Blockchain-Based Computing Resource Trading in Autonomous Multi-Access Edge Network Slicing: A Dueling Double Deep Q-Learning Approach

Thomas Kwantwi, *Graduate Student Member, IEEE*, Guolin Sun, *Member, IEEE*, Noble Arden Elorm Kuadey, *Graduate Student Member, IEEE*, Gerald Maale, *Graduate Student Member, IEEE* and Guisong Liu

***Abstract*—We investigate the computing resource allocation in multi-access edge network slicing (NS) in the context of revenue and multi-access edge computing (MEC) resource management. The significant variety of slice resource utilization levels across slice tenants (i. e., Mobile Virtual Network Operators (MVNOs)) challenges MEC resource management in NS with MEC, leading to virtual machine resource (VMR) (i. e., computing resource) wastage or scarcity. As a result, for efficient MEC resource management, the infrastructure provider (InP) encourages dynamic resource sharing and trading (DRST) of unutilized slice VMR quotas. Nevertheless, cellular network security and privacy issues deter MVNOs from collaborating on effective DRST. The security characteristics inherent in blockchain have recently gained much interest for secure resource trading. Thus, this paper proposes a unique hierarchical blockchain-based inter-slice computing resource trading (ISCRT) scheme for peer-to-peer (P2P) MVNOs in an autonomous multi-sliced MEC-based 5G network. For secure ISCRT transactions, a consortium blockchain network with hyperledger smart contracts (SC) is designed. We model the demand and pricing problems of buyer and seller MVNOs for the unutilized VMRs using a two-stage Stackelberg game. Then, to obtain the Stackelberg equilibrium (SE), an enhanced dueling double deep Q-network (D3QN) algorithm is proposed, which intelligently determines the optimal demand and pricing policies of MVNOs for the unutilized VMRs during ISCRT transactions at negotiation intervals. Simulation analysis shows that the proposed enhanced D3QN algorithm outperforms benchmark schemes in terms of the MVNO slice-level satisfaction and VMR utilization while reducing double-spending attacks in ISCRT settings by 16% and increasing both players' utility.**

***Index Terms*—blockchain, resource trading, deep reinforcement learning, Stackelberg game, MEC, network slicing, 5G**

I. Introduction

THERE has been a tremendous increase in network capacity and speed, necessitating the development of innovative application services that were previously unimaginable, thanks to the fifth-generation (5G) and beyond 5G (B5G) networks. Serving many applications on the same network adds complexity that must be managed to achieve the desired quality of service (QoS). Network Slicing (NS) is a revolutionary technology that simultaneously allows multiple applications to run on the same physical infrastructure [1]. Thus, with NS, an infrastructure provider (InP) abstracts and partitions its substrate physical network and its resources into virtual networks (slices), then assigns them to slice tenants (i. e., mobile virtual network operators (MVNOs)) for complete control and independent management. These virtual resources are utilized through their allocation to customers to meet the different QoS needs. While there are three NS architectures [2], this paper focuses on the radio access network (RAN) slicing, with the physical RAN being abstracted and partitioned into separate network slices. In the context of 5G and B5G, the key enabling technologies for RAN slicing are network function virtualization (NFV) and software-defined networking (SDN), which enable the utilization of both physical and virtual resources to deliver specific services [3].

Multi-access edge computing (MEC) is an emerging enabling technology that provides cloud computing services and network functionalities closer to mobile users at the RAN edge. Mobile users can offload complex computing tasks of their mobile user equipments (MUEs) to the network's edge for processing, reducing transmission delays and easing the strain on backhaul and core networks [4]. NS uses a variety of cloud and network resources, including computing, caching, communication resources, and network functionalities, to meet the service requirements of MVNOs. As a result, MEC plays a critical role in 5G RAN slicing design, bringing multi-dimensional resources and network functions to the edge of the network for different services. MEC addresses the high latency needs of vital services while maintaining the QoS and user experience [5]. To that end, recent studies have extensively investigated MEC and NS integration. [6] presented a federated NS scheme by investigating slice-level and user-level MEC resource allocation and sharing. A new management architecture for NS with MEC was presented in [7] to provide enhanced slicing capabilities at the 5G network's edge. In [5], a framework for NS in MEC systems, comprising slice request admission and a revenue model, was proposed.

Network slices require edge-cloud virtual machine resources (VMRs) (i. e., CPU, storage, and intra-network bandwidth) and radio resources (RRs) in the RAN. Thus, resource management is a critical issue in multi-access edge NS as it impacts the QoS

Thomas Kwantwi, Guolin Sun, Noble Arden Elorm Kuadey, and Gerald Maale are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, 611731-China (email: guolin.sun@uestc.edu.cn).

Guisong Liu is with the School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu, 610074-CHINA.

This work is supported in part by the Natural Science Foundation of China under Grant No.61806040 and Grant No. 61771098; in part by the fund from the Department of Science and Technology of Sichuan Province under Grant No. 2020YFQ0025 and Grant No. DZKJDX2021020005; and in part by the fund from Intelligent Terminal Key Laboratory of Sichuan Province under Grant No. SCITLAB-1018, SCITLAB-20019.

Manuscript received XX, X; XX ,XX.

provided by MVNOs and the utilization of the slice resources. However, due to the considerable variation in the slice resource utilization levels across MVNOs in NS with MEC, the MEC server (MECS) resource management is challenging, resulting in VMR wastage or shortage [8]. At some point, some MVNOs might have available unused slice VMR quotas, while others may have reserved insufficient VMRs. So, the MEC resource allocation method, which is rigid and can't adapt to changes in network traffic and the MVNO resource requirements, is inefficient and costly when slicing the MEC.

To address these issues, dynamic inter-slice resource trading has been proposed in some literary works in the context of NS. The authors in [9] [10], proposed different algorithms for dynamic resource trading among slice tenants for autonomous resource slicing. However, their scenarios only considered RAN slicing without integrating the MEC, and their approach had a high computational cost, making it unfeasible for real situations. In contrast, in this paper we investigate NS in a MEC-based 5G RAN and try to solve some of the technical challenges encountered during resource trading. The resource trading approach described in this paper allows for the sharing and trading of unused VMRs from underloaded MVNO slices with overloaded MVNO slices that value such resources to meet their VMR demand requirements. Furthermore, negotiating and optimizing the allocation of the available unused VMRs among the overloaded MVNO slices is tackled since no single management entity controls the unused VMRs and the underloaded MVNO slices trade their unused VMRs for their benefit. Due to slice isolation issues, allocating the unused VMRs to the overloaded MVNO slices has some critical challenges. (1) Interoperability issues in security and privacy arise between MVNOs, resulting in information leakage [11]. This limits MVNO collaboration and cooperation for effective MEC resource management. (2) Executing large-scale decentralized VMR transactions among MVNOs in an unstable and opaque edge-cloud slicing market poses trust issues because a third party does not audit trading transactions. (3) Issues of over-committing the same unused VMRs to different buyers will lead to double-spending attacks, which slow down the network. (4) The VMR trading procedures for MVNO slices can put the security and privacy of the MVNO's data at risk because a central controller collects sensitive information like the slice's identity, number of users, location, etc.

Blockchain technology's decentralization, anonymity, and trust benefits have been widely investigated to address the security challenges identified in multi-sliced networks. The blockchain concept uses a decentralized ledger-based storage mechanism with no third parties involved to distribute secured transactions [12]. Thus, blockchain has been proposed in MEC slicing to guarantee security in resource trading, dynamic resource provisioning, and better network performance. The authors of [13] and [14] provided a distributed blockchain-based NS (DBNS) scheme that allows dynamic leasing of resources, including MEC by service providers and resource providers, to guarantee good performance for services. According to [15], a secure and transparent slicing technique that creates MVNO slices via blockchain technology was suggested. Even though the proposed solution tackles the double-spending attack problem, there is no interoperability among slice tenants and blockchain-based techno-economic VMR trading procedures. A blockchain network slicing broker (BNSB) was introduced in [16], where an intermediary broker receives resource requests and responses and then allocates MEC resources across slice tenants while scheduling physical resources from InP using smart contracts (SCs). In this work, the problem of resource under/over-utilization is not tackled. Reinforcement learning (RL) [17] is preferred over classic optimization approaches for tackling the problem of MEC resource allocation in NS, while accounting for varying slice tenant requirements and traffic patterns [6]. The work in [16] considered blockchain technology and deep reinforcement learning (DRL) for secure MEC resource allocation at the InP-MEC slice tenants level. However, their research interests did not prioritize resource allocation at the inter-slice level.

Despite the fact that previous works investigated blockchain and DRL for slicing the MEC and secure MEC resource sharing separately, there are still identified research gaps that must be addressed: (1)What dynamic MEC resource-sharing techniques are technically suitable for efficiently managing the limited MECS resources by slicing RAN together with MEC? (2) In MEC slicing, how trustworthy, secure, and fair can the centralized NSB be in the inter-slice computing resource trading (ISCRT) framework? (3) What are the most effective methods for incentivizing MVNOs to safely and securely exchange their unused VMRs, and at what price? (4) How does combining blockchain, MEC slicing, and DRL help in resource allocation in NS with MEC? As far as we know, this is the first effort that tries to combine blockchain, MEC resource trading, and DRL for ISCRT and optimal resource allocation in NS with MEC.

Hence, this paper presents a secured ISCRT scheme and intelligent dynamic VMR allocation in a multi-sliced MEC-based 5G RAN, leveraging on the knowledge of blockchain technology, MEC slicing, and DRL. We provide a permissioned blockchain with a Hyperledger SC for VMR negotiation between an underloaded MVNO slice (seller) and overloaded MVNO slices (buyers). We create a two-stage Stackelberg game model for the seller-buyer interaction and define a pricing and VMR demand problem for each participant, respectively. The motivation for game theory application is to create a competitive market between the sellers who offer their idle VMRs and the buyers who value such resources, where each player ensures the maximization of their utilities. To attain the Stackelberg equilibrium (SE), a model-free dueling double deep Q-learning (D3QN) technique is proposed with multi-objective rewards, which updates the stochastic policy rule without prior information from the MEC environment. The goal of using the DRL is to learn the entities' joint optimal pricing and demand strategies that solve the proposed Stackelberg game without any prior information from the ISCRT environment. The proposed enhanced DRL-based technique increases MEC resource usage and QoS satisfaction levels by independently altering the VMR of the slices of MVNOs. Then, we conduct a MECS-level VMR adjustment reflecting the MVNO slice VMR allocation.

The following are the main contributions of this paper:

- We propose a blockchain-based enhanced DRL scheme with hierarchical levels for securing ISCRT in an autonomous multi-sliced MEC-enabled 5G RAN.
- We set up a hyperledger trading SC on a consortium blockchain network to provide a secure and transparent ISCRT between the seller and multiple buyer MVNOs.
- We devised a two-stage single-leader multiple-follower (SLMF) Stackelberg game model for a single seller and multiple buyer MVNOs based on the pricing and demand strategies, respectively. The seller (leader) initially decides the unit price for its unused VMRs, and then the buyers (followers) determine their demand request for the available unused slice VMRs.
- We propose a new enhanced DRL scheme with multiple objective reward functions to obtain the optimal pricing and VMR demand request policies for intelligent MEC resource management. We aim to optimize the utilities of the VMR seller and the buyers while balancing QoS satisfaction and VMR utilization within the restrictions of an MVNO slice.

The rest of the paper is organized as follows. The related works are found in Section II, while the system model is found in Section III. The problem formulation is contained in Section IV, and Section V introduces the proposed advanced DRL scheme for ISCRT. Section VI provides the performance evaluation. Lastly, Section VII concludes this paper and gives directions for future work.

II. Related Works

Recently, there has been a surge in research towards integrating the MEC with 5G NS [7]. Consequently, the NS technique has been applied to orchestrate MEC infrastructure and resource partitioning in MEC-enabled 5G RANs [5], [6]. The authors in [6] presented a federated NS scheme by researching slice-level and user-level MEC resource allocation. The centralized MVNO controller was installed on the network, ensuring optimal MEC resource distribution to network service providers (NSPs) and providing MEC resources to users. The ability of the centralized MVNO controller to be trusted in the network is a limitation that was not within their purview. [5] proposes a unique framework for NS with MEC, which includes slice request admission and a profit model, to explore the operator's escalation revenue problem considering traffic variances. The security concerns of slice tenants, however, were not addressed. In [18], a survey of current research on decentralizing applications with the integration of blockchain in 5G and beyond networks was presented.

