Reinforcement Learning assisted Adaptive difficulty of Proof of Work (PoW) in Blockchain-enabled Federated Learning

Prateek Sethi

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

Luiz Antonio Pereira da Silva, Chair

Aloizio Pereira Da Silva

Thinh Thanh Doan

Susanna Pirttikangas

July 12, 2023

Arlington, Virginia

Keywords: Blockchain, Machine Learning, Federated learning, Reinforcement Learning, MEC, 5G

Reinforcement Learning assisted Adaptive difficulty of Proof of Work (PoW) in Blockchain-enabled Federated Learning

Prateek Sethi

(ABSTRACT)

This work addresses the challenge of heterogeneity in blockchain mining, particularly in the context of consortium and private blockchains. The motivation stems from ensuring fairness and efficiency in blockchain technology's Proof of Work (PoW) consensus mechanism. Existing consensus algorithms, such as PoW, PoS, and PoB, have succeeded in public blockchains but face challenges due to heterogeneous miners. This thesis highlights the significance of considering miners' computing power and resources in PoW consensus mechanisms to enhance efficiency and fairness. It explores the implications of heterogeneity in blockchain mining in various applications, such as Federated Learning (FL), which aims to train machine learning models across distributed devices collaboratively. The research objectives of this work involve developing novel RL-based techniques to address the heterogeneity problem in consortium blockchains. Two proposed RL-based approaches, RL based Miner Selection (RL-MS) and RL based Miner and Difficulty Selection (RL-MDS), focus on selecting miners and dynamically adapting the difficulty of PoW based on the computing power of the chosen miners. The contributions of this research work include the proposed RL-based techniques, modifications to the Ethereum code for dynamic adaptation of Proof of Work Difficulty (PoW-D), integration of the Commonwealth Cyber Initiative (CCI) xG testbed with an AI/ML framework, implementation of a simulator for experimentation, and evaluation of different RL algorithms. The research also includes additional contributions in

Open Radio Access Network (O-RAN) and smart cities. The proposed research has significant implications for achieving fairness and efficiency in blockchain mining in consortium and private blockchains. By leveraging reinforcement learning techniques and considering the heterogeneity of miners, this work contributes to improving the consensus mechanisms and performance of blockchain-based systems.

Reinforcement Learning assisted Adaptive difficulty of Proof of Work (PoW) in Blockchain-enabled Federated Learning

Prateek Sethi

(GENERAL AUDIENCE ABSTRACT)

Technological Advancement has led to devices having powerful yet heterogeneous computational resources. Due to the heterogeneity in the compute of miner nodes in a blockchain, there is unfairness in the PoW Consensus mechanism. More powerful devices have a higher chance of mining and gaining from the mining process. Additionally, the PoW consensus introduces a delay due to the time to mine and block propagation time. This work uses Reinforcement Learning to solve the challenge of heterogeneity in a private Ethereum blockchain. It also introduces a time constraint to ensure efficient blockchain performance for time-critical applications.

Dedication

I dedicate my thesis to the memory of my late grandmother, whose unwavering belief in me and relentless encouragement have been a guiding light throughout my academic journey. You always saw the potential in me, even when I doubted myself. Your constant words of encouragement and support gave me the strength to pursue my dreams in engineering. Although you are no longer with us, your spirit lives on in my heart and inspires me daily. This thesis is dedicated to you as a tribute to your unwavering faith in my abilities. My parents, whose love, sacrifices, and unshakable belief in my abilities, have driven my accomplishments. Your guidance, wisdom, and unconditional support have shaped me into who I am today. To my younger brother, whose trust, enthusiasm, and unbounded support have reminded me of the joy of learning. To my loving girlfriend, whose unwavering support, understanding, and encouragement have been instrumental in my academic journey. Your belief in me and endless words of motivation have kept me going despite challenges. Finally to my friends and colleagues who have helped me balance the stresses of work with moments of joy and adventure. I dedicate this thesis to you for motivating me through your achievements and the competitive environment that has helped me surpass my boundaries.

Acknowledgments

I would like to express my deepest gratitude and appreciation to all the individuals who have supported and guided me throughout this challenging yet rewarding journey of completing my thesis. This endeavor would not have been possible without my advisor Dr. ALoizio P. DaSilva and my committee members, Dr. Luiz, Dr. Susaana, and Dr. Thinh, and my fellow researchers Dr. Mayukh Roy Chowdhury and Tri Nguyen. Their expertise, encouragement, and constructive feedback have been instrumental in shaping the direction and quality of this work. I am truly fortunate to have had their mentorship. I am indebted to the staff and fellow researchers at Commonwealth Cyber Initiative, Virginia Tech, who provided me with their unwavering support and assistance. Their contributions in facilitating access to resources, technical assistance, and administrative support have been invaluable. My heartfelt thanks go to my girlfriend, friends, and family, who have been my pillars of strength throughout this journey. Their belief in my abilities and constant encouragement has motivated me during challenging times. Their love and support have been my anchor, and I am forever grateful to them. In conclusion, this thesis represents the culmination of the efforts and contributions of numerous individuals. I am deeply thankful to each and every person who has played a role, no matter how big or small, in making this work a reality. Thank you.

Contents

List of Figures

List of Tables

Acronyms

pBFT Practical Byzantine Fault Tolerance 8

PoA Proof of Authority 8

PoB Proof of Burn ii

PoS Proof of Stake ii

PoW Proof of Work ii

PoW-D Proof of Work Difficulty ii

PPO Proximal Policy Optimization 49

RAM Random Access Memory 27

RAN Radio Access Network ii

ReLU Rectified Linear Units 27

RFCU Reward For Compute Utilized 42

RIC RAN Intelligent Controller 4

RL Reinforcement Learning

RL-MDS RL based Miner and Difficulty Selection ii

RL-MS RL based Miner Selection ii

RMR RIC Message Router 63

SARSA State–action–reward–state–action 35

sb3 Stable-Baselines3 49

SDR Software Defined Radio 4

SGD Stochastic Gradient Descent 27

UE User Equipment 63

VM Virtual Machine 16

VR Virtual Reality 67

Chapter 1

Introduction

1.1 Motivation

Blockchain technology relies on Peer-to-Peer (P2P) ledgers that require a mechanism to maintain security and ensure consensus. Various algorithms exist for this purpose, such as Proof of Work (PoW), Proof of Stake (PoS), and Proof of Burn (PoB). PoW has succeeded in two well-known blockchain implementations: Nakamoto PoW for Bitcoin [1], a cryptocurrency use case, and Ethash for Ethereum [2], a blockchain-based smart contract platform. In a PoW consensus, participants must dedicate significant computational power to solve a cryptographic puzzle per consensus round. The difficulty of PoW in large public blockchains, such as Ethereum, is adjusted based on the time the winning miner takes to mine the last block since the system is unaware of the computation power in the blockchain. Intuitively, the efficiency and fairness of the PoW consensus mechanism can be improved by considering the miners' computing power. As per the Ethereum yellow paper [2], the PoW consensus mechanism is used for two main purposes:

1. PoW should be accessible to many people.

2. Miners should be unable to make super-linear profits, especially with a high initial barrier. Such a mechanism would allow a well-funded adversary to gain a troublesome amount of the network's total mining power. As such, it would give them a super-linear

reward (thus skewing distribution in their favor) and reduce network security.

Consortium and private blockchains are permissioned blockchains that provide greater visibility of the machines in the setup as opposed to public blockchains. The miners need access to the ledger to participate in the blockchain. These machines might be dedicated to more than just blockchain operations and can be utilized for other purposes, which introduces heterogeneity in computing power. Additionally, limited devices exist in a private or consortium blockchain; hence, a device with more computing resources is more likely to win the mining process. This weakens the second requirement and leads to the well known vulnerability of the PoW consensus algorithm - the 51% attack [3]. The 51% attack occurs when a group of miners controls over 51% of the resources, which is easier to replicate in a private blockchain. This challenge surfaces in many applications that use blockchain to share data and to ensure transparency and privacy. To our knowledge, this challenge has not been considered in any other work.

The challenge of heterogeneity of compute resources can be seen in applications such as Federated Learning (FL), a popular ML technique. ML has gained much attention [4] in areas such as healthcare [5, 6], image detection [7, 8, 9], voice recognition [10, 11, 12] and big data [13]. Due to a large amount of data and the distributed nature of this data, FL suffers from challenges such as high communication costs, data aggregation, and data privacy issues [14, 15]. FL [16] tackles the security, data privacy, and data aggregation challenges, enabling multiple devices to learn a shared model collaboratively. However, the single aggregation server and communication costs in a FL system pose a significant challenge to the effectiveness and efficiency of the system.

Blockchain-enabled FL has been used to tackle these challenges [17] [18] by creating a decentralized and secure environment for sharing the model parameters and updating the shared model in FL. The miners in the setup have varying resources, such as computing power and

memory. Recent works have focused on data privacy, security, and heterogeneity of the clients for training the FL model [18]. But most of these works consider only the public blockchain and do not consider the heterogeneity of miners. This thesis focuses on challenges arising from the heterogeneity in compute of the miners in a consortium or private blockchain due to the many heterogeneous devices in the network.

1.2 Research Objectives and Contributions

Public blockchain gained attention with the introduction of Bitcoin and Ethereum. Later, the concept of consortium and private blockchains started gaining interest, especially in the Industrial Internet of Things (IIoT) setting. As private and consortium technologies are adopted, they can expand to more areas, such as smart cities. Multiple heterogeneous devices can be utilized in such setups to run blockchain-enabled Federated Learning. Hence, the challenge of heterogeneity of miners will scale quickly as this technology is adapted to more and more use cases. *To this end, in this work, we tackle the heterogeneity problem in a consortium blockchain to provide fairness amongst the miners. We propose a RL based approach which decides the miners and the difficulty of PoW based on the computing power of the selected miners.* The major contributions of this work are summarized below:

Contribution 1: Two novel RL-based techniques are proposed for miner selection and dynamic adaptation of Proof of Work Difficulty (PoW-D) in a private blockchain setting to ensure fairness in the mining process by modifying the Ethereum code - RL based Miner Selection (RL-MS) and RL based Miner and Difficulty Selection (RL-MDS)

Contribution 2: We contribute modifications to the Ethereum code for dynamic adaptation of difficulty of PoW in a consortium blockchain setting. We replace the built-in

mechanism for predicting PoW difficulty with the proposed RL agent.

Contribution 3: We integrate the CCI [19] xG testbed [20] with an AI/ML framework to assist in deploying the proposed RL solution. This framework can orchestrate and deploy any ML service.

Contribution 4: We designed and implemented a PoW consensus simulator for experimentation based on real data collected through the blockchain deployment on the CCI xG testbed.

Contribution 5: We evaluated different RL algorithms on the novel PoW consensus simulator and performed comparative analysis.

The work in **[Contribution 1-3]** is submitted to IEEE GLOBECOM[1], titled - *"Fair Consensus in Blockchain with Heterogeneous Miners using Reinforcement Learning aided Adaptive Proof of Work"*

In addition to the thesis's core contributions, the additional contribution includes:

Contribution 6: A published paper, titled *"AI/ML Data-driven Control Loop for Managing O-RAN SDR-based RANs"*, accepted as an IEEE INFOCOM 2023 demonstration paper[2]. The specific contributions of the paper include integrating an intelligent rApp and the O1 interface for the first end-to-end O-RAN demo using SDR and open-source srsRAN[3] stack.

Contribution 7: A journal paper yet to be submitted, titled *"Closing the Loop for End-to-end O-RAN Experimentation: Holistic Integration of Near- and Non-Real Time RICs"*.

[1]https://globecom2023.ieee-globecom.org/
[2]https://infocom2023.ieee-infocom.org/
[3]https://www.srsran.com/

Additionally, we identify smart cities as a context in which this approach can be applied. This study proposes two architectures to implement blockchain-enabled FL and intelligently decide the mining nodes and the PoW-D consensus mechanism used in the blockchain [21]. We integrate the solution with the CCI xG testbed to demonstrate a real-life implementation.

1.3 Thesis Organization

This thesis comprises five chapters, each serving a distinct purpose. Chapter 2 provides background and an overview of critical topics relevant to the research, highlighting the state-of-the-art research in the respective topics. Chapter 3 describes the proposed system model and the problem formulation to tackle the heterogeneity in miners in the blockchain. Chapter 4 outlines the metrics utilized to measure the proposed model's performance and presents the research's outcome, including results and implications. Finally, Chapter 5 summarizes the work conducted in the thesis, highlighting overall contributions and outlining future directions and opportunities for further exploration.

Chapter 2

Background and Related Work

2.1 Background

To better understand the research presented in this work, we will introduce some important concepts related to blockchain technologies, Federated Learning (FL), and Reinforcement Learning (RL). The main contributions of the work are on blockchain, and blockchain-enabled Federated Learning is the underlying application. We then introduce smart cities, the context in which this approach is applied, followed by an overview of the CCI xG testbed, where we deploy the proposed model. Finally, this chapter presents state-of-the-art research on these broad topics that closely relate to this work.

2.1.1 Federated Learning

FL was first introduced by Google [16] in 2017. It is widely accepted by the community and applied in various use cases [22]. We use FL as an example of the many possible applications that use blockchain for adding privacy and resilience. FL is a distributed technique of training an ML model using heterogeneous devices that carry out training locally. These devices can use their local data and train part of the model based on the initial parameters shared with the device. The trained parameters are then shared with the central server and integrated with the global model. The integration uses the Federated Averaging Algorithm [23]. All the

devices in the system can download the global model. Once downloaded, the global model can be modified per the device's requirements using local information and parameters. This approach uses the secure Aggregation Protocol [24] to share model updates and to ensure individual updates from the devices cannot be accessed before they are averaged. This is illustrated in Figure 2.1.

Since the aggregation is done only on the central server, multiple challenges are associated with FL [22]. These include having a single point of failure, privacy protection, high communication costs, and system heterogeneity [25]. Blockchain (BC) can mitigate these shortcomings and provide redundancy and increased security (as further discussed in Section 2.1.3).

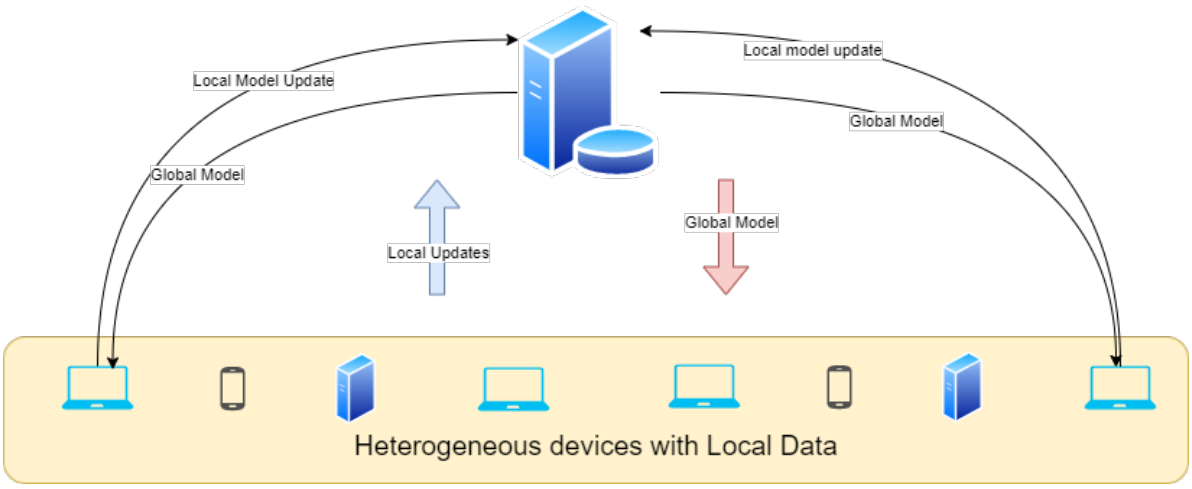


Figure 2.1: FL architecture.

2.1.2 Blockchain

Blockchain is a distributed, decentralized, and immutable ledger technology [26] that simultaneously stores and updates data in multiple devices. The transactions in the ledger are transparent and cannot be modified. It was introduced by Satoshi Nakamoto in 2008

[1] as a form of digital currency – bitcoin. [27]. Vivekanadam [27] provides a detailed survey of trends and applications of blockchain. Blockchain technologies are no longer limited to just the finance domain [28]. They are used for applications such as Distributed Application (Dapp), insurance, energy, mobility, and logistics. This is due to the benefits of using blockchain [28], such as data traceability, systems interoperability, flexibility, and token-based ecosystems.

There are different flavors of blockchain. They can be categorized into public and blockchain governance [28]. The governance can further be classified into consortium and private. A detailed analysis of blockchain technologies can be found in [29]. The consortium blockchain is studied in detail, along with a case study on Ethermint, in [28].

[29] shows that there are many advantages of the PoW consensus mechanism. This includes its truly decentralized nature. In every round, the miners compete to solve a complex puzzle, and the blockchain participants verify this puzzle's result to reach a consensus. All the blockchain participants can participate in the consensus process, making it highly decentralized. In contrast, consensus algorithms such as Proof of Authority (PoA) and Practical Byzantine Fault Tolerance (pBFT) rely on a predetermined set of validators, which limits the number of participants and introduces a centralized aspect to the consensus mechanism. Moreover, PoW has succeeded in two widely used implementations of blockchain - Bitcoin [1] and Ethereum [2], which marks its success and acceptance by the industry. Hence, PoW can fit a private or consortium blockchain setting well.

Some shortcomings in using the PoW consensus mechanism; include its vulnerability to the 51% attack [3] and its high resource and power consumption [30]. This work tackles these shortcomings and introduces fairness in the system. This is discussed in detail in Section 3.1.

Due to consistency and decentralization, blockchain technology's usage confronts performance-

related issues, especially in public blockchains. With the well-known blockchain-based smart contract, Ethereum enables decentralized application and decentralized autonomous organization. Therefore, during the last decade, many studies have been empowered by Ethereum for a transition toward a decentralized environment via trust and autonomous organization [2]. However, despite interesting Ethereum features with the blockchain-based smart contract, many studies face performance issues, especially from PoW, a consensus mechanism asking for common agreement among network participants for consistency. PoW requires network participants to join a puzzle-solving competition: the leader at each consensus round acts as the decider for the next system state. Thus, the rules for selecting an acceptable leader per consensus round are extremely vital. The time for solving this puzzle varies with difficulty and can affect the application's performance on the blockchain. The proposed approach aims to constraint this time between a lower and an upper bound, as discussed in Section 3.

