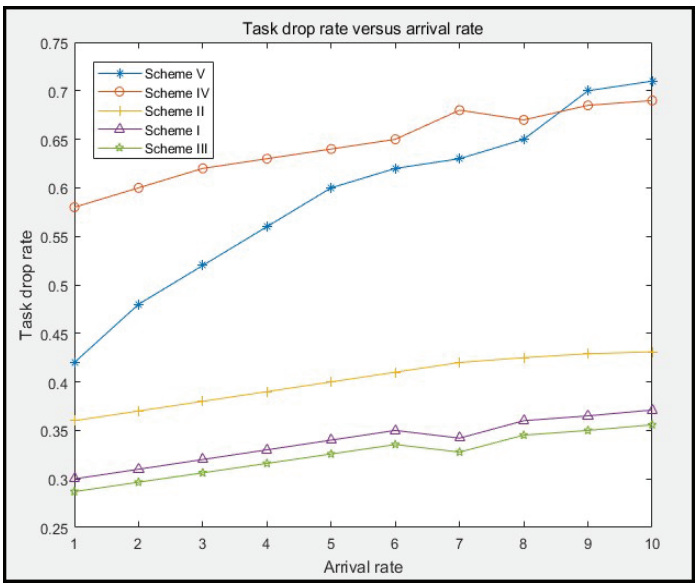

Figure 5.4: Task drop rate versus arrival rate

rate rises with the increasing of arrival rate in general. Easy to understand that servers cannot process so many computation tasks if the arrival rate is large, and then several tasks will be dropped. Additionally, the existing scheme (scheme V) shows some instability in the experiment compared with the proposed schemes in this dissertation, which means the proposed scheme can change smoothly relatively.

5.3 Performance with Distinct Edge Processing Speeds

In this part, the average reward, the total delay and the task drop rate are compared among five mentioned schemes with the change of processing rates φ_i of servers. The simulation results are demonstrated in figure 5.5, 5.6, 5.7.

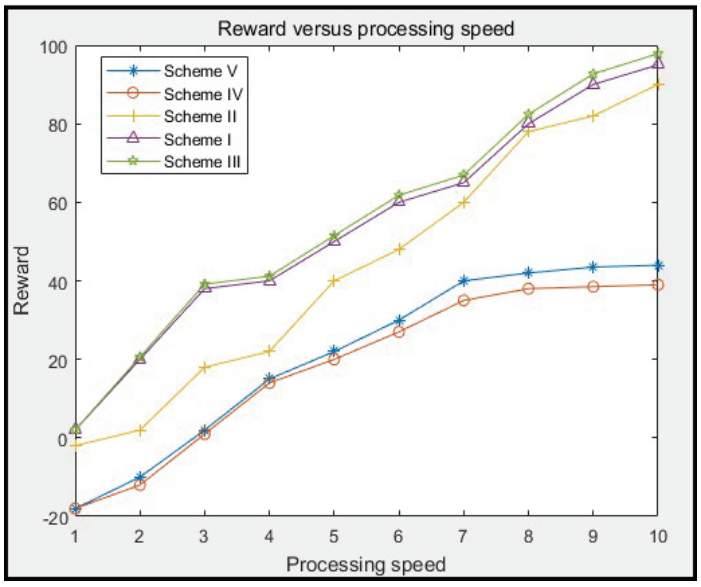

Figure 5.5: Reward versus processing speed

Figure 5.5 demonstrates the relationship between reward and processing speeds by adopting a baseline scheme and four proposed schemes. It's illustrated that all the schemes increase gradually during the whole process, but tend to be stable after the

processing speed approximating to 9, especially the scheme IV and V. Particularly, at the beginning, there is a certain gap between the scheme III and other schemes, which gets smaller with the process speed growing, and converges at last. The reason is that the ECSs have sufficient computing resources when the arrival rate is low. All the resources of servers have already been utilized for tasks if the arrival rate is high enough, so the average reward basically remains unchanged although the rate increases. For the scheme II, the reward is relatively lower on account of the strict delay constraints. In particular, although the performance of scheme III shows the best result, scheme I and scheme III are getting close to each other with the increasing of processing speeds. The reason is that servers in two schemes can be both capable of computing the specific size of segments under the high speed. Compared with scheme V, the first four schemes (I, II, III, IV) performs quite better in this experiment. It is easy to understand that all the process tasks can be finished by unitizing any scheme if the computing power of servers is large enough, which makes ECSs obtain high rewards eventually.