Integrating blockchain with MEC NS is anticipated to unleash the full benefits of MEC-enabled 5G services. In [19], the authors created sub-slices in a 5G network by deploying a single NSB as an intermediate between NSPs and resource providers. The NSB's centrality, on the other hand, increases communication overhead and calls into question its fairness. An NSB technique proposed in [20] uses an intermediary brokering agency and SC subleasing resources from the InPs to slice tenants securely and automatically. A blockchain NSB (BNSB), acting as an intermediate broker, was proposed in [16]. The BNSB receives resource requests and responses, then allocates resources across slice tenants (STs), as well as schedules MEC resources from the InP via SCs. The QoS criteria of MVNOs were met because MEC resource owners were prevented from over-committing their resources through dynamic configuration. Consequently, the double spending attack is prevented. The authors, however, do not provide procedures for economically slicing and efficiently utilizing the MEC resources in an autonomous slice resource allocation in a secure wireless network environment.

The emergence of game theory provides an efficient tool to model the interactions between competing resource requesters and providers in an inter-slice trading environment, which gives new direction to add to current research. In [21], a dynamic pricing-based resource allocation system using the two-stage Stackelberg game model was proposed. Each player in the formulated game in this work aims to optimize their strategy to maximize their utility. In [22], the authors jointly optimize the computing resource price and demand in a blockchain-based computing resource trading scheme between smart vehicles and mobile users in an Internet of Vehicles (IoV)-assisted smart city through solving a formulated Stackelberg game. The game optimizes each player's strategy to achieve maximum utility. Furthermore, game theory has been applied to resource allocation in MEC and blockchain scenarios [23] [24]. The authors of [23] model the interaction between cloud/fog providers and miners in a PoW-based blockchain network as a two-stage Stackelberg game. In [24], the interaction between edge cloud operators and collaborative mining networks is modeled using a Stackelberg game to obtain the optimal resource price and device resource demands when offloading to edge clouds. These efforts solve the game analytically and only address trading between resource providers and miners for MEC resource management, not security issues in resource trading. In contrast, using blockchain technology, our work secures ISCRT transactions between resource providers and requesters.

Recent breakthroughs in RL demonstrate the technology's potential to learn the stochastic policy of changing environments without prior information. The ISCRT and unused VMR allocation problems in MEC slicing can be formulated with regard to a sequential decision-making problem and solved using the RL technique. However, using classical RL to tackle the ISCRT and allocation problems will likely produce enormous state and action spaces. DRL combines DNN feature representation and RL decision-making and can handle such large dimensional state/action spaces. Thus, the DRL technique can solve the ISCRT and allocation problems [25]. Due to its success in combining Q-learning (QL) and DNNs, the DQN is commonly used in game-play [26]. In [27], DRL was applied to obtain the Nash equilibrium (NE) in a pricing problem modeled using the Stackelberg game framework for the interaction between cloud providers and miners in the blockchain-based Industrial Internet of Things (IIoT) for efficient resource management. [28] proposed using blockchain and AI to create a safe and intelligent 5G wireless network architecture. A D2D cache matching and dynamic bandwidth provisioning problem were solved with a blockchain-enabled DRL scheme to maximize system utility. Furthermore, some researchers respond to DQN's inadequacies by developing and proposing new DQN approaches. In [10], the authors use an advanced DQN (dueling DQN) approach to solve the SE for the formulated Stackelberg game, which establishes the optimal price and resource demand policies for autonomous bandwidth slice resource allocation during the negotiation interval. In this research, only radio resource allocation in RAN slicing was investigated. The integration of MEC and the fact that different MEC application services have varied QoS needs were not considered. In the related DRL works discussed above, these works solved the game-based resource allocation problems in resource trading. However, their works experienced Q-value overestimation issues and slow convergence speeds. To address these issues, we use an advanced DRL scheme (D3QN) that combines the double DQN and dueling DQN models, which keeps the Q-value from being overestimated and speeds up convergence.

The above-identified drawbacks motivate us in this work to leverage the inherent security qualities of blockchain, MEC slicing, and advanced DRL for secure ISCRT and dynamic VMR allocation in a multi-sliced, MEC-enabled 5G network. We describe a consortium blockchain-based secure ISCRT between an underloaded MVNO slice (seller) and overloaded MVNO slices (buyers) in real-time autonomous MEC slicing based on traffic fluctuations.

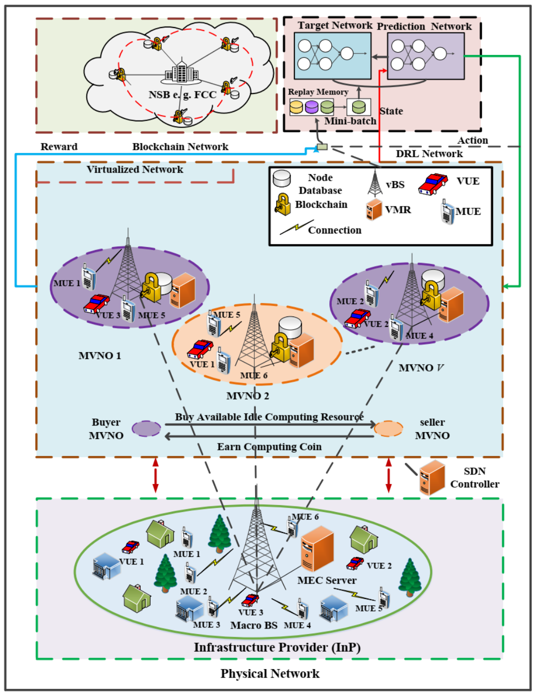


Fig. 1: System Architecture Framework

III. System Model

We implement a system architecture that consists of four components: physical, virtualized, blockchain, and DRL networks. Coexisting in the system are users (MUEs), autonomous virtual networks (MVNO slices), a macro BS (MBS), MECS, SDN controller, NSB, bandwidth resources (BRs), virtual machine resources (VMRs), and a DRL agent. Each MUE transmits its channel together with its status information to the MBS via the SDN controller for correct admission control, MUE association, and effective resource allocation. The NSB is a trustworthy regulatory agency such as the Federal Communication Commission (FCC) that controls the blockchain platform by being atop the network. The radio frequency resources that can be allocated to specific MVNO slices are defined as BRs with granular units of physical resource blocks (PRBs). The MECS provides the VMRs, which include CPU, storage, and intra-network bandwidth, which we divide into distinct categories contingent on the availability of these three types of resources. The set of VMR categories is denoted by the expression $\mathcal{L} = \{1, 2, 3, ..., L\}$, where L represents the total number of VMR categories. The VMR category l has a resource set written as $\mathbf{VM}_l = \{vm_l^{(c)}, vm_l^{(n)}, vm_l^{(s)}\}$, where $vm_l^{(c)}$, $vm_l^{(n)}$ and $vm_l^{(s)}$ are the number of computing units (CUs) of CPU resources, intra-network network, and storage of the VMR category l, respectively. We assume there are enough BRs provisioned for slices and that this work mainly focuses on the sharing, trading, and allocation of unutilized CPU resources. The SDN controller is decentralized by associating it with all blockchain network entities. As a result, the controller's data is replicated across all entities, thus preventing data loss when there is an external attack or a simple controller failure [29]. Because each DRL agent runs on the SDN controller, they are both decentralized by default. Based on what it observes in the network, the DRL agent chooses the appropriate actions to obtain optimal pricing and demand policies for efficient MEC resource management.

As indicated in Fig. 1, we describe the system architecture by assuming that the substrate physical network infrastructure, including MBS and MECS resources, is abstracted and logically partitioned into slices, each independently controlled by an MVNO. The MVNOs provide diverse edge computing services to their assigned MUEs in each MVNO slice. The DRL agent monitors network traffic variations and recommends optimal CPU resource balancing among the multiple MVNO slices for autonomous control of the MECS's VMRs. A blockchain network, which is decentralized, provides secure and transparent CPU resource exchange between MVNO slices to achieve this balance. The NSB certifies, authenticates, and regulates all nodes and their public and private keys. Thus, unlike in [19], the NSB is no longer central. The NSB cannot be attacked because users in the ISCRT utilize dynamic public keys to mask their identities or private information coming from other participants. The NSB also coordinates the ISCRT market by collecting trading information from buyers and sellers; monitoring and regulating the ISCRT market; putting in place the right trading mechanisms to make sure that the supply and demand of CPU resources are in balance; and setting rules for payment and allocating CPU resources.

A. *Operating Business Model*

We create a two-tiered business model where the InP operates a MECS attached to MBS and acts as the slice provider. InP subleases BRs and VMRs to MVNOs, who intend to operate the network slices independently. We assume a set $v \in \mathbf{V} = \{1, 2, 3, ..., V\}$ representing the number of MVNOs as slices, where V indicates the number of slices. Following this sequel, we denote a group of MVNOs who provide their unused CPU resources for a fee as sellers, and MVNOs who purchase these unused CPU resources to meet their resource deficit due to insufficiency or scarcity as buyers. Let $\boldsymbol{V^{sell}} = \{v_i | i \in \{1, 2, 3, ..., N_I\}\}$ represent a set of seller MVNOs and $\boldsymbol{V^{buy}} = \{v_j | j \in \{1, 2, 3, ..., N_J\}\}$ represent set of buyer MVNOs, so that $\boldsymbol{V^{Sell}} \cup \boldsymbol{V^{buy}} = \mathbf{V}$. The untrusted features during transmission and broadcasting in MEC-enabled 5G wireless networks can cause privacy leakages, double-spending attacks, or general system instability in situations where MVNOs do share private data with the regulator [30] during ISCRT. Thus, to solve this problem, we set up a blockchain network that makes ISCRT transactions between MVNOs decentralized, transparent, and trustworthy.

In order to participate securely in the ISCRT scheme using the consortium blockchain platform, each MVNO $v_{i,j}$ requires a unique identification address $ID_{v_{i,j}}$ for identity authentication. An MVNO creates an ISCRT blockchain account and obtains a private key $SK_{v_{i,j}}$, a public key $PK_{v_{i,j}}$, and a wallet address $wADD_{v_{i,j}}$ [31]. The private key is used in signing transactions digitally, and the public key is used to identify nodes anytime other nodes want verification of ISCRT transactions. The ISCRT transactions will be carried out using the wallet addresses. The MVNOs also receive an account balance $accBAL_{v_{i,j}}$. As a result, the NSB creates and stores a mapping list $\{ID_{v_{i,j}}, SK_{v_{i,j}}, PK_{v_{i,j}}, wADD_{v_{i,j}}, accBAL_{v_{i,j}}\}$ representing the ISCRT transaction details of each MVNO $v_{i,j}$.

B. *Slicing Model*

By using the wireless network virtualization (WNV) technique, we abstract the physical network and create partitions of virtual networks as slices. We assume five MVNO slices, each comprised of a VM that reserves a portion of the MECS resources for offloading services with stringent QoS criteria based on the 5G specification (i. e. ultra-reliable low latency communication (URLLC)) [7]. An association manager in each slice assigns MUEs to the physical MBS of the InP based on the slice's aggregated demand and traffic distribution. The individual QoS class flows are mapped onto the QoS classifier index (QCI) table [32] to classify flows into slices. In addition to monitoring slice states, distributed SDN controllers know which flows are in which slices. To guarantee dynamic and effective MEC resource management via secure ISCRT, the SDN controller collects QoS satisfaction and CPU resource utilization data from slices at set time intervals. A deployed DRL agent learns the QoS satisfaction and CPU resource utilization of each slice at regular intervals, ensuring dynamic and effective MEC resource management via ISCRT. The provisioned CPU resources for each slice are mapped to the physical MECS resources, which enables the MUEs to offload their tasks and make use of CPU resources reserved for slices with varied QoS. After each slice is allocated reserved portions of the MECS's CPU resources, it makes optimal computing resource allocation decisions for the MUEs assigned.

1) Network Model: Fig. 1 depicts a multi-user MEC-based 5G network set up by an InP, with a MECS connected to a single MBS and connected MUEs indicated by the set $k \in \mathbf{K} = \{1, 2, 3, ..., K\}$. The MBS has a system BR capacity of S (MHz), which consists of $\widehat{b}$ (PRBs) with w (kHz) time and frequency domain bandwidths, whereas the MECS has a maximum CPU resource capacity of F (GHz), consisting of $\xi^{(c)}$ CUs of CPU resources. Let $\mathbf{K}_v$ represent the set of MUEs served by a specific slice v for computing offloading services and k_v indicate a single MUE served by a slice $v \in \mathbf{V}$. Each MUE in a slice v generates a computation offloading task T_{k_v}, represented by the tuple $T_{k_v} \triangleq (q_{k_v}, \widehat{f}_{k_v}, T^*_{k_v}), \forall v \in \mathbf{V}$, where q_{k_v} is the total data size (in bits) of the task generated, $\widehat{f}_{k_v}$ indicates the needed CPU resources necessary for executing a bit of the data, and $T^*_{k_v}$ is the required time deadline for task execution. Due to the low power and limited processing capability of each MUE, executing the generated task locally within the deadline is challenging. As a result, each MUE k_v offloads its generated task to its associated slice v, where it is remotely executed using the assigned CPU resource of MECS.