Regarding the PoW difficulty value formation [2], the difficulty value of a block header H is adjusted as $D(H)$ (Equation 2.1) via two prime parts, adjustment ς_2 and bomb ϵ. The adjustment is to dynamically modify the homeostasis of time between blocks based on the parent information $P(H)$, while ϵ, called "difficulty bomb" or the "Ethereum Ice Age," affects the $D(H)$ by increasing the difficulty of mining blocks after a period, especially the 100,000 blocks. The details for this perspective are following the yellow paper [2] with H_d as the expected time for the difficulty, H_s as the time stamp, and H'_{i} as the result of subtracting three million from the current block number.

$$D(H) \equiv \begin{cases} 2^{34} & \text{if} \quad H_{\mathrm{i}} = 0 \\ \max(D_{\min}, P(H)_{\mathrm{H_d}} + x \times \varsigma_2 + \epsilon) & \text{otherwise} \end{cases} \tag{2.1}$$

where:

$$D_{\min} \equiv 2^{17} \tag{2.2}$$

$$x \equiv \left\lfloor \frac{P(H)_{\mathrm{H_d}}}{2048} \right\rfloor \tag{2.3}$$

$$\varsigma_2 \equiv \max\left(y - \left\lfloor \frac{H_{\mathrm{s}} - P(H)_{\mathrm{H_s}}}{9} \right\rfloor, -99\right) \tag{2.4}$$

$$y \equiv \begin{cases} 1 & \text{if} \quad \|P(H)_{\mathbf{U}}\| = 0 \\ 2 & \text{otherwise} \end{cases} \tag{2.5}$$

$$\epsilon \equiv \left\lfloor 2^{\left\lfloor H'_{\mathrm{i}} \div 100000 \right\rfloor - 2} \right\rfloor \tag{2.6}$$

$$\tag{2.7}$$

The distributed nature of blockchain makes it an excellent fit for applications that require interactions between dispersed architectures [28]. Federated Learning is a technique discussed in Section 2.1.1. The training data from different nodes can be efficiently shared to train the overall model. Sharing these model parameters via the blockchain reduces the latency and provides fault tolerance and scalability. It also makes it more secure, as the node whose training parameters will be added to the chain is unknown until the end. This makes it difficult for the attackers to compromise the overall model.

2.1.3 Blockchain-enabled Federated Learning

Hyesung Kim et al. introduced the architecture for blockchain-enabled Federated Learning [31], or BlockFL. In this architecture, the devices can update FL parameters, aggregate the local updates, and download the global model without having a single central server. The authors study the end-to-end learning completion latency of BlockFL and share in-

sights about optimizing the block generation rate considering scalability and the robustness of the blockchain. The vanilla FL [32] [33], and the proposed model can be seen in Figure 2.2. This model has many advantages, such as its truly distributed nature, where no single point of failure exists.

However, the miners in the architecture are uniformly randomly selected from a pool of identical miners. If the miners have variable computational resources and memory, this will affect the entire implementation of BlockFL. Suppose the difficulty of the PoW is not adjusted depending on the available miners. In that case, the miner with the most resources will likely win if the mining difficulty is too high compared to the average resources of the miners in the system. The mining process will take a very long time.

In our proposed approach, we adjust the PoW-D based on not only the mining time of the winning miner (Equation 2.1) but also the overall computational resources of all the miners. This approach is discussed in detail in the following chapters.

2.1.4 Reinforcement Learning

Reinforcement Learning (RL) is a form of trial and error learning which consists of two main entities: the environment and the agent. The environment encompasses the surroundings within which the agent exists and interacts. In this work, the Ethereum blockchain deployed using Geth forms the environment. The agent can see a part of the information about the world, which is a part of the state known as observation. If the agent can see the entire state, it is called a fully observable environment; else, it is called a partially observable environment. We use the logs of the miners to derive the required information and transform it into useful parameters. This information is passed to the RL agent that is developed as part of the contributions of this work. The state may change due to the actions taken by the

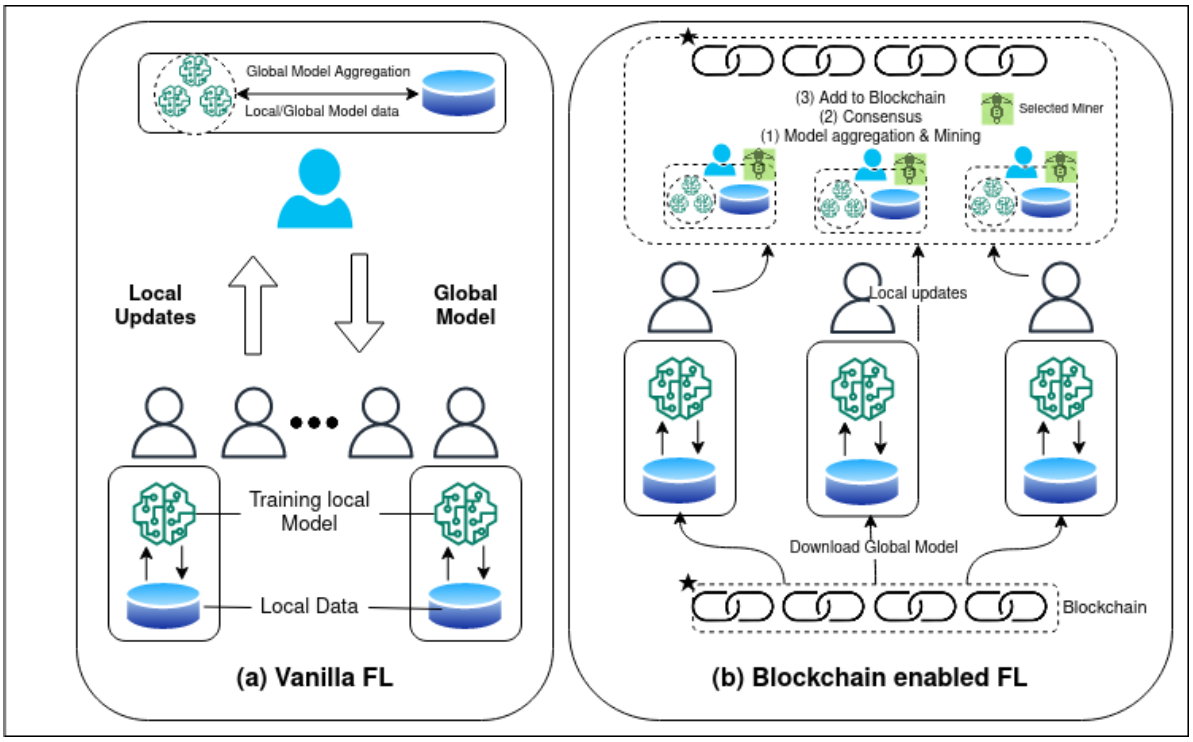


Figure 2.2: BlockFL.

agent or on its own. The state is usually represented using a real-valued vector, matrix, or higher-order tensor. Based on the state, the agent can take different actions. The collection of all these actions is known as the action space. The agent's policy maps the state and action space and decides which action to choose. The agent's main objective is to maximize the overall cumulative reward or value function. [34]. The model and implementation are introduced in Chapter 3, and the RL loop is illustrated in Figure 2.3.

2.1.5 Smart Cities

Recent times have seen significant growth in the number of smart cities and smart hubs, especially in developing countries [35]. The Smart City Observatory [36] provides an in-depth analysis of smart cities globally, defining a smart city index along with the index

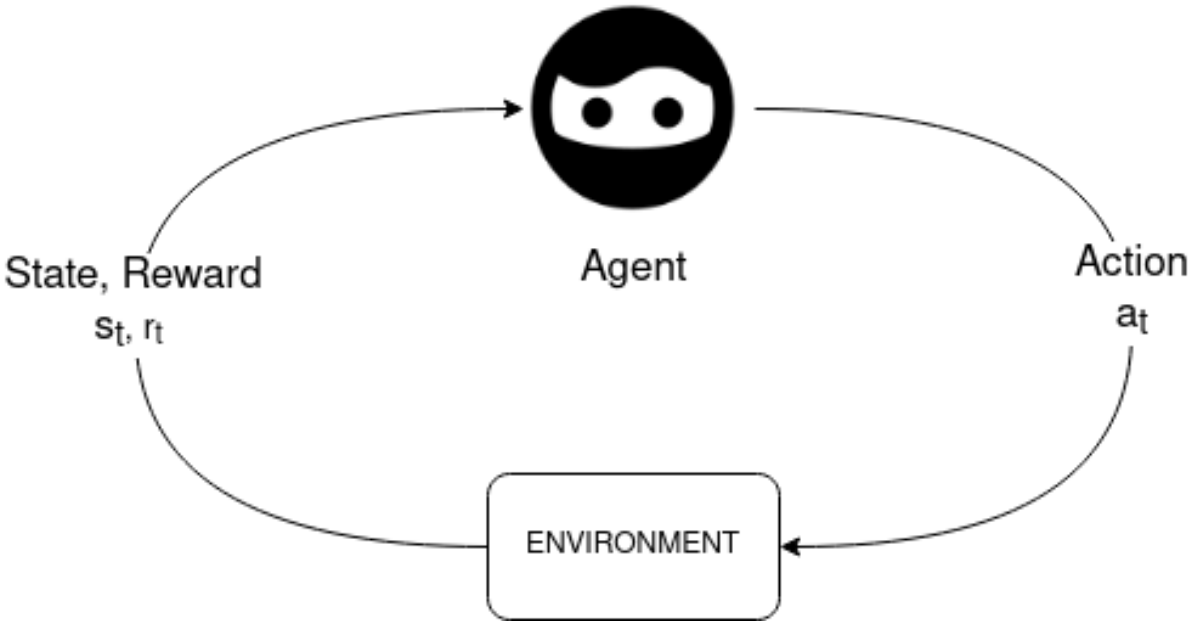


Figure 2.3: Reinforcement learning loop.

methodology and a global ranking of smart cities. For this study, we considered multiple surveys [37] [38] to identify the most common services in a smart city. These services include smart hospitals, smart transportation, smart buildings, smart grid, and smart agriculture. All these services are a combination of multiple sub-services. We will consider one smart hub corresponding to each service, but in reality, a smart city can comprise multiple such hubs.

- *Smart Hospitals:* Healthcare has been a major topic for research regarding smart hubs. Recent studies [39][40] have deeply delved into the type of services present in a smart hospital and have mostly found nine categories: registration, nurse stations, Emergency Department (ED), Imaging, Operating Room (OR), Intensive Care Unit (ICU), patient room, lab, and pharmacy. All these services have different requirements for resources in terms of computing and memory and also have varying high time and low time. High time is the time interval in a day when the service is extensively utilized. Low

time is the time during the day when the service is partially being used or not being used.

- *Smart Agriculture:* Smart IoT sensors are being deployed in large numbers to monitor factors affecting agriculture. These include but are not limited to weather sensors such as temperature, humidity, and rainfall [41][42][43][44][45]. They encompass various parameters such as soil fertility, mineral composition, water retention levels, etc. This data is collected and transferred to data warehouses and data lakes for processing. Various ML algorithms can be applied to the data to make informed decisions that increase agriculture yield. Though the data is mainly collected continuously, the processing and analysis are not always performed. Hence these facilities also have some low time. During this time, the compute resources are not utilized and can be used for other purposes like model training or mining.

- *Smart Transportation:* Automotive and transportation have been innovative fronts for a while [46]. Companies like Tesla [47] have been challenging the norm and developing technologies to improve and make the experience safer. Recent works have tried to suggest architectures for smart transportation [48][49], and a lot of ML applications have also been applied to transportation [50]. In doing so, modern vehicles have several sensors, computational power, and memory. Additionally, a communication network supports the functioning of the features embedded in these machines. With developments in 5G and 6G networks, communication has become more reliable and faster with low latency. More and more vehicles are moving towards an intelligent ecosystem. This ecosystem requires device on-board computational resources, memory, and support from high computing facilities like edge servers. These vehicles are not using all the available resources all of the time. For example, when an electric vehicle is charging, it has a suitable power source, and most of its functions are not being

utilized; hence the computational resources and memory are free and can be utilized by a distributed system to carry out model training.

- *Smart Grids:* The smart power grid is one of the most power-wealthy systems in a smart city [51][52]. It consists of sensors that monitor the trend of current and voltage in different areas in a city, state, or even the entire country. Many recent works have used the power grid information for anomaly detection [53][54][55], intrusion detection [56][57], and predicting blackouts and malicious nodes [58][59]. The smart meters used for this purpose are resource-rich and can be utilized during their low time for trading ML models.

- *Smart Homes:* Smart devices have become commonplace in many households [60][61], ranging from smart speakers such as Alexa to smart refrigerators and machines. Many invest in other smart devices, such as smart cameras and lighting. All this is managed by a private home network where most of the computation is offloaded to the cloud. As more and more devices are connected to this ecosystem, there will be a need for a stronger communication network. The services must be migrated to the edge cloud from the central cloud to provide low latency and higher throughput. Hence, edge servers must be installed to support technologies such as millimeter waves. These servers will be used to run these smart services. Depending on the services and buildings associated with these servers, the availability of these edge servers will follow a trend of high and low time. The resources at these servers can be utilized more efficiently during their low time to carry out model training.

2.1.6 CCI xG Testbed

The CCI xG Testbed [20] is created and managed by the CCI [19] and is available to CCI researchers, industry partners, and government agencies. The testbed prioritizes programmability and interoperability, using open interfaces and open-source software. The primary focus areas are securing 5G and next-generation mobile networks and AI assurance. Core principles of the CCI xG Testbed include openness, accessibility, programmability, interoperability, and modularity. The testbed allows researchers to design, develop, prototype, and deploy solutions for next-generation mobile networks. It features advanced SDR nodes, fiber-optic back-haul networks, and edge and core cloud computing infrastructure. It consists of two sites:

- Radio Network (Arlington, VA): An indoor setup consisting of more than 72 SDR and the deployment of open-source software solutions.

- Citizens Broadband Radio Service (CBRS) Private Network (Blacksburg, VA): An outdoor setup of real base stations deployed at multiple sites.

We have used the cloud capabilities of the indoor testbed to deploy our setup and to collect results. The deployment is discussed in Chapter 3. The indoor setup consists of a ceiling rack with 72 SDRs. The backhaul consists of two core switches and two access switches. The computing power is distributed amongst one controller server, and four compute servers connected using fiber cables and 100GB Ethernet. The VMs, containers, and networking are orchestrated and managed using OpenStack. This can be seen in Figure 2.4.

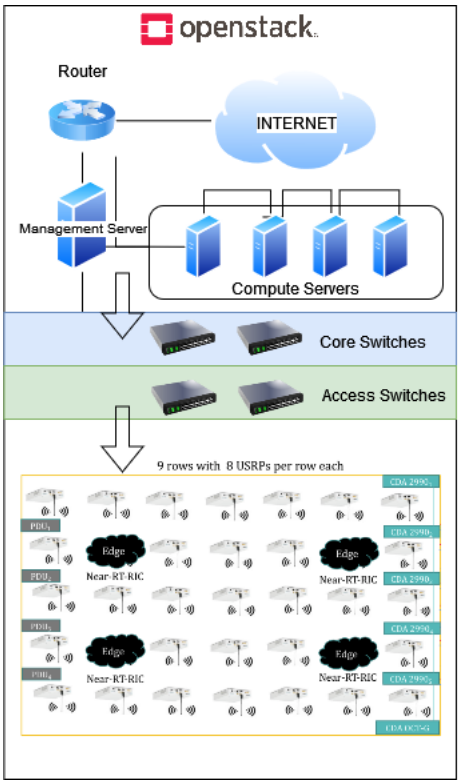


Figure 2.4: CCI xG testbed cloud infrastructure.

2.2 Related Work

FL is a machine learning technique that allows training models on data distributed across multiple devices without aggregating the data on a central server, as discussed in Section 2.1.1. This approach can help preserve data privacy and security and allow for training models on devices with limited computational resources. There is extensive research on FL, covering various aspects – McMahan et al. [62] present a novel approach for FL, which allows for the training of deep networks on decentralized data without aggregating the data on a central server. Minder et al. [63] explore the use of FL for mobile keyboard prediction, showing that it can be used to train accurate models without compromising user privacy. The authors in [64] focus on FL with non-Independent and identically distributed (IID) data and propose a novel approach that improves performance. Niknam et al. [65] explore

the potential and challenges of using FL in wireless communications. The authors highlight the benefits of FL in wireless communications, such as improved privacy protection and performance, but also discuss challenges, such as data heterogeneity and security and privacy concerns that must be addressed. The authors in [66] provide an overview of using FL in smart healthcare. The paper discusses the concepts, algorithms, and use cases of FL in smart healthcare, the challenges and limitations, and future research directions. In [67], Ali et al. present a FL approach for privacy protection in context-aware recommender systems. The authors propose a FL method to train a context-aware recommender model while preserving user privacy and evaluating its performance compared to traditional ML techniques.

The challenges in FL include data heterogeneity, communication efficiency, privacy, and security. These challenges have been discussed in several works [65, 66, 68], which provide a good overview of the current challenges and suggest possible solutions to address these challenges. These papers highlight future directions and opportunities in FL. Recent works provide valuable insights into the potential applications of blockchain technology (section 2.1.2) and suggest areas for future research. They demonstrate the potential for blockchain to revolutionize various industries and improve privacy, security, and efficiency in various systems and applications. These include finance [1, 69, 70] - Zhang et al. [71] identify the challenges facing blockchain in finance and economics and propose countermeasures to address these challenges, Marko Budišić et al. [72] provides a comprehensive review of the literature on the blockchain. Aditi et al. [73] discuss the opportunities and challenges of using blockchain technology for digital financial inclusion. Cimoli et al. [74] provide a systematic review of the impact of blockchain technology on the financial industry. [75] gives an analytical overview of the future of blockchain in the banking and financial services industry. Another area of interest is healthcare: [76] provides an overview of blockchain's history and current state in healthcare and discusses its potential future applications. Mohammad

et al. [77] review the current state of blockchain-based Electronic Health Record Systems (EHR) systems, highlighting their potential benefits and challenges. In [78], the authors propose a blockchain-based system for secure health information exchange. The system aims to ensure privacy and security in exchanging health information. Further, O'Ceallaigh et al. [79] provide a systematic review of the potential applications of blockchain technology in healthcare, including its use in Electronic Health Record Systems (EHR) systems, drug supply chains, and clinical trial management. Other applications include governance, supply chain management [80, 81, 82], energy, IoT [83], real estate, social media, and digital identity. These works highlight the relevance and scope of FL and blockchain. All these works collectively show blockchain's acceptance and widespread adoption to add qualities such as security, transparency, resilience, and immutability to various applications. Since our work ensures fairness in the blockchain consensus mechanism, all these applications benefit from this work.

Blockchain-enabled FL (section 2.1.3), or BlockFL, was first introduced by Hyesung Kim et al. [31]. The authors propose a blockchain-based system for conducting on-device FL. While the paper provides a detailed analysis of the system's latency, the study has several shortcomings. First, the evaluation is limited and does not comprehensively analyze the system's performance. Second, the proposed system has not been deployed in a real-world setting, making it unclear how it would perform under practical constraints. Third, the system may face scalability issues as the number of participating devices increases. Lastly, blockchain implementation is relatively simplistic and may not be suitable for more complex applications. [84] presents a model that optimizes the algorithm to ensure the protection of user data. [85] introduces DeepChain, a secure and decentralized deep learning framework based on blockchain that uses an incentive mechanism and provides confidentiality, audibility, and fairness in gradient in gradient collecting and collaborative decryption. In

[17] the authors present an algorithm to adjust local iterations in FL scenarios with limited resources to achieve better model performance and evaluate its performance in a simulated environment. [86] presents an approach called FedCS for client selection to train FL models on heterogeneous mobile edge. They aim to make FL more efficient. Our study is more generic and not limited to blockchain-enabled FL. [87] explores the opportunities of FLchain in Mobile Edge Comute (MEC) networks.