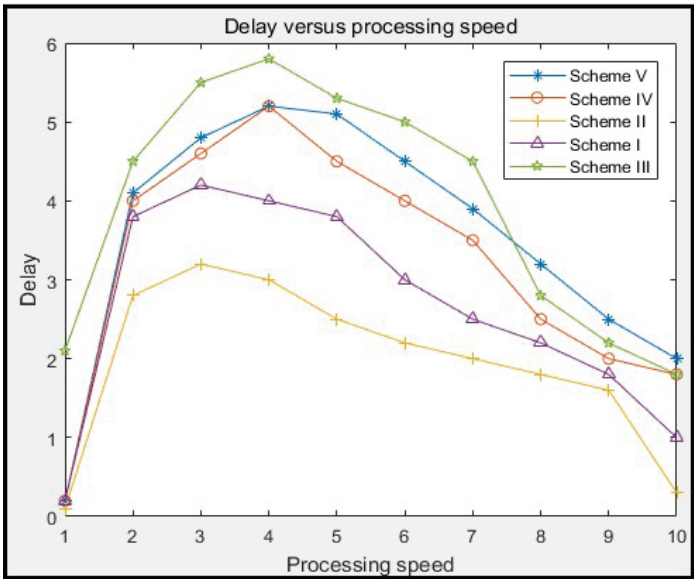

Figure 5.6: The delay versus processing speed

Figure 5.6 illustrates that the average total delay changes with processing speed

in five various schemes, which includes computing time and transmission time in the whole process. Unlike other figures, a very interesting phenomenon has emerged in the experiment. The delay time of all the schemes rises with the speed getting larger, but there is a peak in the curve (at around 4 of the processing speed). Afterwards, the delay time has been decreasing since then. The reason is that when the arrival rate remains unchanged, servers can process and compute more task with the increasing of processing speed. When the arrival rate equals to the processing speed, the peak occurs in the figure. Then, servers have ability to process more tasks compared with tasks sent from EDs, so the total delay time decrease continuously. For the scheme II, it reached the maximum delay among five schemes. The reason is that the algorithm makes servers consider maximizing the reward only if there is no delay constraint, which results in high reward and long latency. On the contrary, the delay will be larger in other schemes since the smart contract needs to balance more reward and low delay simultaneously. Particularly, it's easy to tell that the scheme IV has the smallest delay due to the static channels, so there is no transmission time in the network.

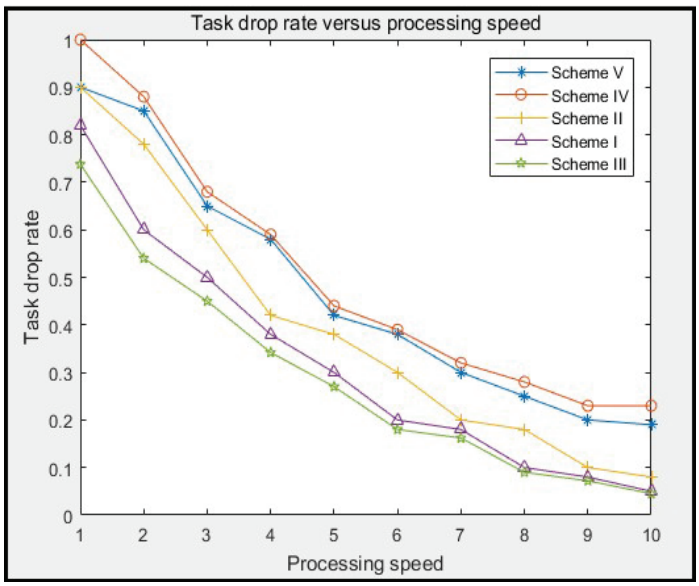

Figure 5.7: Task drop rate versus processing speed

Figure 5.7 shows task drop rates in five schemes. The result makes clear that the drop rate in five schemes descents all the time with the speed getting larger, which indicates the processing speed is an all-important factor influencing the drop rate. If the processing speed remains completely enough, there is little task being dropped.