2) Remote Computing Model: This work considers an orthogonal frequency division multiple access (OFDMA) system to reduce intra-cell interference. The MBS determines how much of the system BR is allocated to each MUE for offloading computation tasks to the MECS, as well as how much CPU resources of slice v are allocated to each MUE to execute offloaded tasks. In a slice v, the MUE k_v achievable data rate can be indicated by the expression:

$$\varrho_{k_v} = \alpha_{k_v} S \log_2 \left(1 + \frac{X_{k_v} Y_{k_v}}{N_\sigma}\right), \quad \forall v \in \mathbf{V}, k_v \in \mathbf{K}_v \quad (1)$$

where α_{k_v} indicates the allocated fraction of BRs to MUE k_v in slice v, X_{k_v} the uplink transmission power of the MUE k_v, Y_{k_v} the channel gain between MUE k_v and MBS in slice $v \in \mathbf{V}$, and N_σ the Gaussian noise power.
We express the time necessary for the MUE k_v in slice v to offload its generated task T_{k_v} to the MECS as:

$$\widehat{T}^{off}_{k_v} = \frac{T_{k_v}}{\alpha_{k_v} S \log_2 \left(1 + \frac{X_{k_v} Y_{k_v}}{N_\sigma}\right)}, \quad \forall v \in \mathbf{V}, k_v \in \mathbf{K}_v \quad (2)$$

After the MECS has received all of the MUEs' offloaded tasks from all slices, the MECS allocates the CPU resources required to execute the MUEs' offloaded tasks. As a result, the execution service delay encountered by the MUE k_v of a slice v at the MECS is expressed as:

$$\widehat{T}^{exe}_{k_v} = \frac{q_{k_v} \cdot \widehat{f}_{k_v}}{f_{k_v} \cdot \{vm^{(c)}_l\}}, \quad \forall v \in \mathbf{V}, k_v \in \mathbf{K}_v \quad (3)$$

where $f_{k_v} \cdot \{vm^{(c)}_l\}$ denotes the CPU resource allocation to the MUE k_v of slice v and f_{k_v} indicates the minimum number of VMR category l with CPU capacity $vm^{(c)}_l$. Eventually, the total time delay that will be experienced by an MUE k_v of slice v in performing the remote computation is as follows:

$$\widehat{T}^{d}_{k_v} = \widehat{T}^{off}_{k_v} + \widehat{T}^{exe}_{k_v}, \quad \forall v \in \mathbf{V}, k_v \in \mathbf{K}_v \tag{4}$$

The total time delay $\widehat{T}^{d}_{k_v}$ experienced by MUE k_v in processing its task at MECS should not exceed the maximum completion time $T^{*}_{k_v}$, expressed as:

$$\widehat{T}^{d}_{k_v} \leq T^{*}_{k_v}, \quad \forall v \in \mathbf{V}, k_v \in \mathbf{K}_v \tag{5}$$

C. Utility Model Formulations

1) Utilization of VMRs (CPU resources): The CPU resource utilization of a slice $v \in \mathbf{V}$ is defined as the ratio of the total number of CUs of CPU resources required to compute the task of all MUEs in the slice v to the fraction of the CPU resource pool allocated to a slice v. Thus, the CPU resource utilization of a slice v can be expressed as:

$$\Theta_v(t) = \frac{\sum_{k_v=1}^{K_v} f_{k_v} \cdot \{vm_l^{(c)}\}}{vm_l^{(c)}}, \quad \forall v \in \boldsymbol{V} \tag{6}$$

where $\sum_{k_v=1}^{K_v} f_{k_v} \cdot \{vm_l^{(c)}\}$ represents the total amount of CPU resources of the MECS VMR occupied by MUEs under slice v, and $vm_l^{(c)}$ denotes the amount of CPU resource capacity of VMR category l of the MECS assigned to a slice v.

2) MVNO slice QoS satisfaction: The slice QoS satisfaction is defined by the user's experience with the latency (delay) requirements using the QCI table for computing its offloaded tasks. In a slice v, we define the QoS satisfaction in the range [0, 1], where 0.5 or higher indicates user satisfaction and less than 0.5 indicates user dissatisfaction. The sigmoid function is used to model user QoS satisfaction in a slice v with satisfaction priority on the delay in processing the MUE offloaded task and can be defined as [10];

$$\varpi_{k_v} = \frac{1}{1 + \exp(-\eta(T^{*}_{k_v} - \widehat{T}^{d}_{k_v}))} \tag{7}$$

where η indicates a constant determining the shape of the satisfaction curve, and $T^{*}_{k_v}$ denotes the maximum completion time of executing an MUE task q_{k_v} in a slice v. The QoS satisfaction ϖ_v experienced by MUEs under a slice v is given by the sum of the individual delay ϖ_{k_v} users associated with a slice v experience while offloading and computing their tasks, and can be expressed as:

$$\varpi_v = \sum_{k_v=1}^{K_v} \varpi_{k_v} \tag{8}$$

3) Trading utility function for MVNOs: Buyer MVNOs must pay a seller MVNO after utilizing its unused CPU resources to rectify a CPU resource deficit as offloading service demand increases. So, any seller MVNO generates revenue by selling its unused CPU resources to buyer MVNOs. At each ISCRT transaction interval t, let P_i denote the unit price set by a seller MVNO v_i for its unused CPU resource supplied to buyer MVNO v_j and c_j indicate the amount of the unused CPU resources demanded by buyer MVNO v_j. The seller MVNO v_i trades its unused CPU resources while keeping in mind the unit cost ω it paid to the InP for CPU resources. As a result, the utility function for a seller MVNO v_i is represented as follows:

$$U_{v_i}(\boldsymbol{P}, \boldsymbol{c}) = \sum_{j=1}^{J} (P_i - \omega) c_j \tag{9}$$

where $\boldsymbol{P}$ denotes the unit price vector for the unused CPU resources represented as $\boldsymbol{P} = [P_1, P_2, P_3, ..., P_{N_I}]^T$ and $\boldsymbol{c}$ indicating the vector of the amount of unused CPU resource purchased by buyer MVNOs with $\boldsymbol{c} = [c_1, c_2, c_3, ..., c_{N_J}]^T$. We indicate that $\forall_{v_j}$, the amount of the unused CPU resources c_j intended to be purchased, is a function of the set price P_i, (i. e., $c_j \triangleq f(P_i)$), which means that the amount of unused CPU resources demanded by buyer MVNO v_j is determined by the unit price set. Furthermore, we denote the seller MVNO v_i's available unused CPU resources as $\phi_i = vm_l^{(c)} - \sum_{k_v=1}^{K_v} f_{k_v} \cdot \{vm_l^{(c)}\}$. The total available unused CPU resources that can be allocated to all buyer MVNOs can not exceed ϕ_i. To determine the cost of purchasing available unused CPU resources, each buyer MVNO v_j considers its real demand shortfall while purchasing unused CPU resources from a seller MVNO v_i. Thus, the utility function of a buyer MVNO v_j can be expressed as:

$$U_{v_j}(\boldsymbol{P}, \boldsymbol{c}) = \beta \log_2 \left(1 + \frac{c_j}{\mu_j}\right) - c_j \cdot P_i \tag{10}$$

where $\log_2 \left(1 + \frac{c_j}{\mu_j}\right)$ indicates the additional CPU resource obtainment satisfaction gain, β denotes a positive coefficient which is used in converting the additional CPU resource obtainment satisfaction gains into monetary reward, and $c_j \cdot P_i$ represents the unit cost incurred for purchasing the obtained allocation c_j of unused CPU resources from seller MVNO v_i. The additional CPU resource obtainment satisfaction gain indicates an increasing function of the additional CPU resource purchased c_j. Furthermore, the logarithmic function can disclose the degree to which buyer MVNOs are satisfied while getting c_j extra CPU resources via ISCRT in comparison to their actual CPU resource demand shortfall μ_j.

IV. Problem Formulation

In this section, we describe the process for a secure P2P ISCRT among the buyer and seller MVNOs and the Stackelberg game formulation for optimal trading strategies. At each ISCRT negotiation interval, MVNOs negotiate CPU resource allocation updates in response to traffic variations. At the end of each ISCRT transaction, the MVNO slice CPU resources are readjusted based on who needs additional CPU resources and revenue.

A. Secure P2P ISCRT with Consortium Blockchain

The ISCRT market is vulnerable to double-spending and hacking attacks. A double-spending attack occurs when the same unused CPU resources are subleased to different buyers simultaneously, causing business friction. The hacking attack happens when nodes maliciously tamper with ISCRT transaction records. Thus, the role of the consortium blockchain ensures that ISCRT transactions between a seller MVNO and a buyer MVNO are secured. Each ISCRT transaction is encrypted, signed digitally, recorded, and hashed in a block resistant to tampering, preventing double-spending and hacking attacks. The consortium blockchain, like the hybrid blockchain, combines public and private blockchain features to provide authorization and solve the monopoly issue. In a resource-sharing environment like ISCRT, it is preferred over public and private blockchains for security, high throughput, and low latency. Because there are no processing fees and publishing new blocks is not computationally expensive, mining costs are reduced. The consortium blockchain network comprises MVNOs (buyers and sellers). The allocation of the unused VMRs (CPU resources) to buyer MVNOs is based on executing SCs. These SCs are executed using Hyperledger [33], hence it is referred to as the Hyperledger Iroha SC. When specific conditions are satisfied, hyperledger Iroha SCs automatically execute lines of code stored on a blockchain platform. The SC is preferable to the traditional SLA due to the advantages of speed, precision, and confidence. We provide below the operational steps in the consortium blockchain:

1) Initializing system: Before participating in the consortium blockchain network, nodes must register with the regulator (NSB). We use an Elliptic Curve Digital Signature Algorithm (ECDSA) and public-key encryption to initialize the system [34]. Following the formation of a blockchain account, the NSB gives a certificate $Cert$ to each node that qualifies for unique transaction details in the tuple $\{ID, SK, PK, wADD, accBAL\}$. Using the public-key encryption technique, we can mathematically provide an expression for the data integrity from sender x during the system initialization as [35]:

$$\mathbf{C}_{PK_x}(Sign_{SK_x}(Hash(MSG))) = Hash(MSG) \tag{11}$$

where $Sign_{SK_x}$ indicates the sender x's digital signature with its private key SK_x, $\mathbf{C}_{PK_x}$ decodes the signed data using the sender's public key PK_x, and the hash digest of message is $Hash(MSG)$.

2) Choosing a verifier based on their reputation: Only seller MVNOs can be selected as verifiers since they have enough CPU resources for the block verification process. However, to verify and audit blockchain blocks, we presume that not all seller MVNO nodes are trustworthy. Because of this, we use a subjective logic model to select verifiers based on their reputations [31]. It specifies a level of trust that all nodes agree to. All nodes must vote "truthful" or "untruthful" based on previous verification and audit records to convey their opinions about other nodes. Following the vote, Eq. (12) imposes a selection criterion on the selection process, where REP_{v_i} represents the MVNO v_i's reputation and REP_{thr} is the minimum threshold. After that, only the "truthful" nodes are chosen for block verification and audit.

$$REP_{v_i} \geq REP_{thr} \tag{12}$$

3) The ISCRT between a seller MVNO and buyer MVNOs: A buyer MVNO submits a request req_{v_j} to the decentralized blockchain network in order to bid for unused CPU resources. The blockchain network's SC nodes then agree on the transactional rules and assess any potential exceptions. In an interest-based competition game, buyer MVNOs try to maximize their utility with an optimal demand for unused CPU resources. Simultaneously, the seller MVNO seeks an appropriate price strategy for its unused CPU resources. The NSB connects the seller to legitimate buyers, while the SC mediates. The selling MVNO then allocates unused CPU resources to the buyer MVNOs, who pay for the unused CPU resources they are allocated.

4) Generating and broadcasting a blockchain block: All nodes on the blockchain network choose a block-creating node N after a successful ISCRT transaction. The *opinion* of each node determines which node is the *leader* for the block construction of a given transaction until its admission into the continuing chain. Each transaction is encrypted and signed digitally by the elected leader, and the transaction is recorded and stored in a tamper-resistant block. Finally, all network nodes are notified and are able to verify and audit the block.