Many works [88] [89] have focused on comparing the performance of the consensus algorithms in the private blockchain continuum. The PoW consensus algorithm has the major advantage of being completely decentralized as compared to PoA, PoS, and pBFT. Still, it is shadowed by its two major cons - high resource and power consumption and its vulnerability to the 51% attack [3]. The 51% attack refers to an attack on the blockchain when more than 51% of the resources are controlled by a group of miners. It is easy to replicate this attack in a private blockchain by a single miner that is significantly more powerful. Significant research has been done to reduce this attack's effect. [90] proposes a 2-hop blockchain method that combines PoW and PoS to prevent fraudulent actions by miners with more than 51% mining power. [91] proposes a deep learning model to detect anomalies in blockchain systems using a sequence-by-sequence neural network model. [92] proposes a method to mitigate 51% attacks on PoW blockchains using weighted historical information to establish if a branch change is required. The proposed protocol is called Historical Weighted Difficulty - Proof of Work (HWD-PoW) and increases the cost of an attack by two orders of magnitude. These approaches' advantages, risks, vulnerabilities, and costs and other state-of-the-art techniques are summarized in [3]. Most of these works have been applied to the public blockchain, whereas our approach considers the private blockchain and reduces the win percentage to mitigate the effects of the 51% attack. We further limit this winning percentage below a fairness factor, as discussed in Chapter 3. Since, in the proposed approach, only some of the

miners are allowed to mine at a given time, it also reduces the power consumption of the machines.

The Ethereum Yellow paper[2] states that a miner's chances of winning a round are directly proportional to their device's computing power. Miners in public blockchains continually increase their computing power and form mining pools to increase their chances of winning. Studies show [93] that less than ten mining pools control almost 95% of Bitcoin's mining power, while less than six control around 80% of Ethereum's mining power. However, this race for more resources leads to wasteful power consumption. It takes an average of 125.36 kWh of electricity to mine ether and 1777.11 kWh for bitcoin [30]. This massive requirement for power poses a significant challenge in integrating blockchain with IoT devices [94]. Public blockchains increase their difficulty level to maintain an average block time of twelve seconds in Ethereum, the minimum time required for a block to propagate in a peer-to-peer network[95]. However, other studies suggest that the average block mining time in Ethereum is around fifteen seconds [96]. The vanilla PoW difficulty adjustment function maintains an average time of about 12 seconds. A complicated puzzle can be solved in an extensive amount of time, but it is sometimes solved in close to a second. This gap nullifies the effect of block mining values, which can be pretty high. While it maintains the average target time, significant mining times can be problematic for time-critical applications. We propose a time constraint to address this issue to adjust the PoW-D considering the application's time-criticality 3.4.

2.3 Summary

The first part of this chapter introduces the building blocks of this work that include: FL, Blockchain along with PoW, and Blockchain-enabled FL. Further, RL is discussed in detail,

which is the technique used by our work to solve the problem that is the focus of this thesis. Later, smart cities are discussed, which is the context in which the proposed approach can be applied. Finally, we describe the CCI xG testbed where the setup is deployed to verify the implementation and generate results.

The related work section mentions how FL preserves data privacy and security and allows training models on devices with limited computational resources. Recent research on FL covers various aspects, including non-IID data, wireless communications, smart healthcare, and context-aware recommender systems. It emphasizes that blockchain technology has the potential to revolutionize various industries and improve privacy, security, and efficiency in multiple systems and applications. The section mentions that Blockchain-enabled FL, or BlockFL, has been proposed by several authors to ensure user data protection, security, and fairness. While the potential applications of blockchain technology in various fields are promising, challenges such as heterogeneity of mining devices still need to be addressed.

Chapter 3

System Model and Problem Formulation

This Chapter introduces the proposed approach for enforcing fairness in block mining. The proposed approach achieves fairness by selecting the miners based on availability, mining rounds won, and compute resources. It also adapts the difficulty of the PoW consensus mechanism to match the capabilities of the selected miners. We first formulate the problem and define the approach; then we provide an overview of the system. Then we discuss the implementation and description of the enablers of the proposed system. The following section provides the steps in the process, followed by the formulation of the RL problem. Finally, we discuss the design and implementation of the novel blockchain simulator.

3.1 Problem Definition and Proposed Approach

The devices in the setup described in Section 3.3 have variable resources that can be seen in Table 3.1; this introduces heterogeneity in the system. Due to the heterogeneity, the miners with higher compute have a greater chance of winning the mining rounds. This makes the system unfair and vulnerable to the 51% attack. There is a need to mitigate the effect of heterogeneity and to maintain fairness in the system. To do so, our proposed RL agents maintain a record of the winning percentages of each miner and can understand

which miner is violating the fairness constraint. The fairness constraint is the maximum allowed winning percentage for all miners. To ensure that the miners' winning percentage stays below the fairness factor, the RL agent identifies the pattern of winning depending on the selected miners. Based on this understanding, the RL agent selects a subset of the miners to compete in a given mining round. This gives the miners with lower compute a higher chance of winning a mining round. Hence the fairness in the system is maintained in RL-MS and RL-MDS. Section 3.4 describes RL-MS and RL-MDS and formulates the RL problem.

In addition to the fairness in the system, RL-MDS also maintains a time constraint for time-critical applications. Time is an important factor in applications such as FL where the global model relies on updates from all the hosts training the model locally. Depending on the compute of the device and the PoW-D the time taken to mine a block changes. An inverse relationship exists between the compute and the mining time and a direct relationship between PoW-D and the mining time. This is discussed in more detail in Section 4.1. The RL-MDS agent can understand and model this relationship and choose the right PoW-D value for the selected miners to compete in a particular mining round. Hence RL-MDS maintains the time constraint in the PoW consensus. This is discussed in more detail in Section 4.2. We will refer to the implementation of the enabler defined in Section 3.3 along with the proposed RL agents as the proposed approach.

3.2 System Overview

To explain the model in more detail, this Section will introduce some terminology. A high-level overview of the system can be seen in Figure 3.1.

Clients: The blockchain participants who are not selected as miners. They have local data,

which they use to train a local model. *Local weights:* The clients have training data that is available locally. The clients use this data to train the local model. The weights of this trained model are referred to as the local weights or local parameters for the client. *Miners:* The miners are the blockchain nodes competing to solve the PoW puzzle. The winning miner gets to add the new block to the main blockchain. *Global weights:* The miners are also responsible for aggregating the weights before they add the block to the blockchain. The weights generated by aggregating the participants' local model weights are called the global weights or global parameters. *RL Agent:* The RL agent is an intelligent, state-aware entity that is responsible for selecting the miners and the difficulty of the PoW puzzle based on the current and the past state of the blockchain. *Orchestration agent:* It is responsible to communicate between the RL agent and the blockchain. *Difficulty:* For better readability, we will refer to the difficulty of the PoW consensus algorithm as difficulty or PoW-D.

3.3 Enablers of the proposed system model

The proposed approach comprises three main building blocks: blockchain, Federated Learning (FL), and intelligent orchestration using the RL agent. These will be discussed in detail in the following subsections.

3.3.1 Blockchain

We use a private Ethereum blockchain setup to run blockchain-empowered FL. The setup is deployed using Geth:1.10.12 and Truffle:5.4.28[1] on the CCI xG Testbed. The client-side interacts with the blockchain via WebSocket from NodeJS's web3[2]. In our experiment,

[1] trufflesuite.com/
[2] web3js.readthedocs.io

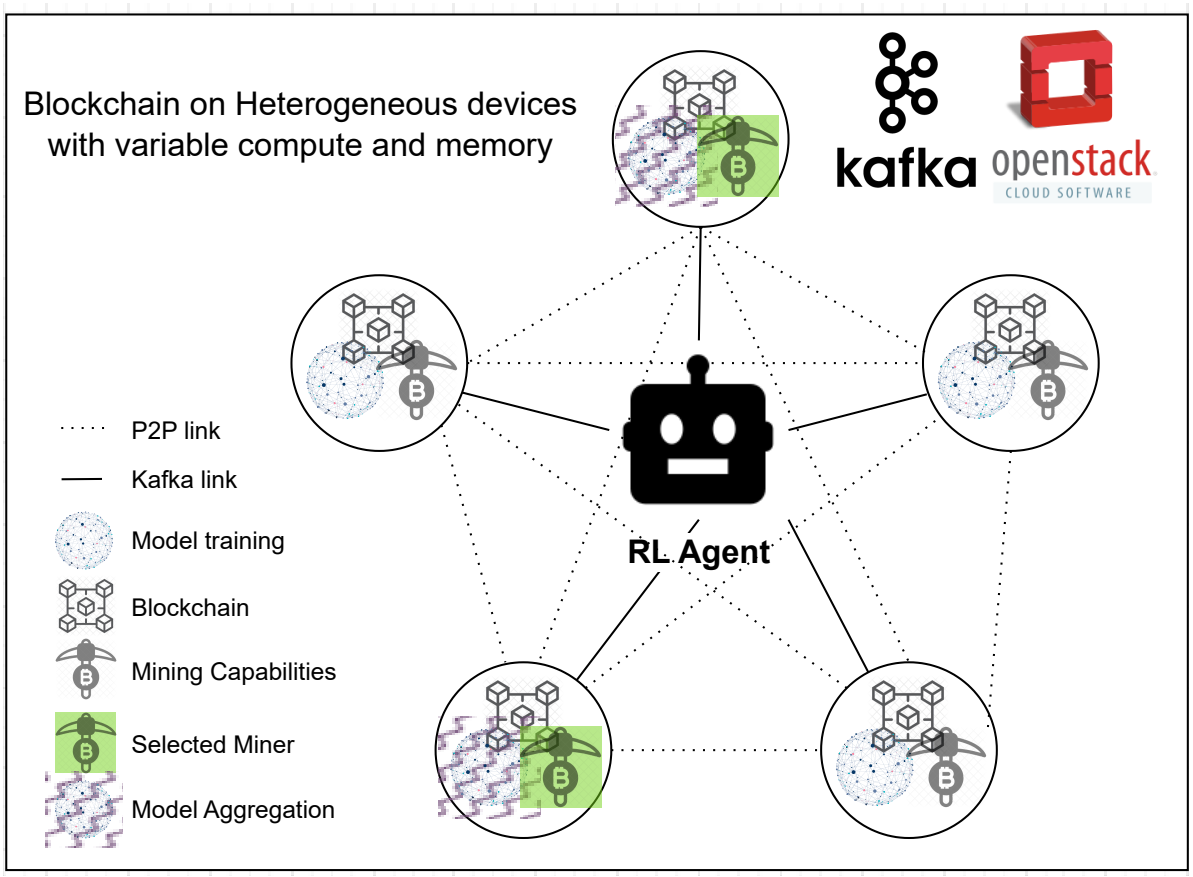


Figure 3.1: High-level architecture of the model.

Ethereum-based FL requires extra configuration to bypass requirements in Ethereum. These include:

- Restriction of transaction size by modifying the txpool.go function;
- Adjusting difficulty value advised by RL by modifying the consensus.go function.

Modification in txpool.go: We modify the transaction and block sizes in the txpool.go file. The original Geth config has a restriction of 128 KB on the transaction size. Our FL local and global parameters are approximately 800 KB. We increased the allowed transaction size to 1 MB to accommodate these updates. The block's size depends on the amount of gas limit in a block which limits the number of transactions. Since the PoW requires solving a

Table 3.1: Experimental configurations for the Blockchain

Machine name	RAM	CPU	Memory	OS
Node1-Miner	6144	4	163840	Ubuntu 22.04
Node3-Miner	16384	8	163840	Ubuntu 22.04
Node5-miner	6144	4	163840	Ubuntu 22.04
Node7-Miner	16384	8	163840	Ubuntu 22.04
Node9-Miner	32768	12	163840	Ubuntu 22.04
Node2-Client	4096	2	163840	Ubuntu 22.04
Node4-Client	4096	2	163840	Ubuntu 22.04
Node6-Client	6144	4	163840	Ubuntu 22.04
Node8-Client	4096	2	163840	Ubuntu 22.04
Node10-Client	4096	2	163840	Ubuntu 22.04

complex puzzle, we keep this value as high as possible to ensure no local updates are lost.

Modification in consensus.go: Several parameters were modified to satisfy the implementation of FL. These include adjusting the minimum difficulty, from 1125000, as per Equation 2.1, to a higher value. This modification is necessary to prevent the miners from losing synchronization. Additionally, we altered the CalcDifficulty function to receive the difficulty value from the RL agent and to update the difficulty of the blockchain. Since our setup runs for less than $100,000$ blocks, we set *homesteadmod* and other parameters for "difficulty bomb" from H_{i}' to 0. These parameters include: "byzantiumBlock", "constantinopleBlock", "eip155Block", "eip150Block", "eip158Block". We configure a Kafka[3] Cluster to communicate between the RL agent and the miners.

Our setup consists of ten VMs running Ubuntu 22.04 created using OpenStack. We used a combination of CPU and RAM to showcase heterogeneity. This can be seen in Table 3.1. The RL agent is hosted in a separate VM.

[3]https://kafka.apache.org/

3.3.2 Federated Learning

For testing FL, we used the Modified National Institute of Standards and Technology database (MNIST) dataset [97], which consists of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9. The MNIST dataset is used commonly for training various image processing systems and fits our use case of FL well. This dataset was divided into testing and training. Ten thousand images belonging to different categories were kept for testing purposes. The remaining 50,000 images were split into ten equal groups and stored on the local memory of the clients in the setup. We used a Sequential Multi-Layer Perceptron (MLP) neural network constructed with an input size of 784, which is the number of pixels in a flattened image from the MNIST dataset. The labels are the digits from 0 to 9. The simple feedforward neural network uses the Keras API[4]. The model consists of three fully connected layers with 200 units each, and ReLU [98] activation functions are used in the first two layers. The final layer uses a softmax activation function to convert the neural network outputs into a probability distribution over the 10 labels. We use the Stochastic Gradient Descent (SGD) as the optimization algorithm and accuracy as the metric to build the model. For the loss function, we use categorical cross-entropy, which is used for multi-class classification tasks. It computes the error between the predicted and true probability distributions. It is commonly used in neural network models trained with the SGD optimizer to minimize the loss and improve the classification accuracy.

Many works have considered the performance of the FL in blockchain-enabled FL ([84] [85][17][86]) and suggested ways to improve the model training, accuracy and have also tackled the problem of heterogeneity of devices for model training. We propose an improvement in the PoW consensus mechanism in a private blockchain, which can be utilized by many applications and is not limited only to FL. Measuring the performance of FL is beyond the

[4]https://keras.io/api/

scope of this work.

Using this model, each client trains a local model based on the data available on the local memory and generates local weights. These local weights form the transactions from each client and are added to the transaction pool. The miners in the blockchain can access these transactions. When all the transactions reach the miners, they aggregate the local weights and form a global model. This global model is evaluated on the testing data on the blockchain. This process continues till the global model achieves the required accuracy.

3.3.3 Intelligent Orchestration

The intelligent orchestration comprises the RL Agent and the Orchestration Agent. The RL Agent interacts with the blochchain using the the Orchestration Agent which uses Kafka to communicate with the miner and the RL Agent.

RL agent: The RL problem is formulated in next section. The agent gets the state information from the orchestration agent using Kafka. This information consists of the various parameters representing the blockchain's state and the mining process's past results. The agent decides the actions based on this state.

Orchestration Agent: The orchestration agent interacts with the blockchain via smart contracts and logs. It gets information about the blockchain from the logs and converts it into a form that can be passed to the RL agent. Based on the decisions the RL agent makes, the orchestration agent modifies the blockchain via the smart contracts. Kafka handles all the communication. The miners have a producer and consumer deployed on them, the RL Agent machine also has a miner and a consumer deployed on it.

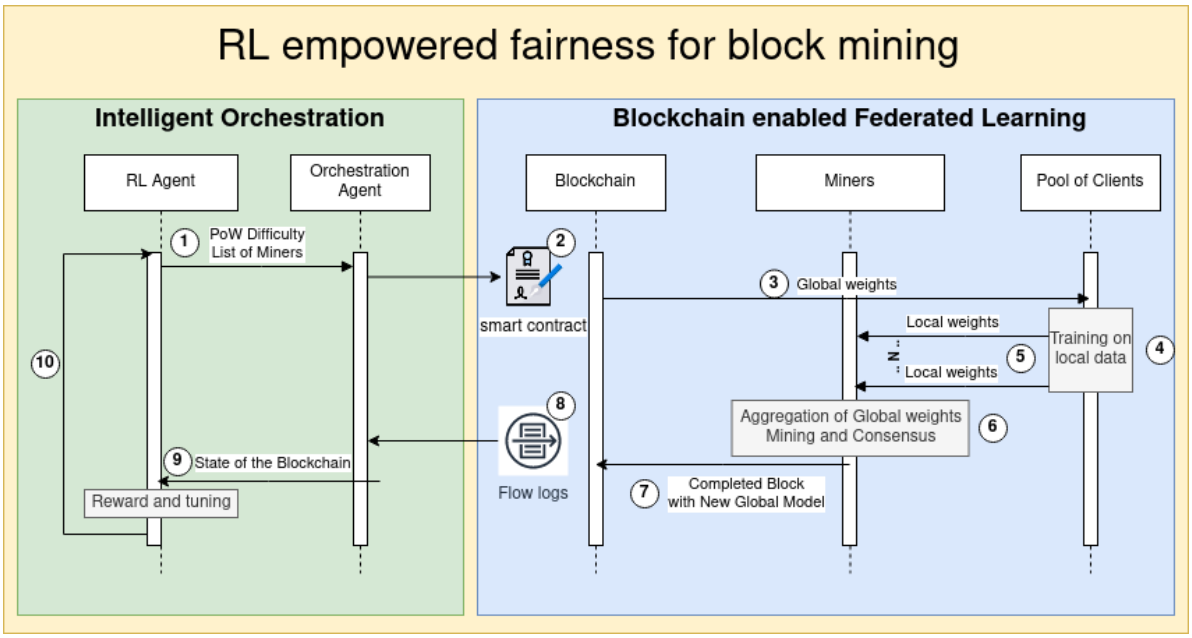


Figure 3.2: Sequence Diagram: RL-empowered fairness for block mining.

3.3.4 RL Assisted Adaptive PoW-D Workflow

The sequence diagram describe the assisted adaptive PoW-D complete process is shown in Figure 3.2. The RL agent decides the difficulty of the PoW algorithm and the miners for the next round of mining (1). This information is passed to the orchestration agent. The orchestration agent interacts with the blockchain through smart contracts (2). We have modified the source code for Ethereum to dynamically change the difficulty value based on the decision of the RL agent. This information is updated on the blockchain. All the clients in the system have access to the shared data, which consists of the global model weights and the testing dataset. The clients download the global weights (3) and create a local model with these weights. The model is trained on the training data available to the respective clients. Once the training is completed, the local weights are added to the transaction pool as transactions (5). The selected miners solve the PoW puzzle whose difficulty is decided by the RL agent (6). The miners wait to get all the model updates. Once all the updates

are available, the miner aggregates the local updates and forms a global model (7). This model is tested on the testing dataset, and the accuracy value is outputted. The miner that first solves the PoW puzzle and wins the consensus adds the block to the blockchain. This process continues till the global model accuracy achieves the desired value. The orchestration agent closely monitors the blockchain's state and behavior through the blockchain logs (8). The orchestration agent parses the logs and passes the useful information to the RL agent (9). The RL agent receives a positive or a negative reward depending on the behavior of the blockchain (10). The agent receives a positive reward if the block mining time is within the constraint and if the win percentage of miners is below a threshold. The agent receives a negative reward otherwise. If the miners wins multiple episodes in succession and gains control, the RL agent can detect this and hence select other miners and the right difficulty values. The RL problem is formulated in the following section.

3.4 RL Problem Formulation

To model the agent's sequential decision-making and maximize the gain, we mathematically formulate the RL problem using the Markov decision process (MDP) framework [99], a mathematical model for decision-making problems. Since the blockchain envirnment is highly dynamic it is important to use MDP for its following properties-

- MDPs offer a formal mathematical framework for representing sequential decision-making problems by defining states, actions, transition probabilities, and rewards.
- The Markov property simplifies the problem by focusing on the current state and action for efficient computation.
- MDPs facilitate optimal decision-making through various algorithms like DQN which

is used in your approach.