5.4 Effect of Verification Failure Probability

Figure 5.8 demonstrates how the reward can be influenced by success probability. For the former experiment, the parameter p_f^j was set up to 0.69,0.67,0.63, and $j \in 1, 2, 3$. In order to explore the relationship and law between the average reward and the success probability, various success probabilities of verification are regarded as an independent variable, and the growth step is 0.1. The figure illustrates that the reward increases significantly with the probability growing. If the success probability is too low, the reward can even be negative. In the meanwhile, this also shows the rationalization of the proposed blockchain-based mobile edge computing paradigm and schemes.

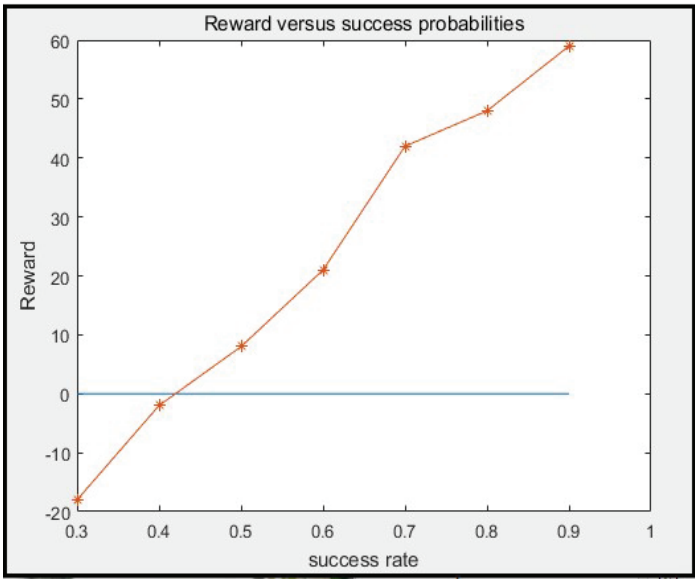

Figure 5.8: Reward versus success probabilities

Chapter 6

Conclusion and Future Directions

In the dissertation, the research mainly focuses on resource allocation for blockchain-based mobile edge computing. First of all, the privacy and security problems of edge-centric computing paradigm were considerate, which needed the blockchain technique to be addressed. Then, a blockchain-based mobile edge computing framework was designed integrally, which included all the steps one transaction would follow in the process. Additionally, a smart contract was proposed to allocate the specific ECS to proceed and compute related data segment sent from EDs. In order to address the proposed issue of the system model, the reinforcement learning method was adopted in this scenario. Not only did this framework combine the blockchain technology with edge-centric computing architecture, but also associated blockchains with artificial intelligence. At last, the experiment and simulation were illustrated in figures and analyzed detailly with all different schemes.

For the future, blockchain-based mobile edge computing paradigm will be more and more significant, and various algorithms for resource management and allocation are actually meaningful.

References

[1] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[2] M Mazhar Rathore, Awais Ahmad, Anand Paul, and Seungmin Rho. Urban planning and building smart cities based on the internet of things using big data analytics. *Computer Networks*, 101:63–80, 2016.

[3] Chao Qiu, Xiaofei Wang, Haipeng Yao, Jianbo Du, F Richard Yu, and Song Guo. Networking integrated cloud–edge–end in iot: A blockchain-assisted collective q-learning approach. *IEEE Internet of Things Journal*, 8(16):12694–12704, 2020.

[4] Chao Qiu, F Richard Yu, Haipeng Yao, Chunxiao Jiang, Fangmin Xu, and Chenglin Zhao. Blockchain-based software-defined industrial internet of things: A dueling deep q-learning approach. *IEEE Internet of Things Journal*, 6(3):4627–4639, 2018.

[5] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. Edge-centric computing: Vision and challenges, 2015.

[6] Jianbo Du, Liqiang Zhao, Jie Feng, and Xiaoli Chu. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Transactions on Communications*, 66(4):1594–1608, 2017.