5) Consensus mechanism for block verification and auditing: We use a lightweight Practical Byzantine Fault Tol-

Algorithm 1 Consortium Blockchain-based ISCRT

```
1:  Initialization: NSB, ($V^{buy}$, $V^{sell}$)
2:  Registration: The buyer and seller MVNOs are registered
3:  for a time period $i$ do
4:      */ SLMF Stackelberg game for ISCRT and block
        creation /*
5:      for all $V$ and NSB do
6:          Using batch information, verify $Cert_{v_{i,j}}$
7:          if $Ver(Cert_{v_{i,j}})$ = TRUE then
8:              Create a two-stage SLMF Stackelberg game
                using Eq. (13) and Eq. (14)
9:              Run and execute SC for ISCRT and construct
                block
10:         else
11:             End SC
12:         end if
13:     end for
14:     */ Verification of block and PBFT consensus /*
15:     for Leader of mining nodes $n$ do
16:         The current Blk data is broadcast to all nodes
17:         for all mining nodes do
18:             if mining node Tx data = Blk data then
19:                 set verify Blk = TRUE
20:             else
21:                 set verify Blk = FALSE
22:             end if
23:             Audit results are broadcast to the mining
                leader $N$ for analysis
24:         end for
25:         Accept and add the block to the current chain,
            or delete the block that does not pass verification
26:     end for
27: end for
```

erant (PBFT) consensus method to approve our consortium blockchain network blocks [36]. Unlike Bitcoin, PBFT does not require intensive block mining, resulting in greater energy efficiency. With few nodes, PBFT encourages each pre-selected verification node. These pre-selected nodes audit blocks and send their findings for analysis by the leader. If consensus is reached, the leader evaluates the audit results of the verification nodes. If all verifiers agree that the block is correct, the leader broadcasts it, and it is added to the chain, containing the preceding block's hash. Algorithm 1 describes the ISCRT steps between seller MVNO and buyer MVNOs for the available unused CPU resources. Achieving consensus for block verification and auditing with the PBFT protocol in Algorithm 1 has a time complexity of $\mathcal{O}(n^2)$, indicating the cost of communication and scalability of the consensus method.

B. The Two-Stage SLMF Stackelberg Game Formulation for Optimal Trading Strategies

Achieving mutual advantage from the ISCRT paradigm requires efficient economic incentives for under and over-loaded MVNO slices to trade available unused CPU resources. So, we built an ISCRT system that benefits both sides. The buyer MVNOs are allocated the unused CPU resources for VMR adjustment at the MECS level to boost QoS. In our scenario, a single-seller MVNO and multiple-buyer MVNOs are ideal for ISCRT and unused CPU resource allocation. Using the hierarchical Stackelberg game model [37], we can simulate single-seller multiple-buyer trading. In a Stackelberg game, players are rational and choose their strategies based on other players' strategies. Using a two-stage SLMF Stackelberg game model, we create an incentive strategy for efficient ISCRT. In stage I, the leader (seller MVNO) announces the unit price for its available unused CPU resources. Then buyer MVNOs, as followers in stage II, make a non-cooperative matching demand decision for the available unused CPU resources. A seller MVNO (leader) must determine the optimal unit price P_i that maximizes its utility. Similarly, buyer MVNOs (followers) should make demand decisions for the unused CPU resource to maximize utility while enhancing their QoS. Thus, the leader and followers are constantly adjusting their strategies to gain more. The primary objective of formulating this game is to identify the SE point(s) that benefit both the leader and followers the most. The ISCRT optimization problems for the formulated SLMF Stackelberg game are provided below:

Stage 1: Leader's pricing for unused CPU resources:

$$\begin{aligned} &\max_{P\geq 0} U_{v_i}(P_i, c_j) \\ s.t. \quad &\sum_{j=1}^{N_J} c_j \leq \phi_i, \quad \mu_{j,i}^t \leq \phi_i, \quad \phi_i > 0 \end{aligned} \tag{13}$$

where $U_{v_i}(P_i, c_j)$ indicates the utility function of a seller MVNO v_i defined in Eq.(9), P_i is the unit unused CPU resource price vector with $[P_1, P_2, P_3, ..., P_{N_I}]^T$, and c_j is a vector of unused CPU resources demand of buyer MVNOs with $[c_1, c_2, c_3, ..., c_{N_J}]^T$. The buyer MVNOs determine their demand for the use of available unused CPU resources to gain more profit based on the pricing strategy of the seller MVNO.

Stage 2: Followers' demand for the unused CPU resources:

$$\begin{aligned} &\max_{c\geq 0} U_{v_j^b}(c_j, P_i) \\ s.t. \quad &U_{v_j} > 0 \end{aligned} \tag{14}$$

where $U_{v_j}(c_j, P_i)$ represents the utility function of a buyer MVNO v_j defined in Eq.(10).

We form the Stackelberg game from Eq.(13) and Eq.(14) with the primary objective of obtaining the SE point, where neither a seller MVNO (leader) nor a buyer MVNO (follower) has the incentive to deviate. We define the SE as follows:

Definition 1 (Stackelberg Equilibrium): We denote the SE as (P^*, c^*), where P^* is the optimal unit price solution set for the seller MVNO problem and c^* is the optimal unused CPU resource demand solution for the buyer MVNO problem. The point (P^*, c^*) is then optimal if, for any (P, c) with $P \geq 0$ and $c \geq 0$,

$$U_{v_i}(P^*, c^*) \geq U_{v_i}(P, c^*) \tag{15}$$

$$U_{v_j}(c^*, P^*) \geq U_{v_j}(c, P^*) \tag{16}$$

By solving for the second-order derivatives of the given utility functions of the buyer MVNO with respect to P and the seller MVNO with respect to c, we can ascertain the uniqueness and existence of a SE in our Stackelberg game formulated. In [38], the authors prove that the backward induction method can be employed to achieve the SE for the formulated two-stage Stackelberg game. However, this conventional analytical method for deriving the SE is a heuristic algorithm that requires precise information from the trading environment [39], which is impractical when trading participants value privacy. In addition, any changes in market conditions will necessitate the computation of new optimal solutions. Due to the seller and buyer MVNOs' non-cooperation, it is a challenge to have access to complete trading information. On the other hand, the DRL method can learn the optimal strategies in a dynamic trading environment without prior knowledge. This offers a new technique for obtaining the SE without the trading entities disclosing their private information. Thus, we design an enhanced DRL-based technique for determining the optimal pricing and demand strategies for the unused CPU resources in an autonomous multi-sliced MEC-based 5G network for effective MECS resource management.

V. Proposed Advanced DRL Based Algorithm for Optimal Pricing and Demand Policies

A. Transforming the Pricing and Demand Problem into a Markov Decision Process (MDP)

In the proposed ISCRT framework, optimal unused CPU demand and pricing is a decision-making process where buyer and seller MVNOs try to maximize their trading utility. Each buyer /seller MVNO tries to find the optimal demand for unused CPU resources and pricing strategies while balancing QoS satisfaction and MECS /slice CPU resource utilization. The optimal pricing and demand decision-making problem for

ISCRT is formulated as a Markov decision process (MDP), which provides a mathematical framework for solving the game problem by RL. The three elements of the MDP model are defined by states, actions, and rewards. Thus, we express the MDP as a stochastic process represented by the tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}(s^{t+1}/s^t, a^t), \mathcal{R}, \mathcal{S}^{'})$, where $\mathcal{S}$ is the state set and $\mathcal{A}$ is the action set. We indicate $\mathbb{P}(s^{t+1}/s^t, a^t)$ as the state transition probability, $\mathcal{R}$ as reward, and $\mathcal{S}^{'}$ as the subsequent state. Using the Markov property, the value function of state s^t with policy π is expressed as [40]:

$$\begin{aligned}\mathcal{V}^{\pi}(s^t) &= \mathbb{E}_{\pi}\left\{\sum_{t=0}^{\infty}\gamma^t r^t(s^t, a^t)\right\} \\ &= \mathbb{E}_{\pi}\left[r^t(s^t, a^t) + \gamma\mathcal{V}^{\pi}(s^{t+1})\right],\end{aligned} \tag{17}$$

where $r^t(s^t, a^t)$ represents the immediate reward, $\mathcal{V}^{\pi}(s^t)$ represents the immediate utility, and $\mathcal{V}^{\pi}(s^{t+1})$ the subsequent utility. The state-value function for acquiring the optimal policy according to the Bellman equation [41] is expressed as follows;

$$\mathcal{V}^{\pi^*}(s^t) = \arg\max_{a^t \in A}\left\{\mathcal{V}^{\pi}(s^t)\right\} \tag{18}$$

We define the Q-function associating with the policy π as the expected discounted reward of taking an action a^t in the state s^t, and is indicated by;

$$\mathcal{Q}^{\pi}(s^t, a^t) = \mathbb{E}_{\pi}\left[\mathcal{R}^{\pi}|s^t = s, a^t = a\right] = r(s^t, a^t) + \gamma \cdot \sum_{s^{t+1}\in\mathcal{S}}\mathbb{P}\big(s^{t+1}|s^t, a^t\big)\left(\sum_{a^{t+1}\in\mathcal{A}}\pi(s^{t+1}, a^{t+1})\mathcal{Q}^{\pi}|(s^{t+1}, a^{t+1})\right) \tag{19}$$

As a result, the optimal Q-functions with optimal policies are denoted by;

$$\begin{aligned}\mathcal{Q}^{\pi *}(s^t, a^t) = r(s^t, a^t) + \gamma \sum_{s^{t+1}\in\mathcal{S}}\mathbb{P}\big(s^{t+1}|s^t, a^t\big)\cdot \\ \max_{a^{t+1}\in\mathcal{A}}\mathcal{Q}^{\pi *}(s^{t+1}, a^{t+1}),\end{aligned} \tag{20}$$

The optimal value function can also be expressed as $\mathcal{V}^{\pi^*}(s^t) = \max_{a^t\in\mathcal{A}}\left\{\mathcal{Q}^{\pi *}(s^t, a^t)\right\}$. In Q-learning the one-step updating rule is indicated as:

$$\mathcal{Q}^{\pi}(s^t, a^t) \leftarrow \mathcal{Q}^{\pi}(s^t, a^t) + \alpha_{\sigma}(y^t - \mathcal{Q}^{\pi}(s^t, a^t)), \tag{21}$$

where the temporal difference (TD) target is $y^t = r^t + \gamma\max_{a^{t+1}\in\mathcal{A}}\mathcal{Q}^{\pi}(s^{t+1}, a^{t+1})$ and the influence of new information on current value is determined by α_{σ}, which is the learning rate. MDPs are classified as model-based or model-free depending on whether the state transition probability can be predicted in advance. Because of the dynamic and uncertain wireless network environment, obtaining the state transition probability for our considered system is difficult. Thus, a model-free MDP is used, and we define the states, actions, and rewards of the defined MDP model as follows:

State(s): The information state of an MVNO slice changes at time intervals due to the varying VMR (CPU resource) requirements of an MVNO in the MEC-based network environment. To deal with the changing requirements, a DRL agent which is situated on an SDN controller monitors the state of each sub-slice and recommends optimal pricing for the unit unused CPU resource and demand prediction policies for CPU resource allocation across buyer and seller MVNO slices. We represent the state $s^t_{v_{i,j}}$ of the seller and buyer MVNO slice $v_{i,j}$ at decision step t with the tuple $s^t_{v_{i,j}} = \{\Theta^t_{v_{i,j}}, \varpi^t_{v_{i,j}}, U^t_{v_{i,j}}\}$, where $\Theta^t_{v_{i,j}}$, $\varpi^t_{v_{i,j}}$, and $U^t_{v_{i,j}}$ represent the QoS satisfaction, CPU resource utilization and the ISCRT utility of MVNO slice $v_{i/j}$, respectively. In the ISCRT process, the trading utility $U^t_{v_{i,j}}$ of an MVNO slice is determined by its role, where the seller MVNO v_i keeps adjusting its strategy to maximize its unit price for its unused CPU resource and buyer MVNO v_j aims at maximizing its profit with an optimal demand for the unused CPU resources. Based on the input states, the learning agent generates an appropriate action to be taken.