- Generalization in MDPs enables RL agents to learn from limited experience and apply knowledge to unseen states this helps the RL agent learn quickly.

Due to its Markov property assumption, we choose the MDP framework. It simplifies the problem by stating that the future state depends only on the current state and action, making the problem more tractable. The functioning of the agent depends not only on the immediate effect of the action on the state but is also affected by the action it takes over a period of time. An MDP is defined by a tuple (S, A, P, R, γ), where S is the State, A is the Action, P is the Policy, R is the Reward, and γ is the Discount factor. These parameters for our setup are described below:

- State S is constituted by the winner matrix $W = [w_{t,n}] \in \mathbf{R}^{\tau_m \times N_m}$ and the block mining time vector $T_{BM} \in \mathbf{R^{1 \times \tau_m}}$, where

$$\begin{aligned} w_{t,n} &= 1, \text{ if } n^{th} \text{ miner won in the } t^{th} \text{ mining round} \\ &= 0, \text{otherwise} \end{aligned} \tag{3.1}$$

  and the elements of the vector T_{BM} represent block mining time in seconds taken by miners in the last τ_m rounds:

$$S = [W|T'_{BM}]^{\tau_m \times N_m + 1}. \tag{3.2}$$

- A is the actions (miner selection and PoW-D selection) the agent can take in each state. The first action factor is the competing miners, given by $A_M = [a_m] \in \mathbf{R}^{N_M \times 1}$

where

$$\begin{aligned} a_m^j = 1, \quad & \text{if the } j^{th} \text{ miner is selected for the next } \tau_m \text{ rounds} \\ & \text{where, } j \in \{1, 2, \cdots, N_m\} \\ = 0, \quad & \text{otherwise} \end{aligned} \tag{3.3}$$

and the second action factor is the difficulty of PoW, given by $A_D \in \{D_{min}, \cdots, D_{max}\}$ where D_{min} and D_{max} are proportional to the least and highest available compute power among the miners, respectively. The difficulty of proof of work is given by A_D, where $A_M = [a] \in \mathbf{R}^{N_m \times 1}$

$$A = [A_M | A_D]^{N_m+1}. \tag{3.4}$$

- R is the reward function that defines the immediate reward the agent receives for taking action in a particular state. In other words, R(s, a, s') is the immediate reward received for transitioning from state s to state s' by taking action a. We model the reward function as:

$$\begin{aligned} r_1 = w_1.10, \quad & \text{if block mining time } t_1 \leq t_{BM} \leq t_2 \\ = -w_2.10 \quad & \text{otherwise} \end{aligned} \tag{3.5}$$

where t_1 and t_2 are the minimum and maximum allowable block mining time for a particular application. We multiply the weights ($w_1 and w_2$) by 10 to make the rewards more noticibly different which helps the RL agent learn better.

$$r_2 = -100 \quad \forall \quad p_w \geq \phi$$
$$\text{where, } p_w \in P_w^{1\times N_m}$$
$$= +100 \quad \text{otherwise} \tag{3.6}$$

$$r = r_1 + r_2 \tag{3.7}$$

When a new block is added to the Ethereum blockchain, the chain waits for 7 rounds to verify if the chain under consideration is the longest [2]. This observation window is required to avoid forking. We use the same observation window size of seven blocks to infer the current state, i.e. $\tau_m = 7$. In Equation 3.6, the vector $P_w^{1\times N_m}$ represents the percentage of rounds won by the miners in the last τ_m rounds, and ϕ is the fairness coefficient specific to the implementation. ϕ can take values from 0 to 1. Intuitively the value of ϕ should be less than 0.5, so no miner can win the majority of rounds. We used the value 0.3 considering 7 rounds of mining. In a five-miner setup, a ϕ value of 0.3 means a miner can win a maximum of two rounds out of seven.

- γ is the discount factor, a number between 0 and 1 that determines the importance of future rewards relative to immediate rewards. A high discount factor means that the agent values long-term rewards more than short-term rewards.

$$\mathrm{P}_{ss'}^{a} = \mathrm{P}[s_{t+1} = s' | s_t = s, A_t = a] \tag{3.8}$$

$$R_s^a = E[R_{t+1} | s_t = s, A_t = a] \tag{3.9}$$

$$\pi(a|s) = \mathrm{P}[A_t = a | s_t = s] \tag{3.10}$$

$$G_t = R_{t+1} + \gamma R_{t+2} + ... + \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.11}$$

The policy function π determines the agent's action (A) in each state. In an MDP, the agent's objective is to discover a policy π (shown in Equation 3.10) that maximizes the return, which is the anticipated sum of discounted rewards over time (as given in Equation 3.11). The Bellman Equations (presented in Equation 3.12) can be derived using MDP and is expressed in the matrix form in Equation 3.13. The Bellman equations separate the value functions into two components: the immediate reward and the discounted value of the successor states:

$$E[R_{t+1} + V(S_{t+1})|s_t = s] \tag{3.12}$$

$$V = R + \gamma P_v \tag{3.13}$$

The RL problem can be solved using various RL algorithms to find an optimal policy. Some popular RL algorithms include Q-learning [100], State–action–reward–state–action (SARSA) [101], and policy gradient methods [102]. Since blockchain cannot be represented as a well-defined environment because of its dynamic nature, our approach uses DQN [103] to tackle the posed problem. Many possible states and actions make it difficult for approaches such as Q-learning to maintain a Q table. In DQN, a deep neural network approximates the Q function, making it a good fit for our work. All the code is available on Github[5].

Now, we introduce two novel RL empowered models - First, we ensure fairness in the system at a fixed difficulty value (i.e. 5 million) corresponding to a time constraint of 20 seconds. Then in the second setup, we dynamically change the difficulty to meet the same time constraint. Both the algorithms RL-MS and RL-MDS are summarized in Algorithm 3.4.

[5]https://github.com/Pseth3/AdaptivePoW

Algorithm 1: Proposed RL-MS and RL-MDS

Initialize and link Geth miners with PoW-D as 1 Million
Geth Clients connect to miners for FL initialization
Start Kafka consumers and producers on both the miners and RL agent
Initialize $Q(s, a) \forall s \in S$ and $a \in A$; $Q(\text{Terminal state}, \cdot) = 0$
for episode = 1 to I **do**
    Send the blockchain state s via Kafka to RL Agent
    for t = 1 to T **do**
        Select Miner if RL-MS, or Miner and Difficulty if RL-MDS
        Choose A from S using policy derived from Q (ϵ-greedy)
        Update all miners through Kafka Producer on RL agent
        Start/Stop miners and adjust PoW-D
        Wait for τ_m mining rounds
        Kafka Producer on miner shares information with RL agent
        Observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
        $S \leftarrow S'$
    end for
end for

3.5 PoW Consensus Simulator

In this section, we will describe the PoW consensus simulator that we have created as part of **Contribution 5** as seen in Figure 3.3. This simulator mimics the behavior of the blockchain and can be used to exponentially reduce the training time of the RL Agent. This enables

us to test multiple RL models and compare their performance to ensure system fairness and maintain a time constraint. The following subsection describes why we require the simulator followed by the logic implemented inside the simulator, and the last section describes the data collection process for the simulator.

3.5.1 Need for the PoW Consensus Simulator

The real-life setup deployed on the CCI xG testbed was used to train and test DQN models with different hyperparameters. Each run took seven days on average to reach convergence. We deployed three similar setups to run DQN models with different hyperparameters. After testing over three months, the best performance was achieved by optimizing the hyperparameters based on previous runs.

This approach is not scalable enough to measure the performances of multiple RL approaches and to optimize the RL agents. To reduce the experimentation time, we required a simulator miming the behavior of the PoW consensus mechanism. This simulator can-

- Reduce the experimentation time
- Make multiple parallel environments
- Optimize different RL models and compare their performance

3.5.2 Simulator Design

The PoW consensus simulator is implemented using Python and mimics the performance of the blockchain setup deployed in the CCI xG Testbed. The simulator decides the RL Agent as input. The decision consists of the list of competing miners and the PoW-D. The

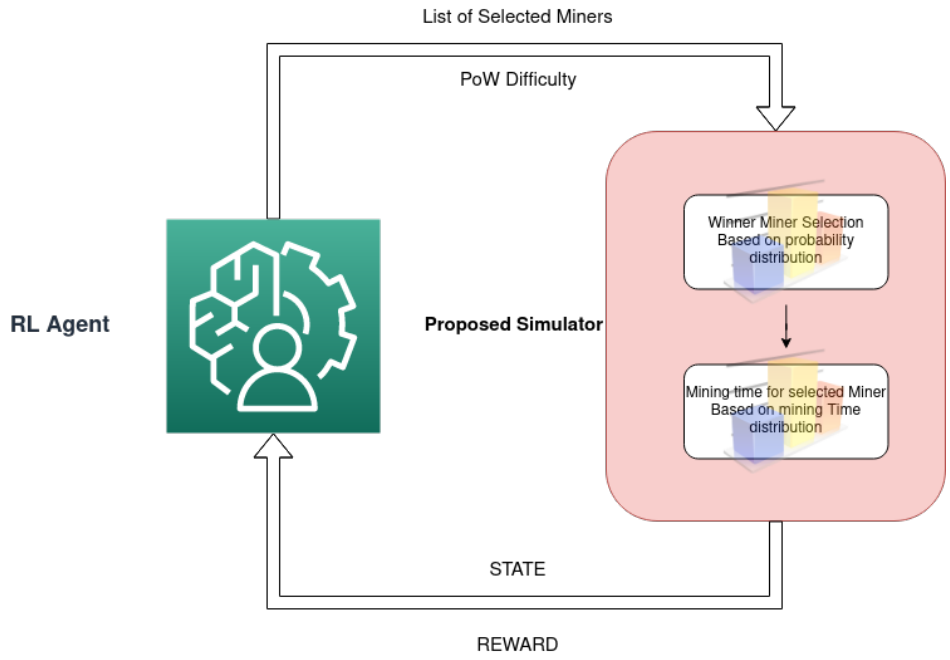


Figure 3.3: Simulator Design for PoW consensus mechanism.

output consists of the winning miner and the mining time. This can be seen in the loop in Figure 3.3.

The PoW consensus simulator uses a probability distribution as in Figure 3.4 to decide the winning miner. This probability distribution is proportional to the compute resources of the miner machines. Based on the winning miner and the PoW-D selected by the RL agent, the mining time is randomly chosen from a mining time distribution collected from the implemented setup described in the following subsection.

3.5.3 Data collection

To create the mining time distribution, we configured the blockchain setup described in Section 3.2 with different combinations of miners and difficulty values presented in Table 3.1. We collected 2000 data points for each combination of a miner and PoW-D and created a

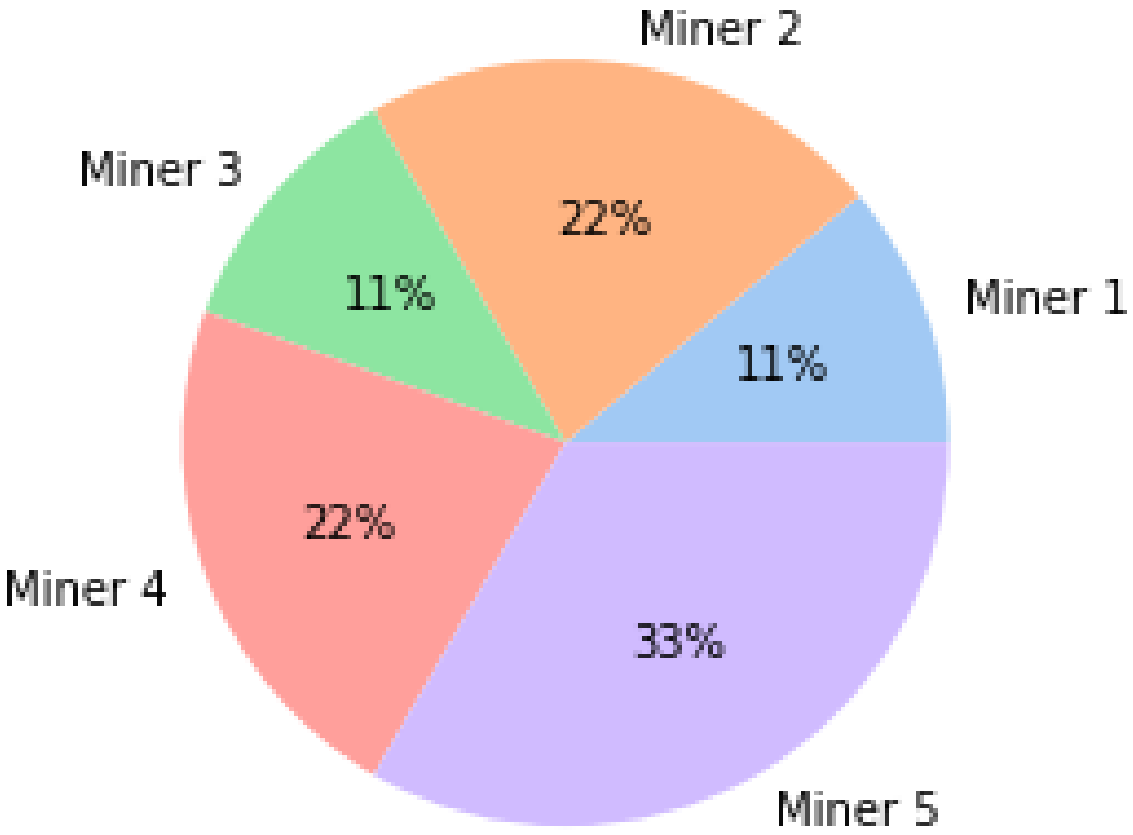


Figure 3.4: Probability Distribution: The miner's Probability of winning a round.

three-dimensional array. Each combination of a miner and difficulty value has a set of 2000 mining time values collected from the setup. A random time value can be selected from this distribution to capture the variations in the setup. This data is plotted in Figure 3.5.

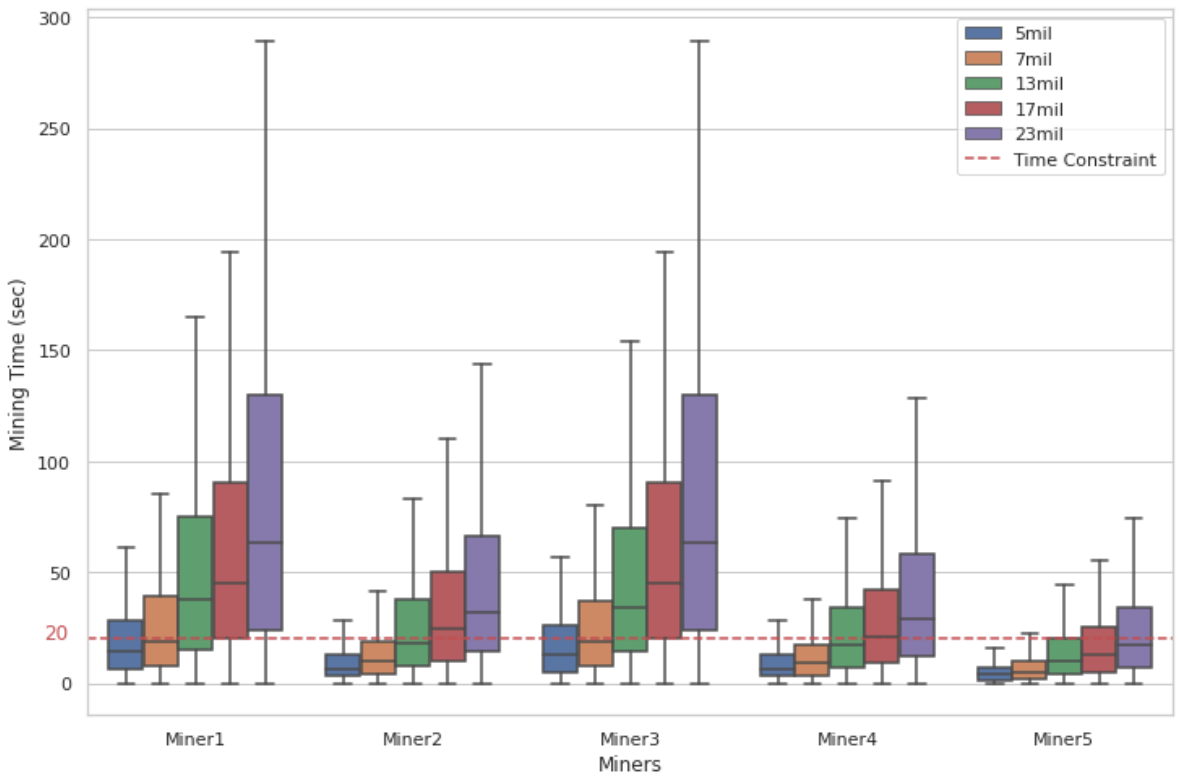


Figure 3.5: Mining time distribution for miners at different fixed PoW-D.

3.6 Summary

This Chapter detailed the proposed model and explained how the RL problem was formulated. It explains the implementation of the model and its integration with the CCI xG Testbed. We also described the design and implementation of the PoW consensus simulator. The next Chapter describes the experimental results and evaluation metrics.

Chapter 4

Results

This Chapter re-introduces the problem using measurements and graphs and evaluates the performance of the proposed RL agent-based PoW modification function. We begin by defining the metrics (i.e., block mining time, the average block mining time, and the winning percentage of each miner) used for evaluation. Then we measure the performance of the setup described in Chapter 3, using the default difficulty adjustment function from Ethereum 2.1. This forms the baseline for our measurements. Then we replace the built-in difficulty adjustment function with the proposed RL agent-based functions. The performance of the two proposed approaches is compared and evaluated RL-MS - Fairness at a known difficulty (miner selection) and RL-MDS - Fairness with variable difficulty (miner and difficulty selection) based on the metrics defined in Section 4.1. Then we compare different RL algorithms applied on the proposed simulator. Finally, Section 4.4 summarizes the two publications produced as part of this thesis and the specific contributions.

4.1 Metrics and Observations

Previous works [104] [88] have used block mining time as a metric to measure the performance of consensus mechanisms. To measure the performance, with the vanilla difficulty adjustment function and the proposed RL agent-based PoW modification function on the deployed setup, we consider three metrics:

1. Block mining time: is the duration it takes a miner to validate all the transactions in a block and solve the PoW puzzle.

2. Average block mining time: is the rolling average of the block mining times and is used for modifying the difficulty of PoW.

3. Fairness Metric - Winning percentage of each miner: To measure and ensure fairness in the blockchain, we introduce a fairness metric - the winning percentage of miners. It is defined as the percentage of mining rounds won by a given miner out of the total mining rounds.

4. Fairness Metric - Reward For Compute Utilized (RFCU): We introduce another metric to monitor the fairness in the system called RFCU. It is an adaptation of the Cost-Effectiveness Ratio (CER) [105] used in medical research and the Resource-Effectiveness Metric used in manufacturing processes [106]. Utilization is the total resources a machine dedicates to mining the blocks. It depends on the Availability of the device, in our case, the miner, and the mining capability or the compute dedicated to mining. For our setup, the VMs were used only for the blockchain, whereas in a real-life deployment, the machines can also be used for other applications. This utilization can be collected from the system statistics to see how many resources are dedicated to block mining.

$$U = Av \times C \tag{4.1}$$

Av is set to 100% when the RL agent selects the miner and 0% when the miner is not selected since it is not allowed to mine. We choose a value of 100% since, in our system, the VM is dedicated entirely to the blockchain. As suggested by [107], the distribution of CPU utilization has a mean of approximately 100% when there are no other resource bottlenecks. The Compute depends on the machines' resources, as

shown in Table 3.1. The reward that the miner receives is proportional to the winning percentage of the miner and is represented as Re. Re is directly proportional to the utilization of the miner, as can be seen in eq 4.2

$$\begin{aligned} U &\propto Re \\ Av \times C &\propto Re \\ Av \times \frac{C}{Re} &= K \\ Av \times \frac{C}{Re} &= RFCU \end{aligned} \tag{4.2}$$

For the system to be fair, each miner should be rewarded proportional to its compute and the resources it dedicated to the mining process. Therefore, this metric's value should be the comparable for all the miners irrespective of the percentage of rounds won and their mining power. The RFCU has is used to measure the fairness of the system and is a dimensionless quantity.