[7] Jianbo Du, Liqiang Zhao, Xiaoli Chu, F Richard Yu, Jie Feng, and I Chih-Lin. Enabling low-latency applications in lte-a based mixed fog/cloud computing systems. *IEEE Transactions on Vehicular Technology*, 68(2):1757–1771, 2018.

[8] Shan Jiang, Jiannong Cao, Julie A McCann, Yanni Yang, Yang Liu, Xiaoqing Wang, and Yuming Deng. Privacy-preserving and efficient multi-keyword search over encrypted data on blockchain. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 405–410. IEEE, 2019.

[9] Kimchai Yeow, Abdullah Gani, Raja Wasim Ahmad, Joel JPC Rodrigues, and Kwangman Ko. Decentralized consensus for edge-centric internet of things: A review, taxonomy, and research issues. *IEEE Access*, 6:1513–1524, 2017.

[10] Shihao Shen, Yiwen Han, Xiaofei Wang, and Yan Wang. Computation offloading with multiple agents in edge-computing–supported iot. *ACM Transactions on Sensor Networks (TOSN)*, 16(1):1–27, 2019.

[11] Xiaofei Wang, Yiwen Han, Victor CM Leung, Dusit Niyato, Xueqiang Yan, and Xu Chen. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(2):869–904, 2020.

[12] Xiaofei Wang, Chenyang Wang, Xiuhua Li, Victor CM Leung, and Tarik Taleb. Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching. *IEEE Internet of Things Journal*, 7(10):9441–9455, 2020.

[13] Huaqing Zhang, Yong Xiao, Shengrong Bu, Dusit Niyato, F Richard Yu, and Zhu Han. Computing resource allocation in three-tier iot fog networks: A joint

optimization approach combining stackelberg game and matching. *IEEE Internet of Things Journal*, 4(5):1204–1215, 2017.

[14] Xiaofei Wang, Yiwen Han, Chenyang Wang, Qiyang Zhao, Xu Chen, and Min Chen. In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network*, 33(5):156–165, 2019.

[15] Xiuhua Li, Xiaofei Wang, Peng-Jun Wan, Zhu Han, and Victor CM Leung. Hierarchical edge caching in device-to-device aided mobile networks: Modeling, optimization, and design. *IEEE Journal on Selected Areas in Communications*, 36(8):1768–1785, 2018.

[16] Shan Jiang, Jiannong Cao, Hanqing Wu, and Yanni Yang. Fairness-based packing of industrial iot data in permissioned blockchains. *IEEE Transactions on Industrial Informatics*, 17(11):7639–7649, 2020.

[17] Hanqing Wu, Jiannong Cao, Yanni Yang, Cheung Leong Tung, Shan Jiang, Bin Tang, Yang Liu, Xiaoqing Wang, and Yuming Deng. Data management in supply chain using blockchain: Challenges and a case study. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–8. IEEE, 2019.

[18] Ruizhe Yang, F Richard Yu, Pengbo Si, Zhaoxin Yang, and Yanhua Zhang. Integrated blockchain and edge computing systems: A survey, some research issues and challenges. *IEEE Communications Surveys & Tutorials*, 21(2):1508–1532, 2019.

[19] Mengting Liu, F Richard Yu, Yinglei Teng, Victor CM Leung, and Mei Song. Computation offloading and content caching in wireless blockchain networks with mobile edge computing. *IEEE Transactions on Vehicular Technology*, 67(11):11008–11021, 2018.

[20] Junfeng Xie, Helen Tang, Tao Huang, F Richard Yu, Renchao Xie, Jiang Liu, and Yunjie Liu. A survey of blockchain technology applied to smart cities: Research issues and challenges. *IEEE Communications Surveys & Tutorials*, 21(3):2794–2830, 2019.

[21] Muhamed Turkanović, Marko Hölbl, Kristjan Košič, Marjan Heričko, and Aida Kamišalić. Eductx: A blockchain-based higher education credit platform. *IEEE Access*, 6:5112–5127, 2018.

[22] Xiulong Liu, Jiannong Cao, Yanni Yang, and Shan Jiang. Cps-based smart warehouse for industry 4.0: a survey of the underlying technologies. *Computers*, 7(1):13, 2018.