Action(a): The learning agent takes an action $a^t_{v_{i,j}}$ at each state $s^t_{v_{i,j}}$ based on observations from the environment. The DRL agent can choose from two alternative sets of actions taken independently and sequentially by buyer MVNOs and a seller MVNO: the extra CPU resource demand action c_j and the surplus unit pricing action P_i. Thus, we denote the set of extra CPU resource demand actions taken by each buyer MVNO v_j as $c^t \in a^t_{v_{i,j}}$ and the set of unit prices for surplus CPU resource taken by seller MVNO v_i as $P^t \in a^t_{v_{i,j}}$ at any decision step t. Each buyer MVNO and the seller MVNO act separately and in sequence. According to the actions taken by the players to maximize their ISCRT utility, the DRL agent alters the CPU resource allocation of an MVNO slice, using the values from the given set $\delta^t \in a^t_{v_{i,j}}$ as $\delta^t = \{-0.5, -0.4, -0.3..., 0.5\}$. The set δ^t contains normalized values of percentage upward and a downward adjustment to the CPU resource allocation of an MVNO's slice. As a result, a complete action set is defined as $a^t_{v_{i,j}} = \{c_j, P_i, \delta^t\}$. The specified action updates the CPU resource allocation of an MVNO slice at the MECS level.

Reward(r): Through the wireless network environment interaction, the DRL agent explores potential alternative actions while utilizing the possible optimal actions. We aim to maximize the ISCRT utility of the slices of MVNOs while balancing the QoS satisfaction of a slice and CPU resource utilization. As a result, the reward function of the DRL scheme is defined as:

$$r^t_{v_{i,j}} = U^t_{v_{i,j}} + (\bar{x}\cdot\Theta^t_{v_{i,j}} + \bar{y}\cdot\varpi^t_{v_{i,j}}) \tag{22}$$

The constant values $\bar{x}$ and $\bar{y}$ represent the relative weights given to various aspects of QoS, such as user (MUE) satisfaction and efficient utilization of CPU resources.

The DRL agent learns the optimal decision policy over time based on the MDP model by maximizing the rewards in its interactions with the wireless network environment [41]. A decision policy π of an agent comprises a series of subsequent actions and the policy $\pi(s^t, a^t)$ equals the probability of taking an action a^t dependent on the state s^t, $\sum_{a^t\in A}\pi(s^t, a^t) = 1$.

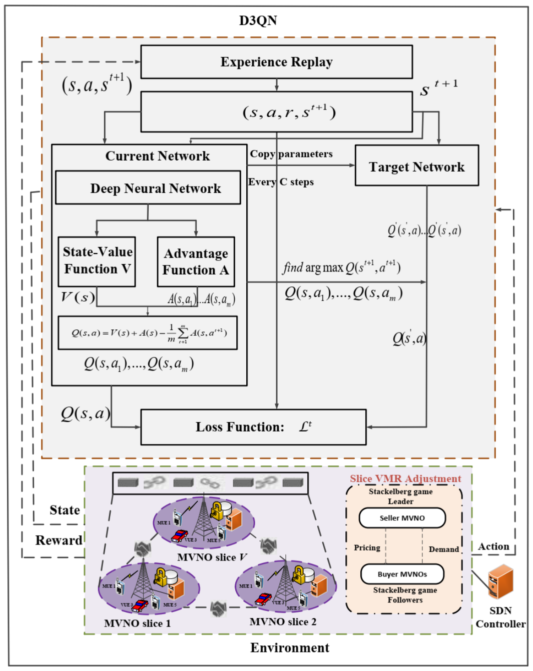


Fig. 2: The framework of D3QN

The purpose of the DRL agent is to learn optimal policies (strategies) that maximize the cumulative expected rewards. Consequently, the long-term cumulative discounted reward starting from state s^t at time step t can be expressed as follows: $\mathcal{R}^\pi = \sum_{t=0}^{\infty} \gamma \cdot r_v^t(s^t, \pi(s^t)), \forall s^t \in \mathcal{S}$, where $\gamma \in (0,1)$ represents the discount factor which balances the significance of the interim and distant future rewards. The aim is to finding the optimal policy π^* that can maximize the reward $\mathcal{R}^\pi$, which is written as $\pi^* = \text{argmax}_\pi \mathcal{R}^\pi$.

B. D3QN for ISCRT and MEC Resource Management

Given a small state space and action space, the QL algorithm can effectively find the optimal policy. However, in the presence of an uncertain wireless network environment and no prior knowledge, there is a large belief space, and the QL algorithm may fail to find the optimal policy. Thus, we can use the DQN algorithm to approximate the action-value function [40] to solve our formulated MDP (i.e., $\mathcal{Q}(s^t, a^t; \theta) \approx \mathcal{Q}^*(s^t, a^t)$), where θ represents the weight parameter set of the Q-network. However, solving the formulated MDP pricing and demand problem with the native DQN algorithm causes slow convergence speed and overestimates the Q-value of the action, which causes the estimation error to be large as the number of actions increases [42]. The positive bias in updating the maximum function in QL is the cause of the overestimation. To address these concerns, the ISCRT scheme uses an improved network model called D3QN to solve the pricing and demand problems. Thus, in this paper, the D3QN algorithm is used to train our agent for improved accuracy and stability. The model of the proposed D3QN-based strategy is indicated in Fig. 2.

1) Overview of D3QN-based Algorithm: The D3QN has a structure that is similar to DQN. However, its concept is based on combining the double DQN [42] and dueling DQN [10] methods, which bring significant adjustments to improve performance and minimize overestimation of action values. The double DQN effectively mitigates overestimation and increases learning performance, whereas the dueling DQN improves convergence speed and achieves higher stability. In the conventional DQN, the target network is used to compute the target Q-value y_i^{DQN} in the non-terminal state indicated as:

$$y_i^{DQN} = r^t + \gamma \max_{a^{t+1} \in \mathcal{A}} \mathcal{Q}(s^{t+1}, a^{t+1}; \theta_i') \quad (23)$$

where θ_i' indicates the network parameter for computing the target at iteration i. The double DQN, on the other hand, takes the maximum Q-value computed by the current network to determine the appropriate action a^t and then uses it to obtain the matching $\mathcal{Q}_{max}$ value from the targeted network, indicated as:

$$y_i^{DDQN} = r^t + \gamma \mathcal{Q}(s^{t+1}, \underset{a^{t+1} \in \mathcal{A}}{\text{argmax}}\, \mathcal{Q}(s^{t+1}, a^{t+1}; \theta); \theta') \quad (24)$$

By separating the target Q-value action selection process from the target Q-value computation process, the double DQN prevents the selection of sub-optimal actions that are over-estimated. The learned loss function is expressed as follows:

$$\mathcal{L}^t = \left[y_i^{DDQN} - \mathcal{Q}(s, a; \theta) \right]^2 \quad (25)$$

Also, the proposed D3QN framework includes a dueling Q-network that makes learning across actions more general without changing the underlying DQN algorithms. This provides the algorithm with more stability and efficiency [10]. The dueling Q-network consists of two-stream estimator functions, namely the state-value estimator function and the state-dependent action advantage function. Thus, given a policy π, we can express the state value $\mathcal{V}^\pi(s^t)$ and state-action value pair $\mathcal{Q}^\pi(s^t, a^t)$ as:

$$\mathcal{V}^\pi(s^t) = \mathbb{E}_{a \sim \pi(s)}[\mathcal{Q}^\pi(s^t, a^t)] \quad (26)$$

$$\mathcal{Q}^\pi(s^t, a^t) = \mathbb{E}_\pi \left\{ r(s^t, a^t), \pi \right\} \quad (27)$$

Then using Eq.(26) and Eq.(27), the advantage function is defined as:

$$\mathcal{A}^\pi(s^t, a^t) = \mathcal{Q}^\pi(s^t, a^t) - \mathcal{V}^\pi(s^t) \quad (28)$$

The advantage function $\mathcal{A}^\pi(s^t, a^t)$ indicates the necessity of taking actions in specific states, while $\mathcal{V}^\pi(s^t)$ and $\mathcal{Q}^\pi(s^t, a^t)$ tells the goodness of a state and the value of selecting a specific action in a state, respectively. With an optimal policy given as $max_{a^t \in \mathcal{A}} \mathcal{Q}(s^{t+1}, a^{t+1})$, we indicate that $\mathcal{Q}^\pi(s^t, a^t) = \mathcal{V}^\pi(s^t)$, and accordingly $\mathcal{A}^\pi(s^t, a^t) = 0$ and the output given by the dueling network is expressed as:

$$\mathcal{Q}(s^t, a^t; \theta, \varsigma, \vartheta) = \mathcal{V}(s^t; \theta, \varsigma) + \mathcal{A}(s^t, a^t; \theta, \vartheta), \quad (29)$$

where we indicate the common network parameter as θ, the advantage stream parameters as ϑ, and the value stream parameters as ς. The Q-values for an action a^t taken at state s^t are generated as follows:

$$\mathcal{Q}^{\pi}(s^t, a^t; \theta, \varsigma, \vartheta) = \mathcal{V}^{\pi}(s^t; \theta, \varsigma) + (\mathcal{A}^{\pi}(s^t, a^t; \theta, \vartheta) \\ -\frac{1}{m}\sum_{a^{t+1}} \mathcal{A}(s^t, a^{t+1}; \theta, \vartheta)), \quad (30)$$

where *m* is the action space size and $\frac{1}{m}\sum_{a^{t+1}} \mathcal{A}(s^t, a^{t+1}; \theta, \vartheta)$ indicate the mean of the advantage value for all actions in state s^t. Next, we expressed the computation for the TD target in the proposed D3QN algorithm as:

$$y_t^{D3QN} = r^t + \gamma \mathcal{Q}^{\pi}(s^{t+1}, \underset{a^{t+1}}{\operatorname{argmax}}\, \mathcal{Q}^{\pi}(s^{t+1}, a^{t+1}; \theta); \theta'), \quad (31)$$

where $\theta = (\varsigma; \vartheta)$ indicates the parameters of the enhanced current network. During the training phase, the current network parameters are updated using a one-step gradient descent, which performs Eq. (32).

$$\Delta_{\theta} \frac{1}{|\mathcal{D}|} \sum_{(s,a;\theta)\in\mathcal{D}} \left(\mathcal{Q}^{\pi}(s, a; \theta) - y_t^{D3QN}\right)^2, \quad (32)$$

where $\mathcal{D} = \{(s^t, a^t, r^t, s^{t+1})\}$ denotes a mini-batch of experiences randomly sampled from a replay buffer $\mathcal{B}$, which is the experience replay memory mechanism considered for storing each experience (s^t, a^t, r^t, s^{t+1}) to eliminate the correlation between consecutive states and to speed up the algorithm's convergence rate. In this paper, the target parameters θ' are updated every ℓ step via a soft update, which is expressed as follows:

$$\theta' \leftarrow \nu\theta' + (1-\nu)\theta, \quad (33)$$

where ν denotes the factor for the soft update.

2) Optimization of Exploration Strategy: The $\varepsilon - greedy$ policy used as the exploration strategy in the native DQN algorithm has the same exploration probability for all actions. Thus, it may not explore the optimal solution in random selection, resulting in inefficient learning. Hence, we utilize the Boltzmann exploration method [43] to improve our exploration strategy in this paper. This technique selects actions based on their Q-value probability rather than actions with equal probability. Thus, the selection probability of each action is expressed as;

$$p(a^t|s^t) = \frac{\exp(\tilde{\beta}\cdot Q(s^t, a^t; \theta))}{\sum_{a^{t+1}\in\mathcal{A}} \exp(\tilde{\beta}\cdot Q(s^t, a^{t+1}; \theta))} \quad (34)$$

where $\tilde{\beta}$ represents the exploration coefficient, which when larger indicates a marginal exploration range. By applying the Boltzmann exploration strategy, we ensure a greater probability of exploring better options. The $\varepsilon - greedy$ and Boltzmann exploration approaches are combined in the proposed D3QN algorithm, which selects the exploration action with ε probability using Eq. (34), otherwise the optimal action is selected.