We collect the baseline measurements by running the Ethereum blockchain with three different configurations as seen in Figure 4.1:- PoW-D fixed to a very high value(i.e. 30 million), PoW-D fixed to a very low value (i.e. 1 million) and PoW-D modified by the vanilla PoW-D function. We will use the same colors for the three configurations as in Figure 4.1. Our approach uses equally spaced values between the maximum and the minimum difficulty values to discretize the action space. The agent decides a combination of the difficulty value and competing miners as described in Section 3.4.

As explained in Section 2.2, the probability of a miner winning a round is proportional to the computing power the device dedicates to the blockchain. The miner with a higher computing power will have a higher probability of winning. The computation powers of the five miners in our setup can be seen in Table 3.1. Miner 1 and Miner 3 have the same

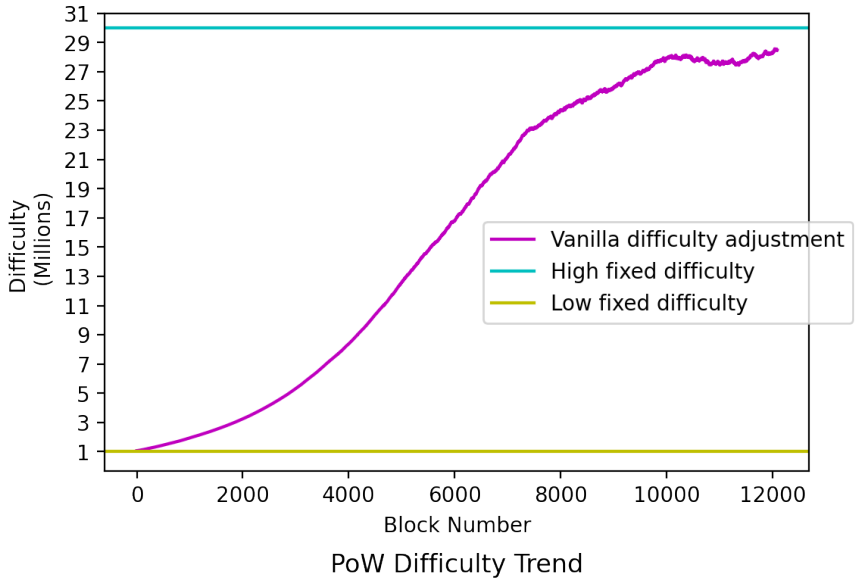


Figure 4.1: Variation of PoW difficulty values

computation power, Miner 2 and Miner 4 have the same computation power, whereas Miner 5 has significantly more computation power. These configurations of the machines help us observe the relationship between the compute and the winning percentage. Due to this resource difference, Miner 5 wins more rounds than the rest of the miners; this can be seen in Figure 4.2. It can be observed that the overall trend remains the same, irrespective of how the difficulty value is varied. This means the fairness in the system cannot be maintained by varying the value of difficulty of PoW

Figure 4.3 shows the mining times taken by the miners to solve the PoW puzzle and also shows the average time taken by the miners along with the time constraint used for our implementation (i.e., 20 seconds). As seen in Figure 4.3 (a), when the difficulty is fixed to a minimal value, the mining times are minimal (close to one second), and the average time is also meager. Most of these times are far below the time constraint. The mining and average mining times are significantly large if the difficulty is fixed to a significant value, as seen in Figure 4.3 (b). There is a substantial increase in mining times exceeding the time

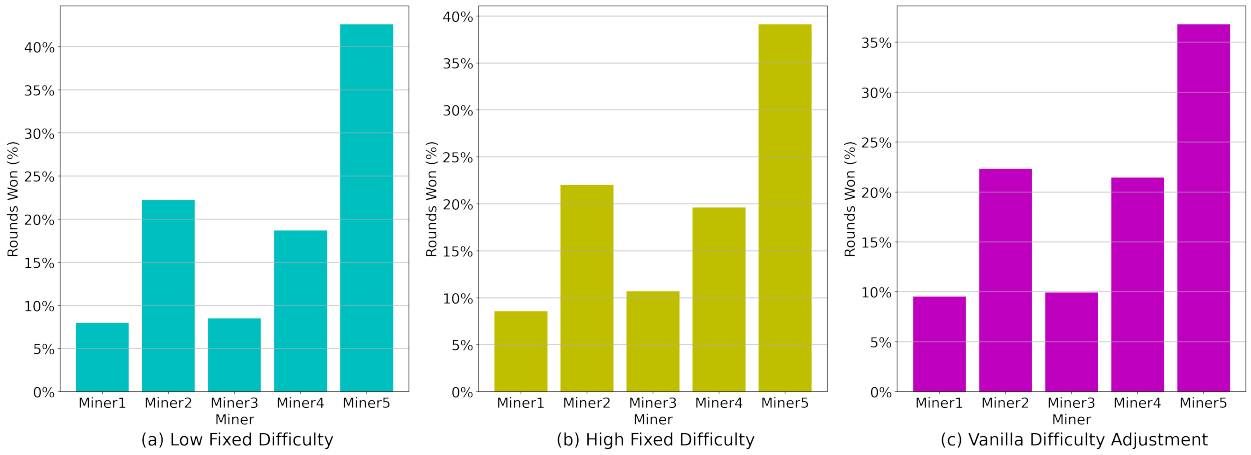


Figure 4.2: Winning percentage of miners with different configurations

constraint. In the case of the vanilla difficulty adjustment function, as seen in Figure 4.3 (c), the mining times increase from extremely low values to higher values that exceed the time constraint. The proportions of mining times exceeding the threshold corresponding to these configurations (i.e., PoW difficulty fixed to a very high value, PoW difficulty set to a very low value, and PoW difficulty modified by the vanilla PoW difficulty function) are summarised in Figure 4.4 (a), Figure 4.4 (b) and Figure 4.4 (c) respectively. It is clear that the mining times are considerable at higher difficulty, and at lower difficulty values, the times are minimal. This insight can help the RL agent pick the PoW-D value for selected miners to maintain the time constraint.

4.2 Performance Analysis

We introduced the performance metrics in the previous section and evaluated the setup with three base configurations. Figure 4.2 shows that the winning percentage of miners follows the same trend irrespective of the difficulty value. Hence a powerful miner can strongly influence the blockchain. The mining times are short at lower difficulties but keep increasing

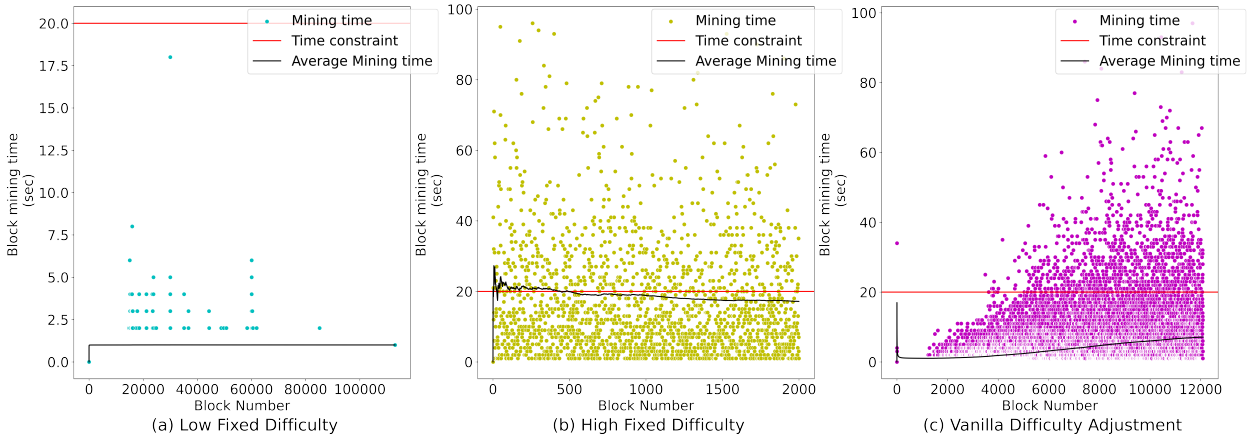


Figure 4.3: Variation in mining times

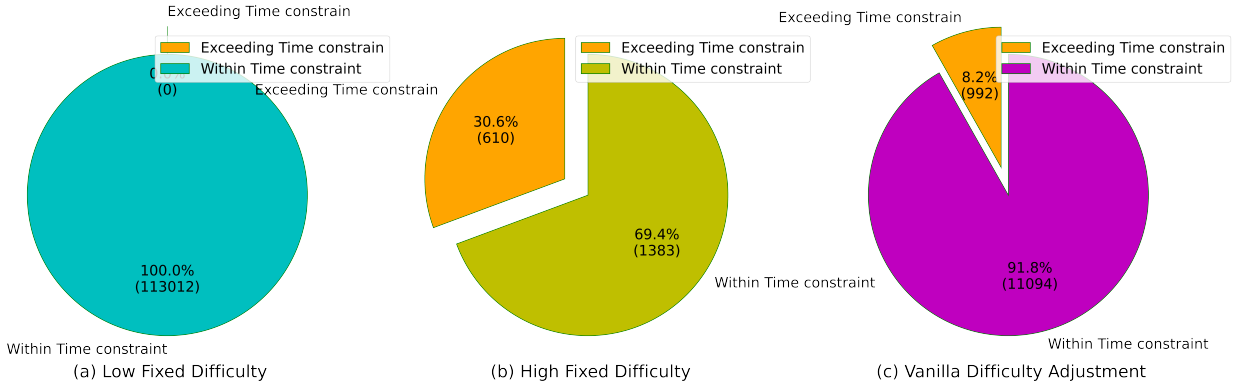
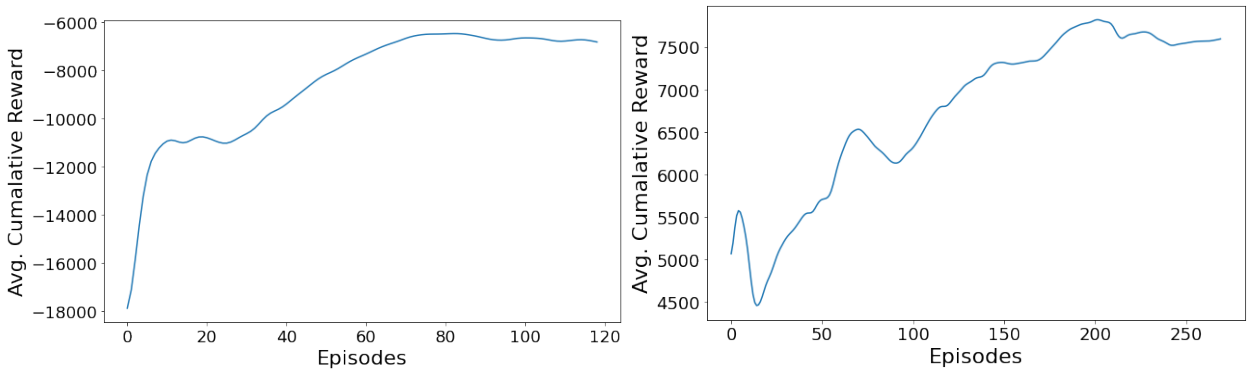


Figure 4.4: Mining time violating the time constraint



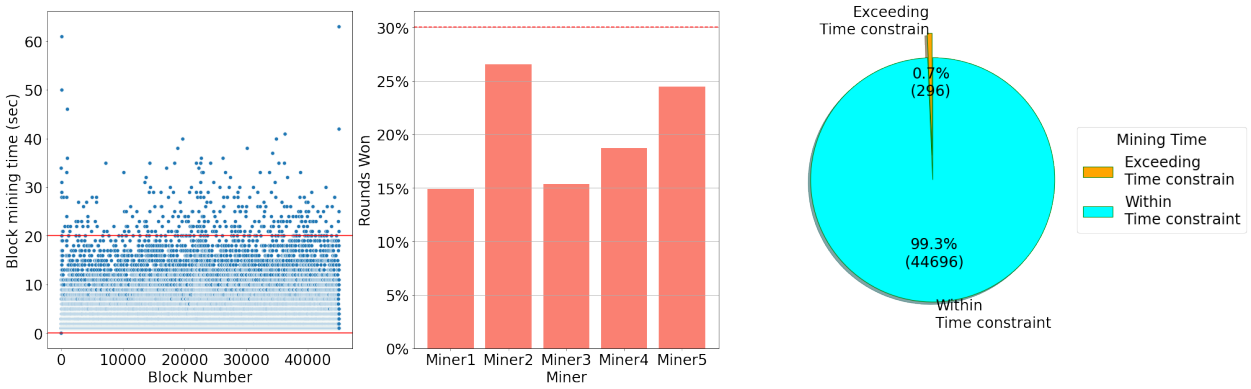
(a) RL Agent Reward for Known Difficulty (b) RL Agent Reward for variable Difficulty

Figure 4.5: Average Cumulative Reward

as difficulty increases (Figure 4.3). Moreover, the mining times are incredibly high at higher difficulty values (Figure 4.3 (b)), which can harm time-critical applications. Also, there is no way to ensure fairness in the system. In this section, as claimed in **Contribution 3**, we deploy and integrate the proposed RL empowered difficult adjustment function on the CCI xG testbed and evaluate its performance for two setups as explained in the following two subsections.

4.2.1 RL based Miner Selection (RL-MS)

First, we ensure fairness at a known difficulty Value:- To get this value, we find the maximum difficulty to meet the constraint of 20 seconds with only the weakest miners in the system (Miner1 and Miner 3) and, through experimentation, find it to be 5 million. The RL agent was trained using the setup described in Chapter 3, and the setup was run for over forty thousand blocks for three days. Our DQN implementation utilized a feed-forward Neural network with two hidden layers of 64 neurons each, connected sequentially. During training, we employed a buffer size of 10^5 and a batch size of 64 for efficient experience replay. A discount factor of 0.99 was used to balance immediate and future rewards. The learning



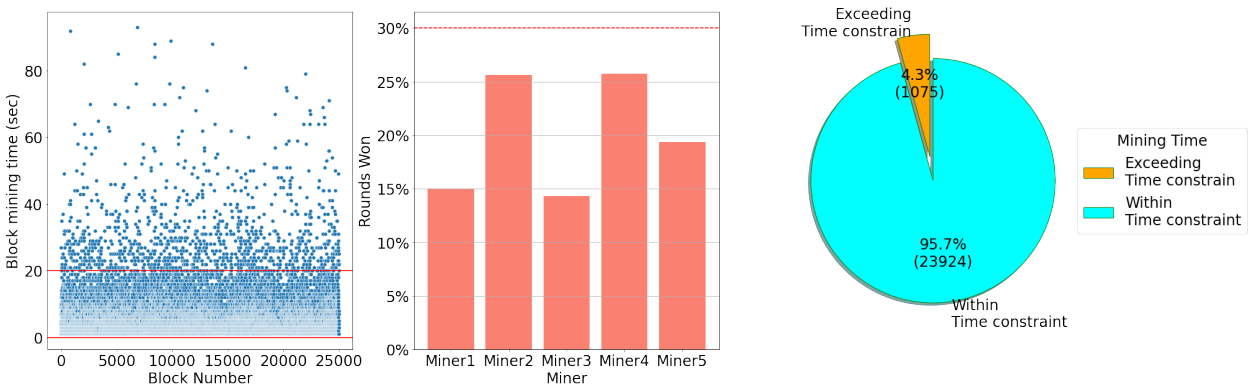
(a) Block Mining Time (b) Miner Winning Percentage (c) Mining Time Violation

Figure 4.6: Fairness at a known difficulty of PoW

rate was set to 0.005, and the network weights were updated every four steps. Each episode consisted of 50-time steps, and the exploration rate decayed from 1 by a rate of 0.995. The model performance can be seen in Figure 4.5a; the model initially saturates at about 20 episodes; this is a local maximum. Through exploration, the agent achieves better rewards and finally saturates after 80 episodes. Figure 4.6(b) depicts the winning percentage brought below the fairness factor ($\phi = 30\%$) for all the miners despite the heterogeneity in the system. This is because the RL Agent can choose the miners that compete at the fixed mining values. If a miner exceeds the fairness constraint, it stops that miner from mining and lets the other miners compete. Moreover, there is a drop in the proportion of mining values exceeding the time constraint Figure 4.6 (c) from 30.6% at high difficulties and 8.2% with the vanilla PoW difficulty function in Figure 4.4 (b),(c) to 0.7% in the proposed approach. This is in line with our claim in **Contribution 1**, the proposed RL agent successfully maintains fairness in the system.

4.2.2 RL based Miner and Difficulty Selection (RL-MDS)

We introduced a more complex set of states and actions to achieve **Contribution 2** and modified the RL agent accordingly. The RL-MDS DQN implementation utilized a feed-forward Neural network with three hidden layers of 64 neurons each, connected sequentially. During training, we employed a buffer size of 10^5 and a batch size of 64 for efficient experience replay. A discount factor of 0.99 was used to balance immediate and future rewards. The learning rate was set to 0.005, and the network weights were updated every four steps. Each episode consisted of 50-time steps, and the exploration rate decayed from 1 by a rate of 0.995. The RL agent was trained using the setup described in Section 3, and the setup was run for over a hundred thousand blocks for seven days. The winning percentage is brought below 30% for all the miners (Figure 4.7 (b)) despite the heterogeneity in the system. This is because the RL Agent can choose the miners that compete at different mining values. If a miner exceeds the fairness constraint, it stops that miner from mining and lets the other miners compete. There is a drop in the proportion of mining time values exceeding the time constraint (Figure 4.7 (c)) from 30.6% at high difficulties and 8.2% with the vanilla PoW difficulty function to 4.3% in the proposed approach. This is achieved by choosing the right difficulty value based on the miners competing to solve the PoW difficulty puzzle. The convergence plot can be seen in Figure 4.5. The model peaks at about 200 episodes and saturates at about 225 episodes. There is a small increase and decrease in the cumulative reward due to the dynamic nature of the blockchain. Sometimes the mining time of some blocks does not follow the general trend. The time is reduced if a miner solves a puzzle on its first attempt by chance. On the contrary, if the miner takes many tries to solve the hash puzzle, this time increases. The reward is based on the time; hence there is some variation.