[23] Hanqing Wu, Jiannong Cao, Shan Jiang, Ruosong Yang, Yanni Yang, and Jianfei Hey. Tsar: a fully-distributed trustless data sharing platform. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 350–355. IEEE, 2018.

[24] Shan Jiang, Jiannong Cao, Juncen Zhu, and Yinfeng Cao. Polychain: a generic blockchain as a service platform. In *International Conference on Blockchain and Trustworthy Systems*, pages 459–472. Springer, 2021.

[25] Dirk Neumann, Christian Bodenstein, Omer F Rana, and Ruby Krishnaswamy. Stacee: Enhancing storage clouds using edge devices. In *Proceedings of the 1st ACM/IEEE workshop on Autonomic computing in economics*, pages 19–26, 2011.

[26] Mithun Mukherjee, Rakesh Matam, Lei Shu, Leandros Maglaras, Mohamed Amine Ferrag, Nikumani Choudhury, and Vikas Kumar. Security and privacy in fog computing: Challenges. *IEEE Access*, 5:19293–19304, 2017.

[27] Shan Jiang, Jiannong Cao, Hanqing Wu, Yanni Yang, Mingyu Ma, and Jianfei He. Blochie: a blockchain-based platform for healthcare information exchange.

In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 49–56. IEEE, 2018.

[28] Bin Liu, Xiao Liang Yu, Shiping Chen, Xiwei Xu, and Liming Zhu. Blockchain based data integrity service framework for iot data. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 468–475. IEEE, 2017.

[29] Nazanin Zahed Benisi, Mehdi Aminian, and Bahman Javadi. Blockchain-based decentralized storage networks: A survey. *Journal of Network and Computer Applications*, 162:102656, 2020.

[30] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE Security and Privacy Workshops*, pages 180–184. IEEE, 2015.

[31] Markus Klems, Jacob Eberhardt, Stefan Tai, Steffen Härtlein, Simon Buchholz, and Ahmed Tidjani. Trustless intermediation in blockchain-based decentralized service marketplaces. In *International Conference on Service-Oriented Computing*, pages 731–739. Springer, 2017.

[32] Trent McConaghy, Rodolphe Marques, Andreas Müller, Dimitri De Jonghe, Troy McConaghy, Greg McMullen, Ryan Henderson, Sylvain Bellemare, and Alberto Granzotto. Bigchaindb: a scalable blockchain database. *white paper, BigChainDB*, 2016.

[33] Muneeb Ali, Jude Nelson, Ryan Shea, and Michael J Freedman. Blockstack: A global naming and storage system secured by blockchains. In *2016 USENIX annual technical conference (USENIX ATC 16)*, pages 181–194, 2016.

[34] Muneeb Ali, Ryan Shea, Jude Nelson, and Michael J Freedman. Blockstack technical whitepaper. *Blockstack PBC, October*, 12, 2017.

[35] Muhammad Salek Ali, Koustabh Dolui, and Fabio Antonelli. Iot data privacy via blockchains and ipfs. In *Proceedings of the seventh international conference on the internet of things*, pages 1–7, 2017.

[36] Joanna Kolodziej, Damián Fernández Cerero, Alejandro Fernández Montes González, et al. Blockchain secure cloud: a new generation integrated cloud and blockchain platforms–general concepts and challenges. *European Cybersecurity Journal, 4 (2), 28-35.*, 2018.

[37] Ranjit Kumaresan and Iddo Bentov. How to use bitcoin to incentivize correct computations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 30–41, 2014.

[38] Zehui Xiong, Yang Zhang, Dusit Niyato, Ping Wang, and Zhu Han. When mobile blockchain meets edge computing. *IEEE Communications Magazine*, 56(8):33–39, 2018.

[39] Nguyen Cong Luong, Zehui Xiong, Ping Wang, and Dusit Niyato. Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach. In *2018 IEEE international conference on communications (ICC)*, pages 1–6. IEEE, 2018.

[40] Yutao Jiao, Ping Wang, Dusit Niyato, and Zehui Xiong. Social welfare maximization auction in edge computing resource allocation for mobile blockchain. In *2018 IEEE international conference on communications (ICC)*, pages 1–6. IEEE, 2018.