The CPU resource pool of an MVNO slice is updated after a specified action has been enforced on the CPU resource pool of an MVNO slice. The CPU resource update expression is written as;

Algorithm 2 Optimized D3QN-based ISCRT Algorithm

```
1: Initialization: Initialize current network parameters $\theta = (\varsigma; \vartheta)$, target parameters equal to current parameters $\theta' \leftarrow \theta$, replay memory size $B$, mini-batch size $D$, epsilon $\epsilon$ and Q-tables.
2: for episode = 1,..., $M$ do
3:    Set up the environment.
4:    for each decision step $t$, do
5:       Observe the state $s^t_{v_{i,j}} = \{\Theta^t_{v_{i,j}}, \varpi^t_{v_{i,j}}, U^t_{v_{i,j}}\}$.
6:       /* ** ISCRT and CPU resource allocation* * */
7:       Through blockchain, execute Algorithm 1.
8:       With $\epsilon$ probability select a random action $a^t_{v_{i,j}}$ based on the probability distribution in Eq. (34), otherwise, select optimal $a^t_{v_{i,j}} = \operatorname{argmax}_{a^t} \mathcal{Q}(s^t, a^t; \theta, \varsigma, \vartheta)$.
9:       Execute action $a^t_{v_{i,j}}$ to update the CPU resource pool $vm^{(c)}_l$ of the MVNO slice $v_{i,j}$.
10:      Observe reward $r^t_{v_{i,j}}$ and the next state $s^{t+1}_{v_{i,j}}$.
11:      Adjust the required computing resource allocation $\{vm^{(c)}_l\}^{t+1}_{i,j}$ at physical MECS-level.
12:      /* * * * ** Learning Update* * * * */
13:      Store the experience $(s^t_{v_{i,j}}, a^t_{v_{i,j}}, r^t_{v_{i,j}}, s^{t+1}_{v_{i,j}})$ into $B$.
14:      Sample randomly $D$ experiences $(s^j, a^j, r^j, s^{j+1})$ from $B$.
15:      for each $i \in D$, do
16:         Compute TD target Q-values using Eq. (31).
17:         Update the weights $\theta$ by computing a one-step gradient descent on the loss function using Eq. (32).
18:         Update target network $\ell$ steps using Eq. (33).
19:      end for
20:   end for
21: end for
```

$$\{vm_l^{(c)}\}_{i,j}^{t+1} = \begin{cases} \{vm_l^{(c)}\}_{i,j}^{t}, & if, \quad a_{v_{i,j}} = 0 \\ \{vm_l^{(c)}\}_{i,j}^{t}(1 + a_{v_{i,j}}), & otherwise \end{cases} \quad (35)$$

where $\{vm_l^{(c)}\}_{i,j}^{t+1}$ indicates the updated CPU resource of MVNO slice $v_{i,j}$, $\{vm_l^{(c)}\}_{i,j}^{t}$ represents the CPU resource amount initially provisioned to the MVNO slice $v_{i,j}$, and $a_{v_{i,j}}$ is the action executed on an MVNO slice $v_{i,j}$ at decision step *t*. The proposed D3QN-based CPU resource pricing and demand prediction solution for ISCRT and allocation are provided in Algorithm 2. The number of trainable parameters and the number of neural networks employed in our proposed D3QN-based ISCRT algorithm determine its computational complexity. Thus, assuming that the agent's state space and action space, denoted by $\boldsymbol{s}$ and $\boldsymbol{a}$, are identical, the number of trainable parameters of Algorithm 2 is $\mathcal{O}(\boldsymbol{s}+\boldsymbol{a})$. Furthermore, the optimized D3QN-based ISCRT algorithm employs three neural networks. The current network, which by itself consists of two neural networks (one for the estimation of the state and the other for the estimation of the optimal action to

TABLE I: Simulation Parameters

Parameters and Units	**Values**	**Parameters Units**	**Values**
Number of MBSs	1	Packet arriving rate	100 packet/sec
Number of MECSs	1	Minimum data rate	500 kbps
Number of Slices V	5 (applications)	Maximum delay	100 ms
System bandwidth W	20MHz	Packet size (task size)	400 kB
Computing capacity of MECS	3000 CUs of CPU resources	Number of hidden layers	3 (64, 32, 32 neurons each)
Number of VMR categories	5	Number of episodes	3000
Transmit power of MBS	30dBM	Discount factor γ	0.85
BS coverage radius	500m	Replay memory size B	$1x10^6$
Noise power density	-174dBm/Hz	epsilon-greedy ϵ	0.1
Number of users	[APP 1:90, APP 2:100, APP 3:110, APP 4:70, APP 5:80]	Mini batch size D	128
User distribution	Uniform	Learning rate α_σ	0.01

TABLE II: VMR category allocation to different applications (MVNO slices)

MVNO Slice 1 (APP 1)	**MVNO Slice 2 (APP 2)**	**MVNO Slice 3 (APP 3)**	**MVNO Slice 4 (APP 4)**	**MVNO Slice 5 (APP 5)**
VM_0	VM_1	VM_2	VM_3	VM_4

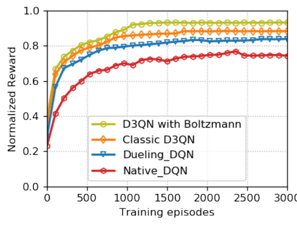
increase training speed and network stability), and the target network. We assume that Ω_i is the number of neurons in the i-th layer of the D3QN network. Because the D3QN-based algorithm is fully connected, its computational complexity can be expressed as $\mathcal{O}(\sum_{i=2}^{i=W-1}(\Omega_{i-1}\Omega_i + \Omega_i\Omega_{i+1}))$, where W represents the number of layers in the networks. As a result, the complexity of the D3QN-based algorithm is $\mathcal{O}((2 \times \psi) \cdot \mathcal{G} \cdot \mathcal{H} \cdot N_b)$, where $\psi = \mathcal{O}(\sum_{i=2}^{i=W-1}(\Omega_{i-1}\Omega_i + \Omega_i\Omega_{i+1}))$, $\mathcal{G}$ indicates the number of episodes, $\mathcal{H}$ is the maximum iteration of each episode, and N_b is the mini-batch sampling size. Furthermore, the existence of an additional network accounts for the number 2 in the complexity of the D3QN-based algorithm.

VI. Performance Evaluation

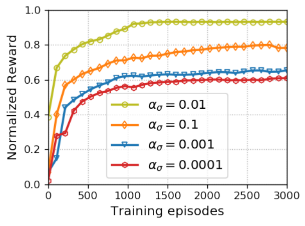
A. Simulation Configuration Scenario and Settings

This section presents the performance evaluation of our proposed blockchain-based enhanced D3QN algorithm through extensive simulations and in-depth analysis of the results. All simulations are carried out following the 5G specifications and recommendations [44]. We run the simulations on a PC with a core i7 CPU running at 2.4 GHz and 16 GB of RAM and a GPU (NVIDIA GeForce GTX 2080Ti) running an Ubuntu 16.04 LTS OS, and we use the Python 3.7 environment, Tensorflow 1.15.0, and the Graphical Network Simulator-3 (GNS-3) platform [45]. The DRL algorithms were implemented in Tensorflow [46], while the GNS-3 platform is used to implement the 5G network simulation environment [47]. Table I provides a summary of all the parameters for our simulation. On the GNS-3 platform, we consider a 500m x 500m coverage area with a single MBS linked to a MECS. The system bandwidth is configured at 20 MHz with 100 PRBs. The MBS transmit power is configured at 30 dBm, assuming insignificant interference. Furthermore, we investigate a log-normal distribution for channel shadow fading and apply the random walk mobility model to forecast MUE mobility across the networks due to a changing MUE population [48]. The MECS with an Intel (R) Core (TM) i7-8550U processor is configured with a capacity of 3000 CUs of CPU resources, and enough storage and intra-bandwidth resources. Five MECS VMR categories denoted as $VM_0 - VM_4$ are configured and with remote execution of web-based services enabled. The CPU resources for $VM_0 - VM_4$ (i. e., $vm_0^{(c)}, vm_1^{(c)}, vm_2^{(c)}, vm_3^{(c)}$, and $vm_4^{(c)}$) are 450, 258, 400, 350, and 300 CUs, respectively. For offloading services, we configure 450 heterogeneous MUE nodes to offload their data to five different applications (APPs) defined as URLLC using the QCI index table [32]. Each APP is treated as a separate slice that an MVNO has deployed. Each MVNO slice (APP) reserves a part of the MECS resources. Table II shows the VMR category assignments to each MVNO slice (APP) in our implementation. The MECS resources for MVNO slices (APPs) are assumed to be heterogeneous. Thus, some MVNO slices (APPs) are allocated more resources than others based on a pre-allocated system and the SLAs between InP and MVNOs.

The consortium blockchain is implemented in our work by setting up a hyperledger Iroha platform [49] that is SC-enabled on the Ubuntu 16.04 LTS OS. The hyperledger Iroha SC for the blockchain platform is deployed to execute the ISCRT scheme securely, accurately, and timely. The hyperledger Iroha SC is coded using a program, often called *Chaincode*. To boost efficiency, we employ a single chaincode that handles all contracts. The chaincode is initialized to handle the transaction ledger states when peer MVNO nodes send ISCRT requests. In the D3QN algorithm, we set the repay memory size to 10^6 and the minibatch size to 128. The discount factor γ, epsilon-greedy ϵ, and learning rate α_σ are set to 0.85, 0.1, and 0.01, respectively, to guarantee consistent performance.

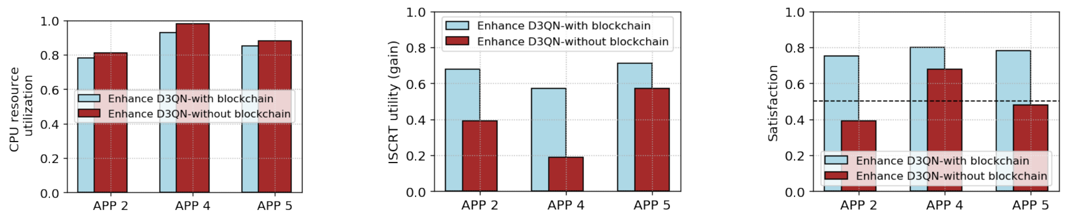


(a) Based on reward



(b) Based on learning rates

Fig. 3: Algorithm convergence analysis



(a) Slice CPU resource usage (b) APPs trading utility (c) Satisfaction of APPs (slices)

Fig. 4: MVNO slice (APP)-level performance

The DNN structure comprises three hidden layers, with the first, second, and third layers having 64, 32, and 32 neurons, respectively. The activation function for the hidden layers is ReLU, and the output layer is the sigmoid function. The AdamOptimizer is used to optimize the loss function, and we perform all of the simulations using 3000 episodes.

B. Convergence Performance Analysis

To evaluate the convergence performance of our proposed D3QN-with-Boltzmann algorithm, we compared it with classic D3QN (i. e., D3QN without the Boltzmann exploration method), dueling DQN, and native DQN based on normalized reward and varied learning rates. The simulations are run for 3000 episodes, with the maximum value taken at every 300 episodes for performance comparison. Fig. 3(a) depicts the convergence of the D3QN-with-Boltzmann, classic D3QN, dueling DQN, and native DQN on normalized reward performance. Fig. 3(b) depicts the effect of varied learning rates on the convergence of the D3QN-with-Boltzmann algorithm.

Fig. 3(a) shows that our proposed D3QN-with-Boltzmann algorithm converges faster at about 1200 episodes as compared to the classical D3QN, dueling DQN, and native DQN. The normalized reward of the enhanced D3QN is 0.93, which is 16% greater than the native DQN due to a better Q-value evaluation. D3QN-with-Boltzmann algorithm can converge roughly 48 episodes earlier than the classic D3QN because of the enhancement with the Boltzmann exploration method, exhibiting a 16% improvement in the convergence time. Overall, the D3QN-with-Boltzmann algorithm's average normalized reward for the pricing and demand problem in the ISCRT framework is around 20% greater than the native DQN. When the results are compared to the optimal solution, our proposed D3QN-with-Boltzmann algorithm achieves higher rewards convergence compared with the other benchmark schemes, where the optimality gap for the proposed enhanced D3QN algorithm is about 1.93%, which is less than the classic D3QN (2.26%), dueling DQN (3.68%), and native DQN (4.59%). Since the D3QN-with-Boltzmann algorithm method gives the fastest convergence and the maximum reward value, it is further evaluated with different learning rates. According to Fig. 3(b), convergence occurs at each learning rate indicated, with $\alpha_\sigma = 0.01$ yielding the highest normalized reward. Thus, we can conclude that our proposed algorithm obtains better convergence by using a higher learning rate. On the other hand, this isn't true in every situation, as the algorithm structure and state-space influence whether or not α_σ is a good decision.