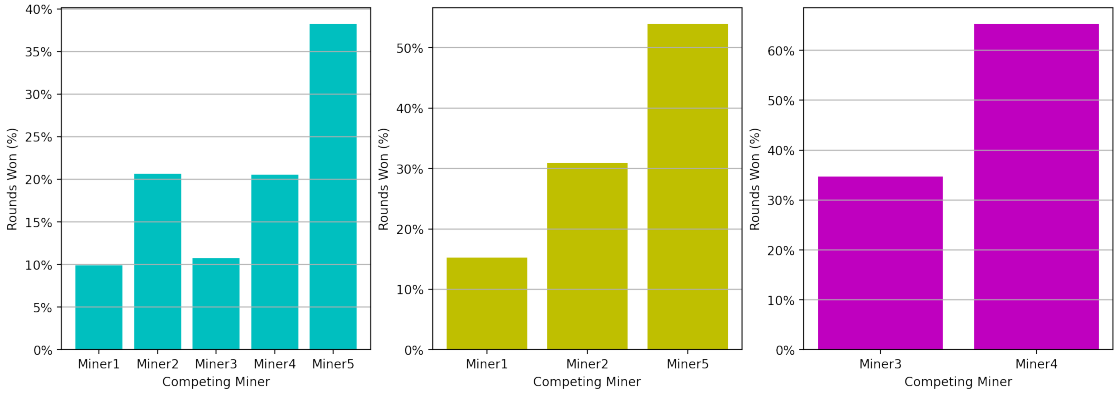
(a) Block Mining Time (b) Miner Winning Percentage (c) Mining Time Violation

Figure 4.7: Fairness at a variable difficulty of PoW

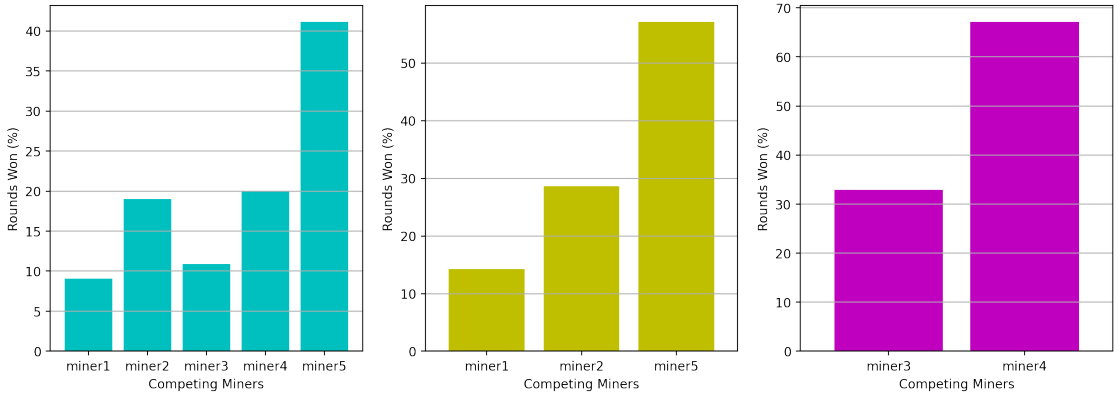
4.3 Simulator Results

We collected data for three setups to compare the simulator's performance to the testbed deployment as in Figure 4.8. The first setup consisted of all five miners. The second setup consisted of three miners having different resources, and the third setup consisted of the two least powerful devices. We collected data for 2000 mining rounds for each setup, as seen in Figure 4.8a. The simulator was run with the same configurations; the results can be seen in Figure 4.8b. The winning percentages of the miners are comparable in all three setups. This shows that the simulations are close to the performance of the testbed deployment. The simulator described in Section 3.5 was wrapped as aGymnasium (gym) environment. gym [1] is a standard API for RL and is a collection of diverse reference environments for training and evaluating RL models. OpenAI[2] develops gym, which is used extensively for comparing the performance of various RL algorithms. We used Stable-Baselines3 (sb3)[3] for training and testing powerful RL algorithms. sb3 is a collection of the PyTorch implementations of state-of-the-art RL algorithms. We implemented DQN[108], Proximal Policy Optimization (PPO) [109] and Advantage Actor Critic (A2C)[110] for comparison.

[1]https://gymnasium.farama.org/
[2]https://openai.com/
[3]https://stable-baselines3.readthedocs.io/en/master/



(a) Winning Percentage statistics for Testbed Deployment



(b) Winning Percentage statistics for Simulations

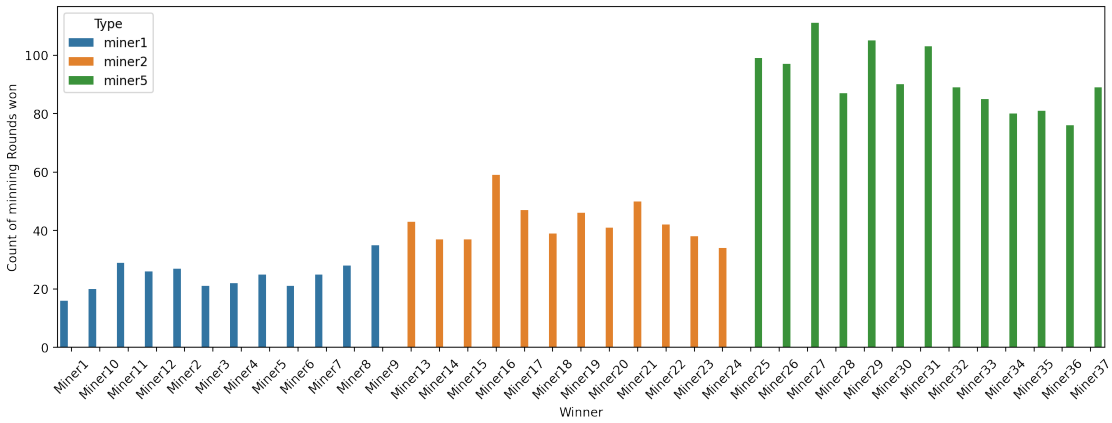
Figure 4.8: Winning percentage statistics for testbed deployment and simulations

4.3.1 Perforamance Analysis

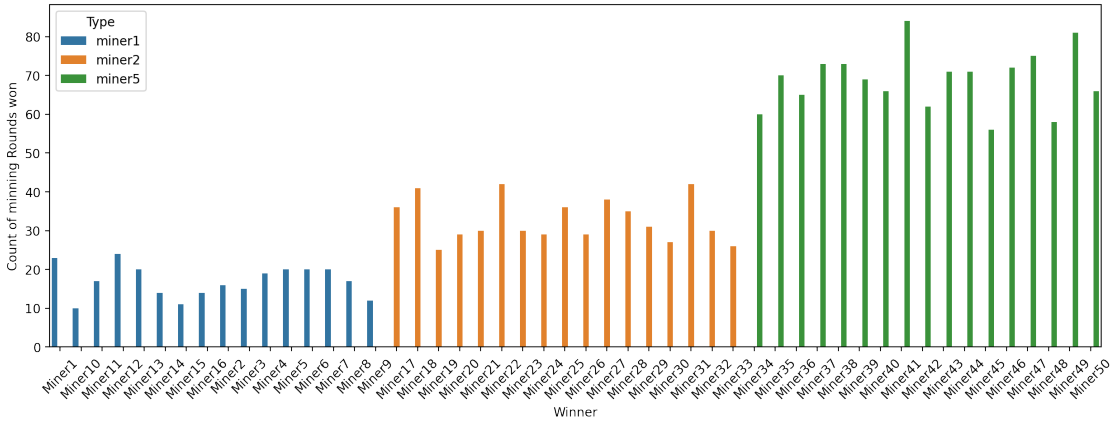
Section 3.5.1 discusses the shortcomings of the testbed deployment. Through the simulations, we were able to make the experimentation more scalable. The simulations were used to plot the performance of a similar setup consisting of 25 devices, 50 devices, and 50 devices, as can be seen in Figures 4.9a, 4.9b, and 4.9c, respectively. The simulations can mimic the performance of multiple instances of the nodes we have used in our testbed deployment. The groups denoted by Miner 1, Miner 2, and Miner 5 denote all the nodes with resources similar to Miner 1, Miner 2, and Miner 5 from Table 3.1.

We used the default configurations for the three algorithms in the sb3 implementations. *The DQN Agent* uses a Learning Rate of 0.0001, a buffer size 0f 1000000, and a batch size of 32. We used a soft upgrade coefficient of 1 and a discount factor of 1. The exploration fraction was set at 0.1 with an initial exploration rate of 1 and a final exploration rate of 0.05. *The A2C Agent* used a Learning Rate of 0.0007, batch size of 5, a gamma of 0.99, bias-variance trade-off factor of 1.0 and an Entropy coefficient of 0. *The PPO Agent* used Learning Rate of 0.0003, a batch size of 64, a discount factor of 0.99, and a clip range of 0.2.

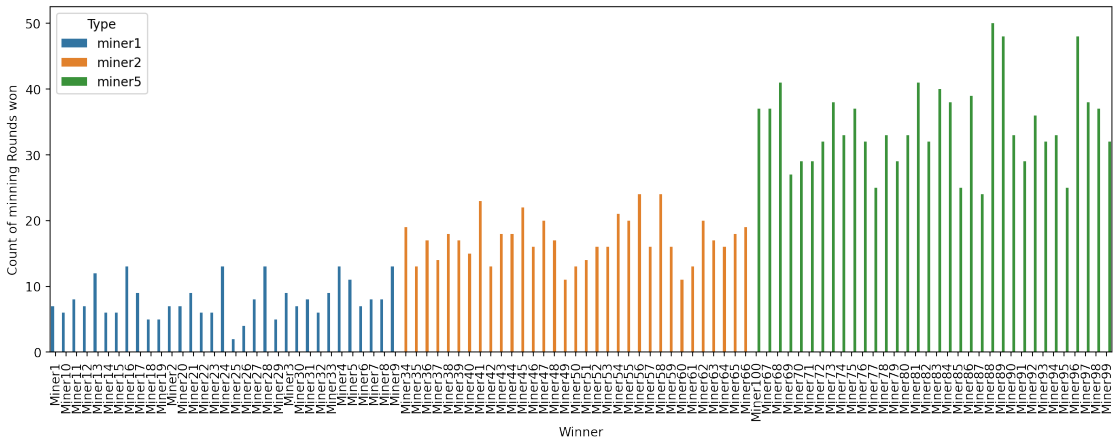
The three models attain comparable performance in terms of fairness as in Figure 4.10. The winning percentage was brought below the fairness factor of 30%, depicted by the red-dotted line. Figure 4.12a, Figure 4.12b, Figure 4.12c show the winning percentage statistics for the five miners in the system while testing for DQN, A2C and PPO respectively. Most of the values are below the fairness factor, proving our claim of maintaining fairness in the system. The three algorithms had comparable results in reducing the mining time as well. The mean is below the 20 second mark as in Figure 4.11. This performance can be further improved by optimizing the hyperparameters of the models under consideration. The mean cumulative training reward and the mean cumulative evaluation reward are summarized in



(a) Simulations for 25 Nodes



(b) Simulations for 50 Nodes



(c) Simulations for 100 Nodes

Figure 4.9: Scalability of the Simulator

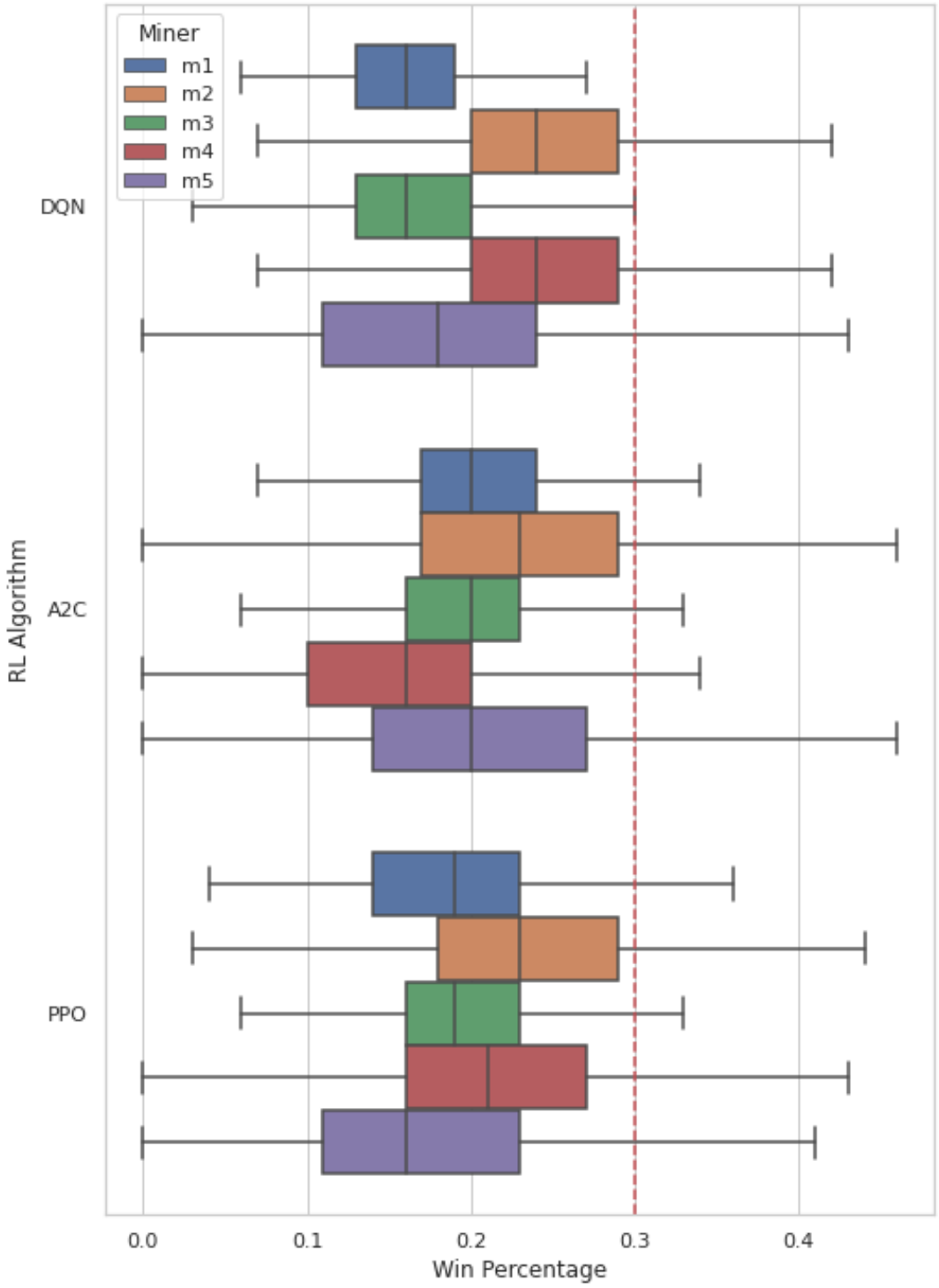


Figure 4.10: Comparison of RL Algorithms - Winning Percentage

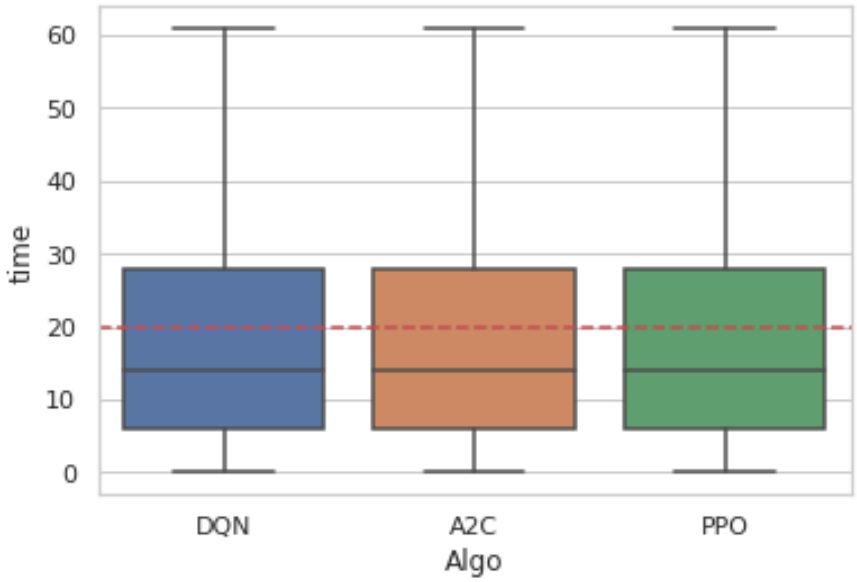
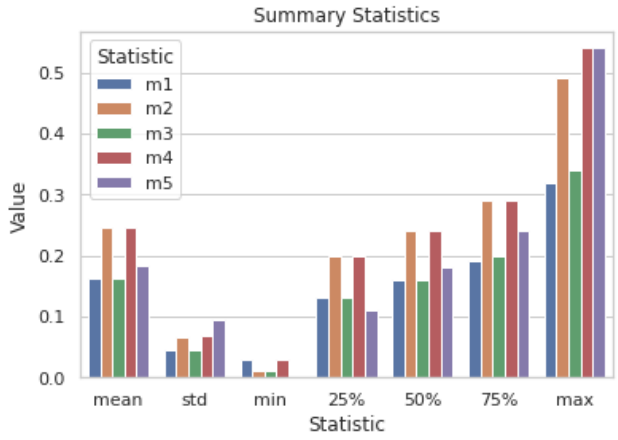
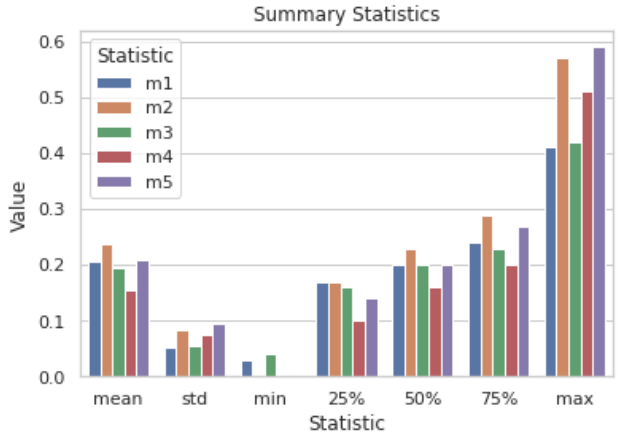


Figure 4.11: Comparison of RL Algorithms - Minning Time

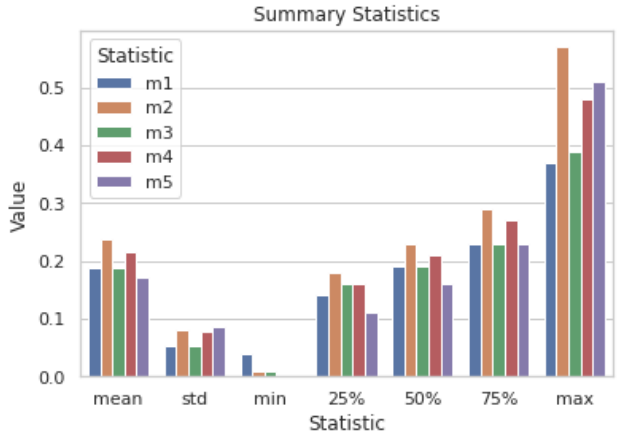
Figure 4.13 and Figure 4.14. While the three models maintained fairness in the system, there was a difference in the reward earned by the agent. It can be seen that the mean cumulative reward is more in the case of A2C as compared to DQN and PPO. DQN shows the least mean cumulative reward as compared to PPO and A2C. This difference is due to the three approaches' different learning methods. DQN, shows the worst performance which is because, by default, the agent is only allowed to explore 10% of the total steps; this can be seen in Figure 4.15. Due to the limited exploration, the agent cannot learn the behavior of the blockchain simulator. In the case of our deployment of DQN on the actual setup, we configured the agent to explore the environment throughout the deployment, where the exploration rate decayed from a value of 1 by a factor of 0.995 till the end of the training. Hence we achieved better results than the default configurations of stable baselines3. DQN is also susceptible to overestimating the Q-values [111] since the blockchain environment is dynamic, the learned Q-values might change over time, resulting in lower rewards. PPO implementation in sb3 does not have an explicit exploration parameter like



(a) Winning Percentage statistics for DQN agent



(b) Winning Percentage statistics for A2C agent



(c) Winning Percentage statistics for PPO agent

Figure 4.12: Winning percentage statistics for RL models deployed on simulator

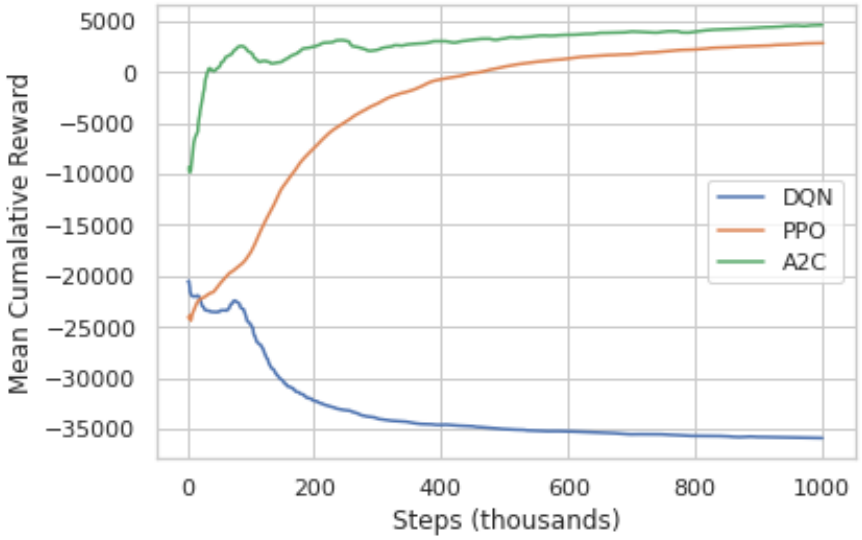


Figure 4.13: Mean Cumulative Training Reward

in DQN it incorporates exploration through the inherent stochasticity of the policy network. In contrast, the A2C agent uses agent primarily focuses on the actor-critic framework for policy optimization and value function estimation. However, the model performance can be improved by optimizing the hyperparameters. This can be a part of future work.