[41] Zehui Xiong, Shaohan Feng, Dusit Niyato, Ping Wang, and Zhu Han. Edge computing resource management and pricing for mobile blockchain. *arXiv preprint arXiv:1710.01567*, 2017.

[42] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 443–458. IEEE, 2014.

[43] Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multiparty computation using a global transaction ledger. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 705–734. Springer, 2016.

[44] Ranjit Kumaresan, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Improvements to secure computation with penalties. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 406–417, 2016.

[45] Iddo Bentov, Ranjit Kumaresan, and Andrew Miller. Instantaneous decentralized poker. In *International conference on the theory and application of cryptology and information security*, pages 410–440. Springer, 2017.

[46] Marcel Von Maltitz, Stefan Smarzly, Holger Kinkelin, and Georg Carle. A management framework for secure multiparty computation in dynamic environments. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7. IEEE, 2018.

[47] Lijing Zhou, Licheng Wang, Yiru Sun, and Pin Lv. Beekeeper: A blockchain-based iot system with secure storage and homomorphic computation. *IEEE Access*, 6:43472–43488, 2018.

[48] Attila Marosi, József Kovács, and Peter Kacsuk. Towards a volunteer cloud system. *Future Generation Computer Systems*, 29(6):1442–1451, 2013.

[49] Diego Montes, Juan A Añel, Tomás F Pena, Peter Uhe, and David CH Wallom. Enabling boinc in infrastructure as a service cloud system. *Geoscientific Model Development*, 10(2):811–826, 2017.

[50] Albertas Jurgelevičius and Leonidas Sakalauskas. Big data mining using public distributed computing. *Information Technology and Control*, 47(2):236–248, 2018.

[51] R Halford. Gridcoin: Crypto-currency using berkeley open infrastructure network computing grid as a proof of work. may 2014.

[52] Martin Grothe, Tobias Niemann, Juraj Somorovsky, and Jörg Schwenk. Breaking and fixing gridcoin. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.

[53] Loi Luu, Jason Teutsch, Raghav Kulkarni, and Prateek Saxena. Demystifying incentives in the consensus computer. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 706–719, 2015.

[54] Jason Teutsch and Christian Reitwießner. A scalable verification solution for blockchains. *arXiv preprint arXiv:1908.04756*, 2019.

[55] Julian Zawistowski, P Janiuk, and A Regulski. The golem project–crowdfunding. whitepaper. *Golem,(Nov. 201)*, 28, 2016.

[56] Cheng Li and Liang-Jie Zhang. A blockchain based new secure multi-layer network model for internet of things. In *2017 IEEE international congress on internet of things (ICIOT)*, pages 33–41. IEEE, 2017.

[57] Pradip Kumar Sharma, Saurabh Singh, Young-Sik Jeong, and Jong Hyuk Park. Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks. *IEEE Communications Magazine*, 55(9):78–85, 2017.

[58] Mayra Samaniego and Ralph Deters. Virtual resources & blockchain for configuration management in iot. *J. Ubiquitous Syst. Pervasive Networks*, 9(2):1–13, 2018.

[59] Mohammad A Salahuddin, Ala Al-Fuqaha, Mohsen Guizani, Khaled Shuaib, and Farag Sallabi. Softwarization of internet of things infrastructure for secure and smart healthcare. *arXiv preprint arXiv:1805.11011*, 2018.

[60] Pradip Kumar Sharma, Mu-Yen Chen, and Jong Hyuk Park. A software defined fog node based distributed blockchain cloud architecture for iot. *IEEE Access*, 6:115–124, 2017.

[61] Mayra Samaniego and Ralph Deters. Hosting virtual iot resources on edge-hosts with blockchain. In *2016 IEEE international conference on computer and information technology (CIT)*, pages 116–119. IEEE, 2016.

[62] Mohammad Ayoub Khan, Mohammad Tabrez Quasim, Fahad Algarni, and Abdullah Alharthi. *Decentralised Internet of Things: A blockchain perspective*, volume 71. Springer Nature, 2020.