C. Analysis of MVNO Slice (APP)-level Performance

We evaluate the performance of our enhanced D3QN algorithm with blockchain in terms of the MVNO slice CPU resource utilization, ISCRT utility, and MVNO slice QoS satisfaction for this experiment. For ISCRT transactions, we compare the performance of our proposed algorithm with and without the blockchain platform. For ease, we analyze three MVNO slices (APP 2, APP 4, and APP 5) as buyers in the ISCRT utility simulation in this experiment and examine the MVNO slices' performance. We note that the MVNO slice CPU resource utilization, ISCRT utility, and QoS satisfaction are determined using Eq.(6), Eq.(9) or Eq.(10), and Eq.(7) or Eq.(8), respectively. Fig. 4(a), Fig. 4(b), and Fig. 4(c) indicate the CPU resource utilization, ISCRT utility, and QoS satisfaction performance levels of the three buyer MVNO slices (APPs), respectively.

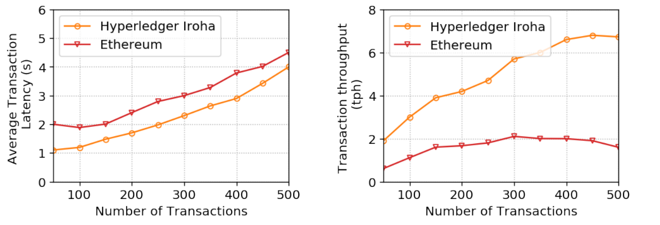
As indicated in Fig. 4(a), we observe that the CPU resource utilization levels of APP 2, APP 4, and APP 5 under the enhanced D3QN with blockchain are around 0.76, 0.90, and 0.82, respectively. However, under the enhanced D3QN algorithm without the blockchain trading platform, we achieve higher CPU resource utilization levels, with APP 2 at approximately 0.81, APP 5 (0.88), and APP 4 at 0.98. The results obtained in Fig. 4(b) show that the enhanced D3QN algorithm with the blockchain platform allows the MVNO slices (APPs) to optimize their profitability better than enhanced D3QN without the blockchain platform. In Fig. 4(c), the proposed algorithm indicates higher QoS satisfaction levels for all the three MVNO slices (APPs) under consideration, which passes the minimal satisfaction requirement level of 0.5. The satisfaction levels for APP 2, APP 4, and APP 5 are approximately 0.77, 0.81, and 0.79, respectively. Under the enhanced D3QN without blockchain, the satisfaction levels decline, with APP 2 and APP 5 slices attaining values of roughly 0.39 and 0.49, respectively, which are below the minimal MVNO slice satisfaction threshold value of 0.5. So, in comparison to enhanced D3QN without the blockchain platform, our proposed blockchain-based enhanced D3QN algorithm balances MVNO slice QoS satisfaction and CPU resource utilization better while maximizing profits.

D. Slice (APP)-level Performance Comparison of Our Proposed Algorithm against Benchmark Schemes

This simulation evaluates the APP-level performance of our proposed D3QN-with-Boltzmann algorithm in terms of

TABLE III: Performance Comparison of Proposed Algorithm against Benchmark Schemes with Blockchain

Name of Algorithm	**Performance Metrics**								
	VMR (CPU resource) Utilization			ISCRT Utility (gain)			QoS Satisfaction		
	APP 2	APP 4	APP 5	APP 2	APP 4	APP 5	APP 2	APP 4	APP 5
Native DQN	0.95	1.00	0.98	0.29	0.12	0.44	0.25	0.40	0.34
Dueling DQN	0.91	1.00	0.95	0.33	0.14	0.48	0.29	0.46	0.39
Classic D3QN	0.84	0.99	0.91	0.60	0.48	0.67	0.72	0.72	0.73
Enhanced D3QN	**0.76**	**0.90**	**0.82**	**0.68**	**0.57**	**0.71**	**0.77**	**0.81**	**0.79**



(a) Latency analysis (b) Throughput analysis

Fig. 5: Blockchain performance analysis

MVNO slice QoS satisfaction, CPU resource utilization, and ISCRT utility compared to benchmark schemes. For ISCRT transactions, we compare the performance of our proposed algorithm to that of classic D3QN, dueling DQN, and native DQN, all using the blockchain trading platform. For ease, we analyze APP 2, APP 4, and APP 5 as buyers in the ISCRT utility simulation in this experiment and examine the three-performance metrics for the slices from the heavy-load perspective.

Table III presents the results in terms of the three performance metrics for APP 2, APP 4, and APP 5, used in evaluating the performance of our enhanced D3QN algorithm compared to the benchmark schemes. We observe that the CPU resource utilization values of APP 2, APP 4, and APP 5 under classic D3QN, dueling DQN, and native DQN are higher than the CPU resource utilization values attained under the enhanced D3QN algorithm. For example, under classic D3QN, the CPU resource utilization values attained for APP 2 are approximately 0.84, APP 5 is 0.91, and APP 4 is 0.99. Under native DQN and dueling DQN, APP 4 attains 100% CPU resource utilization (i. e., a value of 1.00). Concerning the ISCRT utility metric, the enhanced D3QN algorithm enables APP 2, APP 4, and APP 5 to optimize their profitability better than the benchmark schemes, and native DQN obtains the lowest ISCRT utility values for all the MVNO slices (APPs). Regarding the metric for QoS satisfaction, we observe that the satisfaction levels for APP 2, APP 4, and APP 5 decline considerably under dueling DQN and native DQN schemes compared to the satisfaction values attained under classic DQN and the enhanced D3QN algorithms. For example, under the dueling D3QN, APP 2, APP 4, and APP 5 slices attain satisfaction levels of roughly 0.29, 0.46, and 0.39, respectively, which are below the minimal MVNO slice satisfaction threshold. This implies that APP 2, APP 4, and APP 5 do not satisfy some MUEs at this time under dueling D3QN. Under classic D3QN and the enhanced D3QN, all the APP slices attain QoS satisfaction values that pass the minimal satisfaction threshold of 0.5. Thus, we can conclude that the enhanced D3QN and classic D3QN algorithms can attain acceptable MVNO slice-level QoS satisfaction values compared to dueling DQN and native DQN. However, the enhanced D3QN algorithm attains a slightly better QoS satisfaction level than the classic D3QN algorithm. In general, we can conclude that our proposed algorithm compared with the benchmark schemes attains better MVNO slice CPU resource utilization, an appreciable QoS satisfaction level, and higher profits in terms of ISCRT utility.

E. Blockchain Performance Comparison Analysis

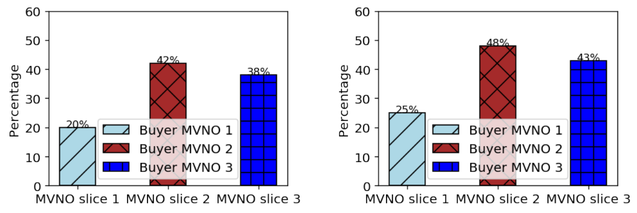
This subsection compares the gains of our proposed consortium blockchain-based algorithm for ISCRT on the hyperledger Iroha platform with SC using the lightweight PBFT consensus to the Ethereum platform with the Casper consensus protocol [50]. We installed the Eth Geth 1.10.8-stable application on the Ethereum platform to connect to the Ethereum main network. And we deployed the solidity compiler locally to compile the SC and generate the private chain. In terms of latency and throughput, the performance of our algorithm employing hyperledger Iroha and Ethereum is analyzed. Transaction latency is indicated as the time taken in executing one transaction process (in seconds). Transaction throughput is the total number of transactions the blockchain platform can handle in a certain period, which in this experiment is per hour.

Fig. 5(a) and Fig. 5(b) compare the latency and throughput performance of our work's hyperledger Iroha-consortium blockchain to the Ethereum-consortium blockchain network. Fig. 5(a) indicates that when the transaction volume increases, the transaction latency of both the Hyperledger Iroha and Ethereum systems increases. The transaction latency of the Hyperledger Iroha is significantly lower compared to Ethereum due to the volume of transactions. Furthermore, as the quantity of transactions increases, so does the disparity in transaction latency between the two platforms. Two factors account for this: (i) time required for negotiations in the ISCRT framework; and (ii) time required to vote on, verify, and commit to an ISCRT transaction using the lightweight PBFT protocol, or in the case of Ethereum, the Casper consensus algorithm. The more transactions there are, the longer it takes to negotiate and accept ISCRT transactions on the blockchain. According to Fig. 5(b), the Hyperledger Iroha platform outperforms Ethereum. When the transaction volume exceeds a threshold of 100, the throughput of both the Hyperledger Iroha and Ethereum platforms increases to a point and then decreases. In our experiment, Hyperledger Iroha achieves maximum throughput when ISCRT transaction volume is 450, while Ethereum achieves maximum when transaction volume is 300. A performance bottleneck develops when the transaction volume exceeds the threshold of 300 to achieve maximum throughput. Furthermore, when the transaction volume increases, the throughput of both the hyperledger and Ethereum improves gradually with a transaction volume of less than 300.

F. Analysis of Double Spending on Unused CPU resources

This simulation examines the impact of allocating the same unused CPU resource to more than one buyer MVNO slice on the aggregate available unused CPU resources of a seller MVNO slice. This is referred to as “double-spending”in wireless network resource allocation. During the ISCRT, we set an underloaded MVNO slice (seller) having an unused 100 CUs of CPU resources and three overloaded MVNO slices (buyers). After the ISCRT transaction, we monitored the update of CPU resource allocation to the various buyer MVNO slices. We compared the performance of the enhanced D3QN algorithm with blockchain to that of an identical scheme without the blockchain in terms of updating and allocating the unused CPU resources to the overloaded MVNO slices (buyers). It is worth noting that we took a snapshot of just a single decision cycle for an easy evaluation. It is assumed that all accessible unused CPU resources of the seller MVNO slice are available for trading. So, all of a seller’s unused CPU resources are traded and assigned to three MVNO slices (buyers).

Fig. 6(a) depicts the enhanced D3QN algorithm using the blockchain platform, and Fig. 6(b) depicts the algorithm without the blockchain platform for ISCRT. The results in Fig. 6(a) show that using the blockchain platform, the three buyer MVNO slices are allocated 20, 42, and 38 of the unused CPU resources, respectively, at a certain MVNO slice update and allocation period following the ISCRT transaction. This demonstrates that the total available unused CPU resources allocated to all buyers are equal to the seller’s total available unused CPU resources throughout the ISCRT transaction period. In Fig. 6(b), the buyers’ allocated unused CPU resources are 25, 48, and 43, respectively, which shows that the total number of allocated unused CPU resources exceeded the total available unused CPU resources from the seller MVNO slice(APP) by 16%. Thus, 16 unused CUs of CPU resources were given to different MVNO slices (buyers) at different negotiating intervals, resulting in multiple allocations. We conclude that by using the enhanced D3QN algorithm with the blockchain platform, the double spending problem is prevented by only allocating the available unused CPU resources to each buyer once over the course of an MVNO slice CPU resource update for a specific ISCRT transaction.



(a) No double spending (b) Double spending

Fig. 6: Double spending impact analysis

VII. Conclusion

The paper proposed a new blockchain-based DRL scheme for a well-defined and secure ISCRT system to allocate unused CPU resources of MVNOs in an autonomous multi-sliced MEC-enabled 5G RAN. Using a consortium blockchain enabled with hyperledger SCs, we establish ISCRT transactions between underloaded MVNOs (resource sellers) and overloaded MVNOs (resource buyers), which considers the matching of pricing and demand in a two-stage SLMF Stackelberg game model. Then, we designed a novel enhanced DRL-based ISCRT scheme that determines the optimal demand and pricing policies of buyers and sellers, respectively, for autonomous slice resource allocation of unused CPU resources. After that, we adjusted the slice CPU resource at the MECS level to reflect the change in the previous MVNO slice CPU resource allocation. The demand and pricing problem for the unused CPU resource was formulated as an MDP and solved using the D3QN algorithm with the Boltzmann exploration optimization technique to improve the exploration approach. With extensive simulation results, we demonstrated the superiority of the proposed enhanced D3QN technique over other baseline DRL schemes. Our proposed D3QN-with-Boltzmann algorithm for ISCRT maximizes players’ profit while balancing QoS satisfaction and CPU resource utilization of MVNO slices. The D3QN-with-Boltzmann algorithm combines well with blockchain technology to address the issue of double-spending attacks during ISCRT in MEC-based 5G NS. Future work will consider an enhanced privacy protection scheme for participant operators in ISCRT through anonymous trading and an improved consensus process for our blockchain implementation.

References

[1] L. U. Khan, I. Yaqoob, N. H. Tran, Z. Han, and C. S. Hong, “Network slicing: Recent advances, taxonomy, requirements, and open research challenges,” *IEEE Access*, vol. 8, pp. 36 009–36 028, 2020.

[2] M. Chahbar, G. Diaz, A. Dandoush, C. Cérin, and K. Ghoumid, “A comprehensive survey on the e2e 5g network slicing model,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 49–62, 2021.