We also used the simulations to run the optimized DQN model for 5000 mining rounds and calculated the average value of RFCU for each miner. The experiment was repeated 100 times to observe the behavior of the fairness metric. The average value of RFCU is comparable for all the miners as in figure 4.16. The maximum value is 1653.79, and the minimum value is 1535.53. The max value of RFCU is $\approx 2\%$ above the mean RFCU value, and the minimum RFCU value is $\approx 5\%$ below the average RFCU value. The error bars are also significantly small for all the miners. This shows that all the miners are rewarded proportional to the resources they dedicate to the mining process.

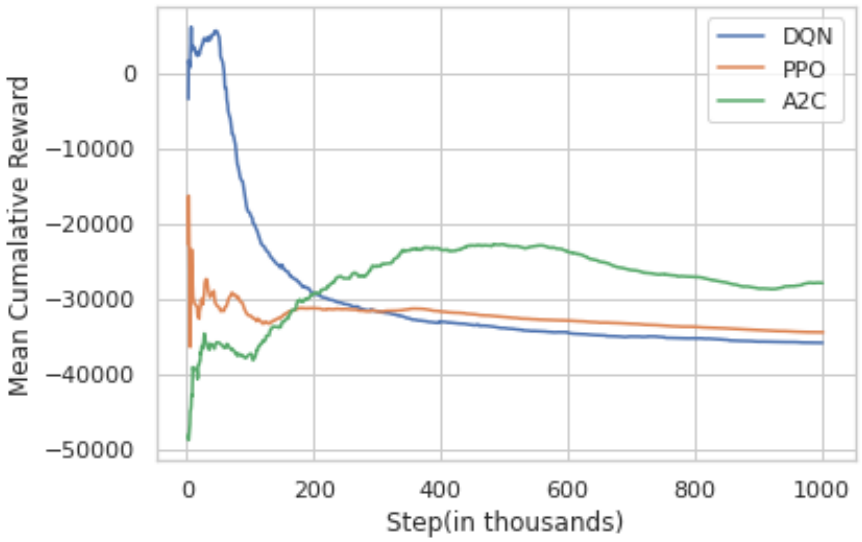


Figure 4.14: Mean Cumulative Evaluation Reward

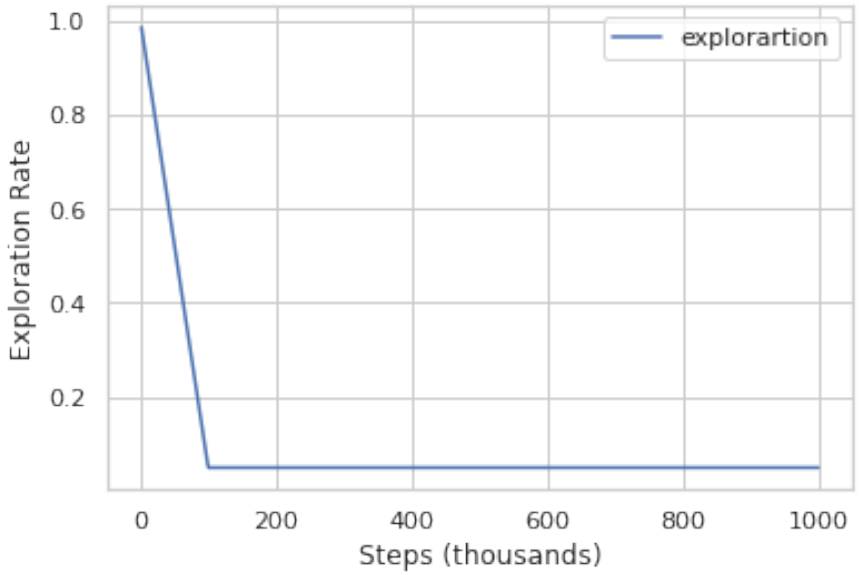


Figure 4.15: Default exploration rate for DQN

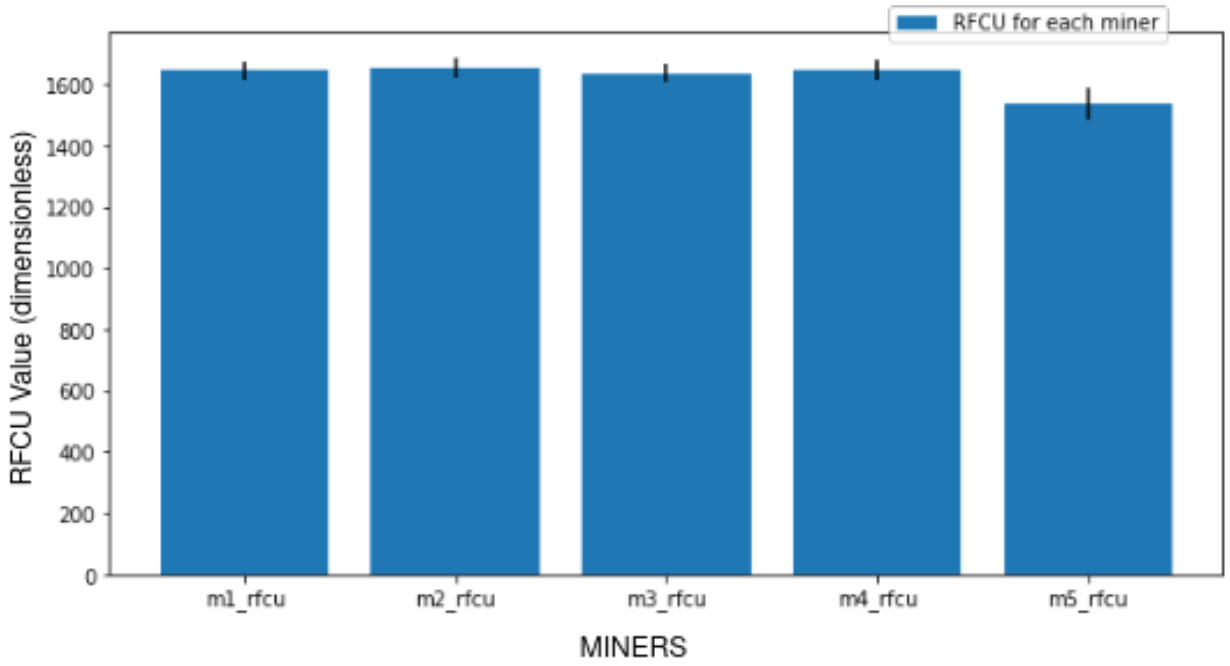


Figure 4.16: Fairness metric - RFCU for miners over 100 experiments

4.4 Other Contributions and Results

This Section will touch upon the additional contributions of this work, which include; **Contribution 6:** A published demo paper, titled *"AI/ML Data-driven Control Loop for Managing O-RAN SDR-based RANs"*, accepted as an IEEE INFOCOM 2023 demonstration paper[4]. **Contribution 7:**A journal paper yet to be submitted, titled *"Closing the Loop for End-to-end O-RAN Experimentation: Holistic Integration of Near- and Non-Real Time RICs"*.

The O-RAN Alliance has brought about a paradigm shift in the mobile network industry by introducing the concept of disaggregation, softwarization, and openness in RAN architecture. The O-RAN ecosystem is designed to support the deployment of 5G networks and beyond by providing a common control and management overlay that enables Mobile Network Operator

[4]https://infocom2023.ieee-infocom.org/

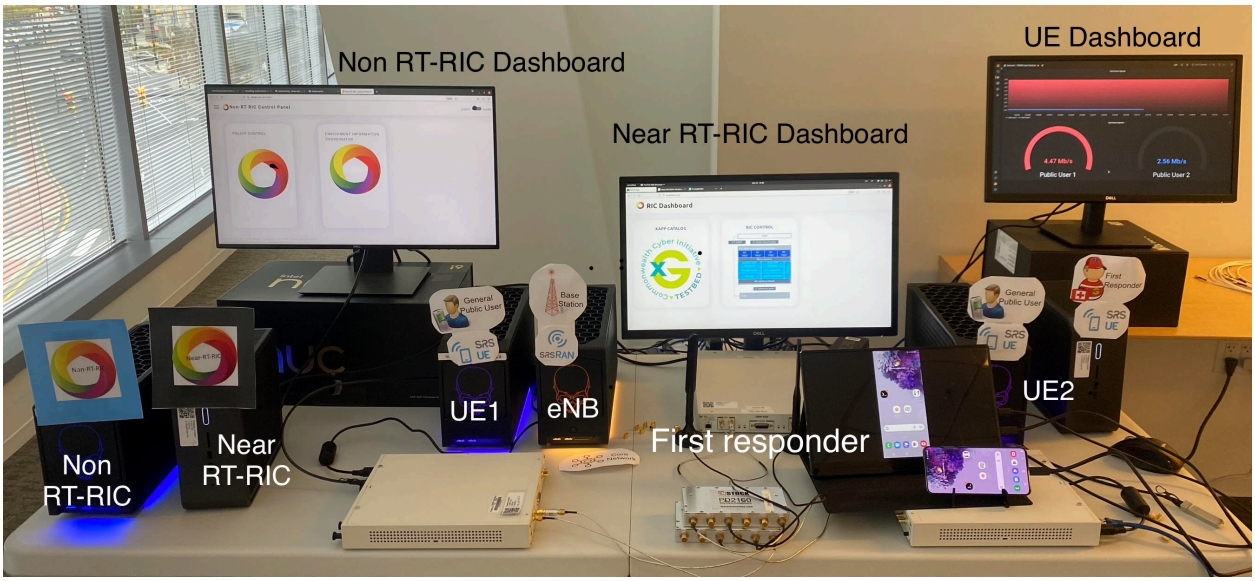


Figure 4.17: End-to-end O-RAN SDR-based network experimental setup.

(MNO)s to orchestrate their entire RAN using custom control logic via standardized third-party applications. The O-RAN ecosystem's common control and management overlay is divided into two RICs hosting xApps and rApps as cloud-based microservices. The near-real-time RIC hosts xApps that perform closed-loop control over RAN components in the order of 10-1000ms, such as load balancing and scheduling. On the other hand, the non-real-time RIC hosts rApps that have a lasting effect based on historical data trends, taking longer than 1000ms, such as data analytics and ML model training to optimize RAN parameters. While the standardization and prototyping of xApps and their interfaces are mature and accompanied by open-source reference implementations, the standardization of rApps, their interfaces, and their generation of policies to interact with xApps lag significantly behind. This lack of well-defined interfaces and open-source reference implementations of rApps and their connection with xApps limits experimental verification of ML data models to manage the RAN, evaluation of the impacts of AI-driven decisions to change the behavior of existing xApps, and development of end-to-end control loops comprised of xApps and rApps to manage the RAN autonomously.

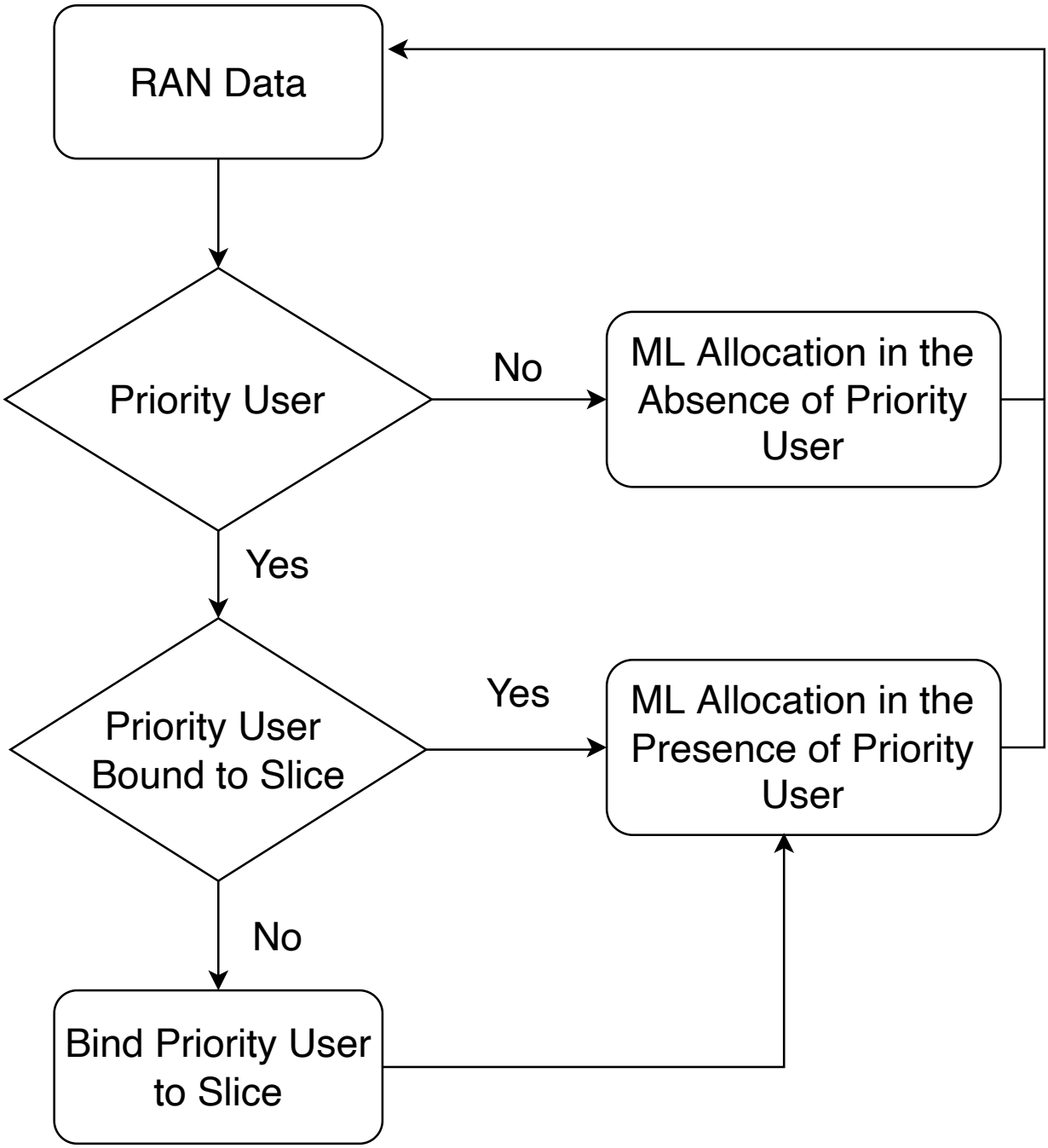


Figure 4.18: Policy workflow.

To address this issue, we presented a demo *"AI/ML Data-driven Control Loop for Managing O-RAN SDR-based RANs"* that introduces the first proof-of-concept rApp capable of generating policies and controlling the behavior of xApps. This demo integrates the rApp with the NexRAN RAN slicing xApp, creating the first experimental ML-based RAN slicing platform based on O-RAN. The setup can be seen in Figure 4.17. This work is further expanded in our journal submission titled *"Closing the Loop for End-to-end O-RAN Experimentation: Holistic Integration of Near- and Non-Real Time RICs"*. We discuss the implementation of the end-to-end, full O-RAN-compliant network deployment and experiments, providing an overview of the O-RAN architecture and its functional components. Then we propose a minimum viable architecture for O-RAN, detailing the design choices to realize this architecture, describing the realization of the end-to-end O-RAN setup, and evaluating its ability to manage an open-source, SDR-based RAN. Specifically, we demonstrate its ability to perform near- and non-real-time control loops to autonomously manage a dynamic RAN slicing use case (Figure 4.18). This involves automatically reacting to changes in traffic demand, predicting a new optimal radio resource allocation, and slicing the RAN accordingly. We also assess the delay associated with operating the near- and non-real-time control loops due to the interactions between the different components of our end-to-end O-RAN setup. Overall, our evaluation highlights the effectiveness of our proposed architecture and the potential of O-RAN technology in enabling more efficient and flexible management of wireless networks. Finally, we discuss the remaining open challenges in the experimental research on O-RAN networks based on developing our end-to-end O-RAN setup and provide recommendations and suggestions. As part of my Master's, I contributed to the publications in the following ways

- *ML model for RAN slicing:* Designed a pre-trained regression model capable of deciding the proportions of predicting the proportions of resources to be allocated to the users

depending on the demand. It intelligently prioritized the First Responder users and maintained minimum operability for the rest of the users.

- *rApp:* The pre-trained model was containerized and launched as a microservice integrated into the Non-RT-RIC. It collected the user demand data and User Equipment (UE) information from the mock O1 interface, and its intelligent decisions were passed to the A1 terminator in the near RT=RIC.

- *Mock O1 Interface:* The mock O1 interface was setup to transfer the information from the RAN to the rApp in the non RT=RIC. We used API calls to share this information, the server was hosted in the rApp, and the data was sent from the base station.

- *xApp helper script:* The communication between the entities in the near RT=RIC are carried out using RIC Message Router (RMR) [112] messaging. The instructions from the rApp were converted into a form understood by the nexRAN xApp, which modified the RAN as per the information from the rApp.