[63] Sanjay Panikkar, Sumabala Nair, Paul Brody, and Veena Pureswaran. Adept: An iot practitioner perspective. *Draft Copy for Advance Review, IBM*, 2015.

[64] Alexandru Stanciu. Blockchain based distributed control system for edge computing. In *2017 21st international conference on control systems and computer science (CSCS)*, pages 667–671. IEEE, 2017.

[65] Ali Dorri, Salil S Kanhere, Raja Jurdak, and Praveen Gauravaram. Blockchain for iot security and privacy: The case study of a smart home. In *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*, pages 618–623. IEEE, 2017.

[66] Ali Dorri, Salil S Kanhere, and Raja Jurdak. Blockchain in internet of things: challenges and solutions. *arXiv preprint arXiv:1608.05187*, 2016.

[67] Ao Lei, Haitham Cruickshank, Yue Cao, Philip Asuquo, Chibueze P Anyigor Ogah, and Zhili Sun. Blockchain-based dynamic key management for hetero-

geneous intelligent transportation systems. *IEEE Internet of Things Journal*, 4(6):1832–1843, 2017.

[68] Walter De Brouwer and Mason Borda. Neuron: decentralized artificial intelligence, distributing deep learning to the edge of the network, 2017.

[69] Mayra Samaniego and Ralph Deters. Using blockchain to push software-defined iot components onto edge hosts. In *Proceedings of the International Conference on Big Data and Advanced Wireless Technologies*, pages 1–9, 2016.

[70] Nicolas Herbaut and Nicolas Negru. A model for collaborative blockchain-based video delivery relying on advanced network services chains. *IEEE Communications Magazine*, 55(9):70–76, 2017.

[71] Danda B Rawat, Md Salik Parwez, and Abdullah Alshammari. Edge computing enabled resilient wireless network virtualization for internet of things. In *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*, pages 155–162. IEEE, 2017.

[72] Sefki Kolozali, Maria Bermudez-Edo, Daniel Puschmann, Frieder Ganz, and Payam Barnaghi. A knowledge-based approach for real-time iot data stream annotation and processing. In *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*, pages 215–222. IEEE, 2014.

[73] Ying He, Nan Zhao, and Hongxi Yin. Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology*, 67(1):44–55, 2017.

[74] Xianping Guo and Onésimo Hernández-Lerma. Continuous-time controlled markov chains with discounted rewards. *Acta Applicandae Mathematica*, 79(3):195–216, 2003.

[75] Lei Wei, Jianfei Cai, Chuan Heng Foh, and Bingsheng He. Qos-aware resource allocation for video transcoding in clouds. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(1):49–61, 2016.

[76] Jia Wang, Jiannong Cao, Milos Stojmenovic, Miao Zhao, Jinlin Chen, and Shan Jiang. Pattern-rl: Multi-robot cooperative pattern formation via deep reinforcement learning. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 210–215. IEEE, 2019.

[77] Jia Wang, Jiannong Cao, and Shan Jiang. Fault-tolerant pattern formation by multiple robots: a learning approach. In *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pages 268–269. IEEE, 2017.

[78] Fei Yu and Vikram Krishnamurthy. Optimal joint session admission control in integrated wlan and cdma cellular networks with vertical handoff. *IEEE Transactions on Mobile Computing*, 6(1):126–139, 2006.

[79] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937. PMLR, 2016.

[80] Chenhan Xu, Kun Wang, and Mingyi Guo. Intelligent resource management in blockchain-based cloud datacenters. *IEEE Cloud Computing*, 4(6):50–59, 2017.

[81] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[82] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[83] Zhiuxan Liang, Jiannong Cao, Shan Jiang, Divya Saxena, Jinlin Chen, and Huafeng Xu. From multi-agent to multi-robot: A scalable training and evaluation platform for multi-robot reinforcement learning. *arXiv preprint arXiv:2206.09590*, 2022.

[84] Zhixuan Liang, Jiannong Cao, Shan Jiang, Divya Saxena, and Huafeng Xu. Hierarchical reinforcement learning with opponent modeling for distributed multi-agent cooperation. *arXiv preprint arXiv:2206.12718*, 2022.