[3] H. D. Chantre and N. L. Saldanha da Fonseca, “The location problem for the provisioning of protected slices in nfv-based mec infrastructure,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1505–1514, 2020.

[4] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, “A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art,” *IEEE Access*, vol. 8, pp. 116 974–117 017, 2020.

[5] J. Feng, Q. Pei, F. R. Yu, X. Chu, J. Du, and L. Zhu, “Dynamic network slicing and resource allocation in mobile edge computing systems,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7863–7878, 2020.

[6] Z. Wang, Y. Wei, F. R. Yu, and Z. Han, “Utility optimization for resource allocation in multi-access edge network slicing: A twin-actor deep deterministic policy gradient approach,” *IEEE Transactions on Wireless Communications*, vol. 21, no. 8, pp. 5842–5856, 2022.

[7] A. Ksentini and P. A. Frangoudis, “Toward slicing-enabled multi-access edge computing in 5g,” *IEEE Network*, vol. 34, no. 2, pp. 99–105, 2020.

[8] A. Antonopoulos, "Bankruptcy problem in network sharing: Fundamentals, applications and challenges," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 81–87, 2020.

[9] U. Akgül, I. Malanchini, and A. Capone, "Dynamic resource trading in sliced mobile networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 220–233, 2019.

[10] G. O. Boateng, D. Ayepah-Mensah, D. M. Doe, A. Mohammed, G. Sun, and G. Liu, "Blockchain-enabled resource trading and deep reinforcement learning-based autonomous ran slicing in 5g," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 216–227, 2022.

[11] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5g network slicing using sdn and nfv: A survey of taxonomy, architectures and future challenges," *Computer Networks*, vol. 167, p. 106984, 2020.

[12] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke, "Blockchain technologies for the internet of things: Research issues and challenges," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2188–2204, 2019.

[13] M. A. Togou, T. Bi, K. Dev, K. McDonnell, A. Milenovic, H. Tewari, and G.-M. Muntean, "Dbns: A distributed blockchain-enabled network slicing framework for 5g networks," *IEEE Communications Magazine*, vol. 58, no. 11, pp. 90–96, 2020.

[14] M. Tahir, M. H. Habaebi, M. Dabbagh, A. Mughees, A. Ahad, and K. I. Ahmed, "A review on application of blockchain in 5g and beyond networks: Taxonomy, field-trials, challenges and opportunities," *IEEE Access*, vol. 8, pp. 115 876–115 904, 2020.

[15] D. B. Rawat and A. Alshaikhi, "Leveraging distributed blockchain-based scheme for wireless network virtualization with security and qos constraints," in *2018 International Conference on Computing, Networking and Communications (ICNC)*, 2018, pp. 332–336.

[16] Y. Gong, S. Sun, Y. Wei, and M. Song, "Deep reinforcement learning for edge computing resource allocation in blockchain network slicing broker framework," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021, pp. 1–6.

[17] Z. Ding, Y. Huang, H. Yuan, and H. Dong, "Introduction to reinforcement learning," in *Deep reinforcement learning*. Springer, 2020, pp. 47–123.

[18] K. Yue, Y. Zhang, Y. Chen, Y. Li, L. Zhao, C. Rong, and L. Chen, "A survey of decentralizing applications via blockchain: The 5g and beyond perspective," *IEEE Communications Surveys Tutorials*, vol. 23, no. 4, pp. 2191–2217, 2021.

[19] B. Nour, A. Ksentini, N. Herbaut, P. A. Frangoudis, and H. Moungla, "A blockchain-based network slice broker for 5g services," *IEEE Networking Letters*, vol. 1, no. 3, pp. 99–102, 2019.

[20] L. Zanzi, A. Albanese, V. Sciancalepore, and X. Costa-Pérez, "Nsbchain: A secure blockchain framework for network slicing brokerage," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[21] T. M. Ho, N. H. Tran, S. M. Ahsan Kazmi, and C. S. Hong, "Dynamic pricing for resource allocation in wireless network virtualization: A stackelberg game approach," in *2017 International Conference on Information Networking (ICOIN)*, 2017, pp. 429–434.

[22] X. Lin, J. Wu, S. Mumtaz, S. Garg, J. Li, and M. Guizani, "Blockchain-based on-demand computing resource trading in iov-assisted smart city," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1373–1385, 2021.

[23] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han, "Cloud/fog computing resource management and pricing for blockchain networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4585–4600, 2019.

[24] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 33–39, 2018.

[25] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[27] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian, "Resource trading in blockchain-based industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3602–3609, 2019.

[28] Y. Dai, D. Xu, S. Maharjan, Z. Chen, Q. He, and Y. Zhang, "Blockchain and deep reinforcement learning empowered intelligent 5g beyond," *IEEE Network*, vol. 33, no. 3, pp. 10–17, 2019.

[29] M. Wu, K. Wang, X. Cai, S. Guo, M. Guo, and C. Rong, "A comprehensive survey of blockchain: From theory to iot applications and beyond," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8114–8154, 2019.

[30] D. B. Rawat, "Fusion of software defined networking, edge computing, and blockchain technology for wireless network virtualization," *IEEE Communications Magazine*, vol. 57, no. 10, pp. 50–55, 2019.

[31] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, and Y. Zhang, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4660–4670, 2019.

[32] M. Mamman, Z. M. Hanapi, A. Abdullah, and A. Muhammed, "Quality of service class identifier (qci) radio resource allocation algorithm for lte downlink," *PloS one*, vol. 14, no. 1, p. e0210310, 2019.

[33] N. Afraz, M. Ruffini, and H. Ahmadi, "Hyperledger blockchain-based distributed marketplaces for 5g networks," *Wireless Blockchain: Principles, Technologies and Applications*, pp. 117–136, 2021.

[34] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 840–852, 2018.

[35] Z. Su, Y. Wang, Q. Xu, M. Fei, Y.-C. Tian, and N. Zhang, "A secure charging scheme for electric vehicles with smart communities in energy blockchain," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4601–4613, 2019.

[36] N. Chondros, K. Kokordelis, and M. Roussopoulos, "On the practicality of practical byzantine fault tolerance," in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 2012, pp. 436–455.

[37] T. D. Tran and L. B. Le, "Resource allocation for multi-tenant network slicing: A multi-leader multi-follower stackelberg game approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8886–8899, 2020.

[38] J. Qiu, D. Grace, G. Ding, J. Yao, and Q. Wu, "Blockchain-based secure spectrum trading for unmanned-aerial-vehicle-assisted cellular networks: An operator's perspective," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 451–466, 2020.

[39] H. Xu, X. Qiu, W. Zhang, K. Liu, S. Liu, and W. Chen, "Privacy-preserving incentive mechanism for multi-leader multi-follower iot-edge computing market: A reinforcement learning approach," *Journal of Systems Architecture*, vol. 114, p. 101932, 2021.

[40] G. Sun, K. Xiong, G. O. Boateng, G. Liu, and W. Jiang, "Resource slicing and customization in ran with dueling deep q-network," *Journal of Network and Computer Applications*, vol. 157, p. 102573, 2020.

[41] H. Dong, H. Dong, Z. Ding, S. Zhang, and Chang, *Deep Reinforcement Learning*. Springer, 2020.

[42] A. Iqbal, M.-L. Tham, and Y. C. Chang, "Double deep q-network-based energy-efficient resource allocation in cloud radio access network," *IEEE Access*, vol. 9, pp. 20 440–20 449, 2021.

[43] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, "Boltzmann exploration done right," *Advances in neural information processing systems*, vol. 30, 2017.

[44] A. M. et. al. "5g-picture, d2.1 5g and vertical services, use cases and requirements, version 2.0," 2022. [online]. available:. https://www.5g-picture-project.eu/.

[45] D. C. da Silva, M. Antonio Firmino de Sousa, G. Bressan, and R. M. Silveira, "5g network slice selector in iot services scenarios with qos requirements guarantee," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 90–95.

[46] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," ser. OSDI'16. USA: USENIX Association, 2016, p. 265–283.

[47] F. G. Lavacca, P. Salvo, L. Costantini, E. Mammi, S. Pompei, L. Rea, and M. Teodori, "Studying and simulation of a ns3 frameworktowards a 5g complete network platform," in *2019 International Workshop on Fiber Optics in Access Networks (FOAN)*, 2019, pp. 62–67.

[48] K.-H. Chiang and N. Shenoy, "A random walk mobility model for location management in wireless networks," in *12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications. PIMRC 2001. Proceedings (Cat. No.01TH8598)*, vol. 2, 2001, pp. E–E.

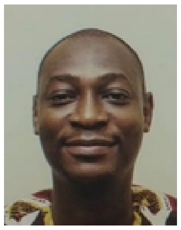
[49] S. Aggarwal and N. Kumar, "Chapter sixteen - hyperledgerworking model." in *The Blockchain Technology for Secure and Smart Applications across Industry Verticals*, ser. Advances in Computers, S. Aggarwal, N. Kumar, and P. Raj, Eds. Elsevier, 2021, vol. 121, pp. 323–343.

[50] A. Iskakova, H. S. V. S. Kumar Nunna, and P. Siano, "Ethereum blockchain-based peer-to-peer energy trading platform," in *2020 IEEE International Conference on Power and Energy (PECon)*, 2020, pp. 327–331.

Thomas Kwantwi received his MEng (Computer Complexes Systems and Networks) from Tver State Technical University (TSTU), Tver-Russia in 2008. He's currently a Ph.D. candidate at the University of Electronic Science and Technology of China (UESTC), Chengdu, Sichuan, China 2019 to date. He majors in Computer Science and Technology at the School of Computer Science and Engineering of UESTC. Since 2010, been working at the University of Mines and Technology (UMaT), Ghana as a lecturer in the department of computer science and engineering. He is also a member of the Mobile Cloud-Net Research Team, at UESTC. His research interests are 5G wireless networks, Mobile Computing, Artificial Intelligence, UAV networks, the Internet of Things, and 5G network slicing.

Guolin Sun received his B.S., M.S., and Ph.D. degrees, all in Comm. and Info. System from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2000, 2003, and 2005. Since he finished his Ph.D. study in 2005, Dr. Guolin has had eight years of industrial work experience in Information and Communication Techniques (ICT) research and development for LTE, Wi-Fi, Internet of Things, Cognitive radio, Localization, and navigation. Before joining the UESTC, as an Associate Professor in Aug. 2012, he was with Huawei Technologies, in Stockholm, Sweden. Dr. Guolin Sun has filed over 40 patents, published over 70 scientific conferences and journal papers, and acted as a TPC member and keynote speaker at many conferences. His general research interest is artificial intelligence, network virtualization, edge computing, blockchain techniques, resource management, and vehicle networks.

Gerald Tietaa Maale received his Bachelor's degree in Information Technology Education from the University of Education, Winneba, Ghana. He also received his M.S. degree in Information Technology from the Kwame Nkrumah University of Science and Technology, Kumasi, Ghana. He is currently pursuing his Ph.D. degree in Computer Science and Technology at the University of Electronic Science and Technology of China (UESTC), Chengdu, China. His research interests include wireless networks, deep learning, content caching, and the internet of things (IoT). He is a student member of IEEE.



Noble Arden Elorm Kuadey (GS'20) received the B.Sc. (Hons.) degree in Computer Science from the University of Cape Coast (UCC), Cape Coast, Ghana in 2007, the Master of Science in Engineering (Information Systems) degree from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway in 2010. Mr. Kuadey is currently a Ph.D. candidate at the University of Electronic Science and Technology of China majoring in Computer Science and Technology in the School of Computer Science and Engineering. He is a member of the International Association of Engineers and a Lecturer in the Department of Computer Science, Ho Technical University, Ho, Ghana. His research interest includes 5G wireless networks, Network Slicing, the Internet of Things, and M-Learning in Higher Education.

Guisong Liu received his B.S. degree in Mechanics from the Xi'an Jiao Tong University, Xi'an, China, in 1995, and the M.S. degree in Automatics and the Ph.D. degree in Computer Science from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2000 and 2007, respectively. He was a Visiting Scholar at Humbolt University, Berlin, Germany, in 2015. Before 2021, he was a Professor at the School of Computer Science and Engineering, at the University of Electronic Science and Technology of China (UESTC). He is currently a Professor and the Dean of the School of Computing and Artificial Intelligence, at the Southwestern University of Finance and Economics, Chengdu. He has filed over 20 patents and published over 70 scientific conferences and journal papers. His research interests include pattern recognition, neural networks, and machine learning.