- *System performance evaluation:* Carefully designed a setup to isolate the different components of the End-to-end O-RAN implementation to ensure the following timings:

  - rApp and model performance time
  - O1 performance
  - xApp helper script performance

4.5 Summary

This Chapter provides the performance evaluation of different RL models on the CCI testbed and a novel simulator measured using three metrics - block mining time, average block mining

time, and fairness. We analyzed the performance of the vanilla difficulty adjustment function in Ethereum using our setup deployed in the CCI xG testbed. We use a novel approach that uses an RL agent to modify the difficulty values and select miners to maintain fairness in the system. The percentage of block mining time violating the constrain significantly reduced from 30.6% at high difficulties and 8.2% with the vanilla PoW difficulty function by up to 97% and up to 85% in RL-MS and RL-MDS respectively. We then analyzed and compared the performance of three RL approaches (DQN, A2C and PPO) on the proposed simulator. The three approaches showcased comparable results, achieved fairness, and reduced mining times. But the performance of A2C algorithm is slightly better than the rest. Finally, the Chapter provides information on the two publications produced as part of this thesis and a brief overview of the work and the specific contributions.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis has worked towards alleviating the effects of heterogeneity of miners and aims to maintain fairness in a private blockchain that uses the PoW consensus algorithm. It also reduces and constrains the mining time to meet the requirements of time-critical applications. We deploy a private Ethereum blockchain using geth and OpenStack on the CCI xG testbed. The implementation comprises five miners, five clients, and one machine running the RL agent. The miners and the clients form a P2P network, and the communication of the blockchain with the RL agent is handled by a Kafka bus. We also deployed an FL algorithm on the blockchain to train a classifier to detect the handwritten digits from the MNIST dataset. The data is distributed in smaller chunks and saved on all the machines. This data is used to train local models, which are used to make the global model.

To measure the performance of the blockchain, in Chapter 4, we introduce three metrics - mining time, average mining time, and winning percentage. The complete setup is described in Chapter 3 and uses three basic configurations to measure the baseline performance. These include very high PoW difficulty (i.e., 30 million), minimal PoW difficulty (i.e., 1 million), and the built-in difficulty adjustment function. It is observed that irrespective of the difficulty values, the trend of the winning percentages of the miners is the same. The winning percentage is proportional to the miner's resources, which shows that the system is unfair

and is susceptible to the 51% attack. It can also be seen that the block mining time increases with increased difficulty; this can be problematic for time-critical applications.

We propose two novel RL empowered difficulty adjustment functions (Chapter 3) to ensure fairness and to reduce block mining times. The proposed approach lowered the miners' winning performance below the fairness factor ($\phi = 30\%$) as seen in Figure 4.6 (b) and Figure 4.7 (b). The block mining times are also reduced below the time constraint ($t = 20seconds$). The percentage of block mining time violating the constrain significantly reduced from 30.6% at high difficulties and 8.2% with the vanilla PoW difficulty function by up to 97% and up to 85% in RL-MS and RL-MDS respectively. The performance of three RL Algorithms was measured by deploying them on the novel blockchain simulator. The three algorithms reduced the winning percentage and maintained time constraints. But the performance of the A2C algorithm was found to be better than DQN and PPO. However, these models can be improved by optimizing their parameters and cleaning the data used for the simulator. The analysis of the RFCU values for all miners indicates that their average value is comparable, with a narrow range between the maximum and minimum values. The small error bars further emphasize the consistency of the rewards, indicating that the miners are being compensated proportionally based on their resource contributions to the mining process. This outcome affirms the fair and equitable distribution of rewards among the miners. The work also discussed the additional contributions in the form of a conference demo paper that present the first end-to-end ML-based RAN slicing platform based on O-RAN and further delves deeply into measuring the performance of the setup in a journal paper which provides insights, suggestions, and lessons learned.

5.2 Future Work

The performance benefits of the proposed approach can be utilized for a broad range of applications such as finance, the medical industry, and Industrial IoT. This work suggests a way to improve the underlying blockchain's performance, which enhances the applications' performance. The approach can be further optimized by tuning the hyperparameters in the training process of the RL agent and by using techniques such as PPO [109] and A2C [110] to make the RL agent learn and perform better. The system can also be tailored to the requirements of different applications by changing the time constraints and the fairness factor. The setup was deployed using OpenStack in VMs on the CCI cG testbed. These VMs can be migrated to low-power devices such as raspberry Pi[1] and other devices that can run a Linux operating system.

We now discuss two novel architectures for implementing blockchain-enabled FL in smart cities. The first architecture consists of multiple devices with compute and memory comparable to that of MEC servers Figure 5.1. These can be autonomous vehicles, Augmented Reality (AR) and Virtual Reality (VR) devices. These devices have times when they are available and times when they are switched off. During their downtime, they can participate in the blockchain, and their resources can be used for applications such as model training in FL.

The second architecture consists of multiple edge servers with compute and memory comparable to the MEC servers Figure 5.2. The edge servers are dedicated to one of the services in a smart city, such as smart health and smart transportation, or are shared by multiple services. These services have sub-services that utilize the resources at the edge servers. The sub-services have different trends and patterns of operations. Each service has a time when

[1]https://www.raspberrypi.com/

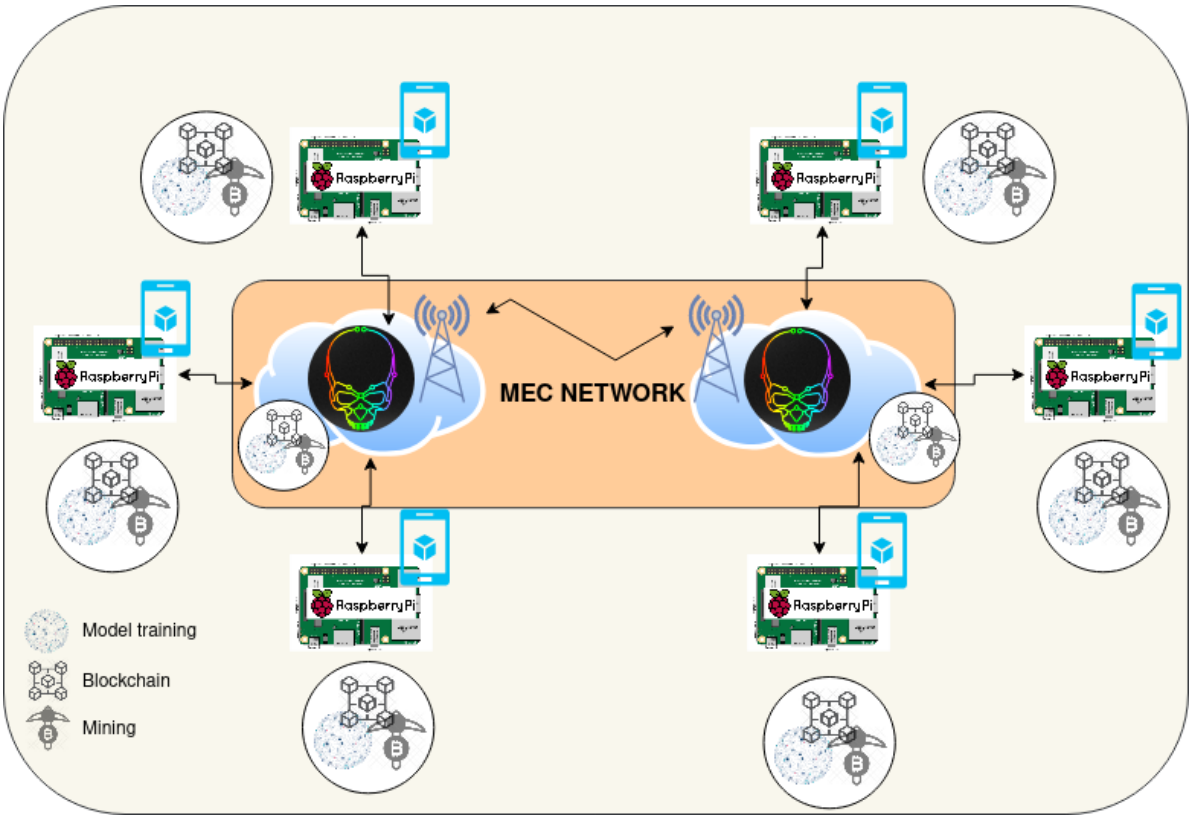


Figure 5.1: UE based Architecture for Blockchain-enabled FL

its utilization of resources is high, and this is the high time. Sometimes, these sub-services are low or moderate, known as low time. These servers can train FL models during their low time to use resources efficiently.

We use a Poisson distribution [113] to model the utilization of these services in both architectures and create a synthetic dataset[2]. A high mean value is chosen for the high time for both architectures. A low mean value is selected for the low time in the second architecture, whereas the low value is set to 0 in the case of the first architecture. This is done because the devices in the first architecture are switched off during downtime. The code for generating the synthetic dataset and the dataset is uploaded to the git repository. As a future work, this data can be integrated with the RL agent to improve its performance based on the

[2]https://github.com/Pseth3/AdaptivePoW

availability trends of these devices.

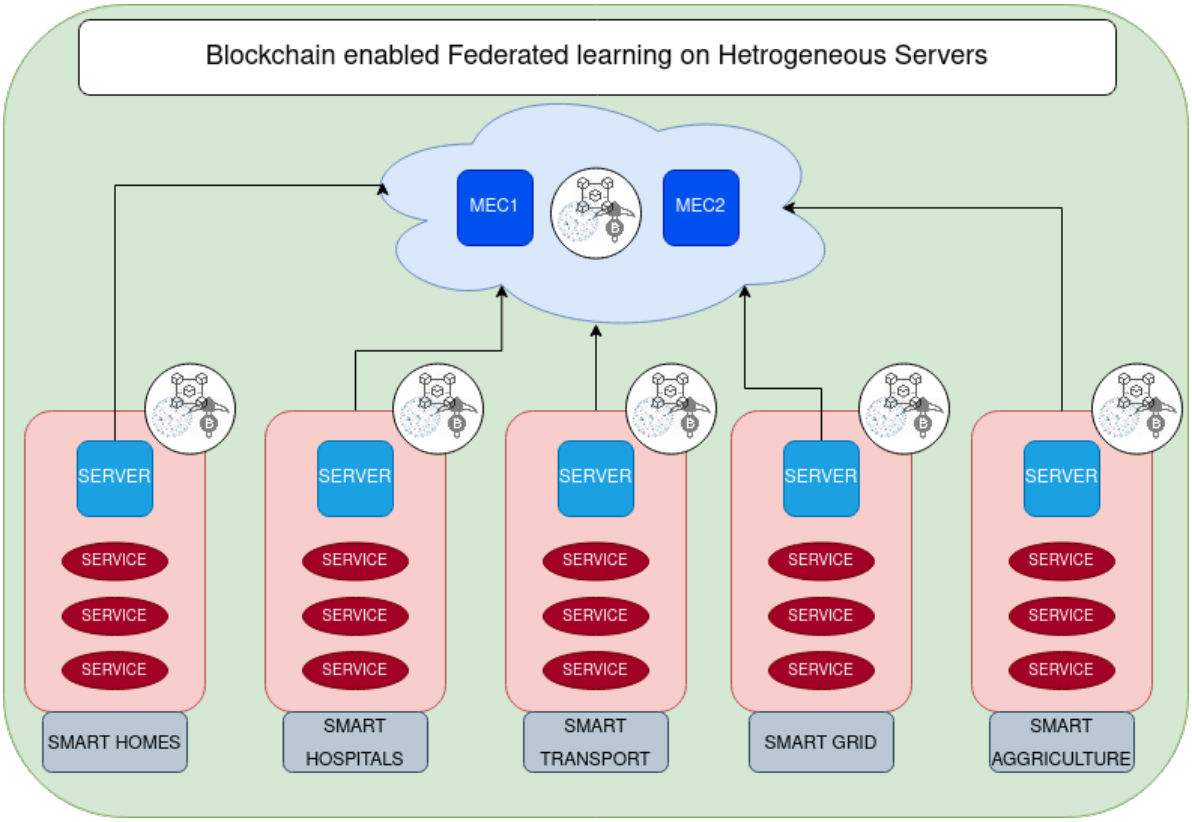


Figure 5.2: Server based Architecture for Blockchain-enabled FL

5.3 Summary

This thesis has successfully delivered the proposed contributions - we proposed, implemented and analyzed two novel RL based algorithms to ensure fairness amongst miners in a Private blockchain. The work further makes the use of blockchain in time-critical applications viable by maintaining application-specific time constraints. Finally, it provides future directions and ways to improve and implement the proposed solutions.

Bibliography

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, Dec 2008. URL https://bitcoin.org/bitcoin.pdf. Accessed: 11. Feb. 2023.

[2] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[3] Fredy Andres Aponte-Novoa, Ana Lucila Sandoval Orozco, Ricardo Villanueva-Polanco, and Pedro Wightman. The 51% attack on blockchains: A mining behavior study. *IEEE Access*, 9:140549–140564, 2021.

[4] Basu Dev Shivahare, Shashikant Suman, Sai Sri Nandan Challapalli, Prakarsh Kaushik, Amar Deep Gupta, and Vimal Bibhu. Survey paper: Comparative study of machine learning techniques and its recent applications. In *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, volume 2, pages 449–454. IEEE, 2022.

[5] Farhad Ahamed and Farnaz Farid. Applying internet of things and machine-learning for personalized healthcare: Issues and challenges. In *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*, pages 19–21. IEEE, 2018.

[6] Pradeep Kumar Kushwaha and M Kumaresan. Machine learning algorithm in healthcare system: A review. In *2021 International Conference on Technological Advancements and Innovations (ICTAI)*, pages 478–481. IEEE, 2021.

[7] Aashish Rai, Clara Ducher, and Jeremy R Cooperstock. Improved attribute manipulation in the latent space of stylegan for semantic face editing. In *2021 20th IEEE*

International Conference on Machine Learning and Applications (ICMLA), pages 38–43. IEEE, 2021.

[8] Qian Wang and Toby P Breckon. Contraband materials detection within volumetric 3d computed tomography baggage security screening imagery. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 75–82. IEEE, 2021.

[9] René García Franceschini, Jeffrey Liu, and Saurabh Amin. Damage estimation and localization from sparse aerial imagery. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 128–134. IEEE, 2021.

[10] Ricardo Flores, ML Tlachac, Ermal Toto, and Elke A Rundensteiner. Depression screening using deep learning on follow-up questions in clinical interviews. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 595–600. IEEE, 2021.

[11] Weilin Sun, Vincent Lu, Aaron Truong, Hermione Bossolina, and Yuan Lu. Purrai: A deep neural network based approach to interpret domestic cat language. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 622–627. IEEE, 2021.

[12] Voula C Georgopoulos. Advanced time-frequency analysis and machine learning for pathological voice detection. In *2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pages 1–5. IEEE, 2020.

[13] Akash Devgun. A machine learning adaptive approach to remove impurities over bigdata. In *2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE)*, pages 220–225. IEEE, 2014.

[14] Christoph Augenstein, Norman Spangenberg, and Bogdan Franczyk. Applying machine learning to big data streams: An overview of challenges. In *2017 IEEE 4th international conference on soft computing & machine intelligence (ISCMI)*, pages 25–29. IEEE, 2017.

[15] Shreya Vasoya, Nishi Patel, Dipak Ramoliya, and Khushi Patel. Potentials of machine learning for data analysis in iot: A detailed survey. In *2020 3rd International Conference on Intelligent Sustainable Systems (ICISS)*, pages 291–296. IEEE, 2020.

[16] Brendan McMahan and Research Scientists Daniel Ramage. Federated Learning: Collaborative Machine Learning without Centralized Training Data, January 2023. URL https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. [Accessed: 23. Jan. 2023].

[17] Qian Wang and Haofei Meng. Blockchain-based federated learning with limited resources. In *2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)*, pages 449–452. IEEE, 2022.

[18] Wael Issa, Nour Moustafa, Benjamin Turnbull, Nasrin Sohrabi, and Zahir Tari. Blockchain-based federated learning for securing internet of things: A comprehensive survey. *ACM Comput. Surv.*, 55(9), jan 2023. ISSN 0360-0300. doi: 10.1145/3560816. URL https://doi.org/10.1145/3560816.

[19] Commonwealth Cyber Initiative, March 2023. URL https://cyberinitiative.org. [Accessed: 10. Mar. 2023].

[20] CCI xG Testbed, March 2023. URL https://cyberinitiative.org/xg-testbed.html. [Accessed: 10. Mar. 2023].

[21] 3GPP TS 28.532, 2022. URL https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3427. "Accessed: 11/02/22".

[22] Subrato Bharati, M. Rubaiyat Hossain Mondal, Prajoy Podder, and V. B. Surya Prasath. Federated learning: Applications, challenges and future directions. *Int. J. Hybrid Intell. Syst.*, 18(1-2):19–35, January 2022. ISSN 1448-5869. doi: 10.3233/HIS-220006.

[23] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y. Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv*, February 2016. doi: 10.48550/arXiv.1602.05629.

[24] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy preserving machine learning. Cryptology ePrint Archive, Paper 2017/281, 2017. URL https://eprint.iacr.org/2017/281. https://eprint.iacr.org/2017/281.

[25] T. Li, Sahu A. K., A. Talwalkar, and V. Smith. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine v37 n3 (2020 05 01): 50-60*, 2020. ISSN 1053-5888. doi: 10.1109/MSP.2020.2975749.

[26] Nabil El Ioini and Claus Pahl. A review of distributed ledger technologies. In *On the Move to Meaningful Internet Systems. OTM 2018 Conferences: Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part II*, pages 277–288. Springer, 2018.

[27] B Vivekanadam. Analysis of recent trend and applications in blockchain technology. *Journal of ISMAC*, 2(04):200–206, 2020.

[28] Omar Dib, Kei-Leo Brousmiche, Antoine Durand, Eric Thea, and Elyes Ben Hamida. Consortium blockchains: Overview, applications and challenges. *Int. J. Adv. Telecommun*, 11(1):51–64, 2018.

[29] Bela Shrimali and Hiren B. Patel. Blockchain state-of-the-art: architecture, use cases, consensus, challenges and opportunities. *Journal of King Saud University - Computer and Information Sciences*, 34(9):6793–6807, 2022. ISSN 1319-1578. doi: https://doi.org/10.1016/j.jksuci.2021.08.005. URL https://www.sciencedirect.com/science/article/pii/S131915782100207X.

[30] Varun Kohli, Sombuddha Chakravarty, Vinay Chamola, Kuldip Singh Sangwan, and Sherali Zeadally. An analysis of energy consumption and carbon footprints of cryptocurrencies and possible solutions. *Digital Communications and Networks*, 9(1):79–89, 2023. ISSN 2352-8648. doi: https://doi.org/10.1016/j.dcan.2022.06.017. URL https://www.sciencedirect.com/science/article/pii/S2352864822001390.

[31] Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. On-device federated learning via blockchain and its latency analysis. *arXiv preprint arXiv:1808.03949*, 2018.

[32] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *arXiv*, October 2016. doi: 10.48550/arXiv.1610.02527.

[33] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[34] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony

Bharath. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Process. Mag.*, 34(6):26–38, November 2017. ISSN 1558-0792. doi: 10.1109/MSP.2017.2743240.

[35] Liudmyla Tsymbal and Iryna Uninets. Development of smart cities: Country analysis. *Management Theory and Studies for Rural Business and Infrastructure Development*, 44(4):482–488, Dec. 2022. doi: 10.15544/mts.2022.47. URL https://ejournals.vdu.lt/index.php/mtsrbid/article/view/3551.

[36] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Smart city observatory, 2021. URL https://wwwcontent.imd.org/smart-city-observatory/home/. [Accessed: 23. Jan. 2023].

[37] Shahid Sultan Hajam and Shabir Ahmad Sofi. Iot-fog architectures in smart city applications: A survey. *China Communications*, 18(11):117–140, 2021. doi: 10.23919/JCC.2021.11.009.

[38] Dudy Effendi, Ferra Syukri, Ahmad Fatoni Subiyanto, and Rona Nandana Utdityasan. Smart city nusantara development through the application of penta helix model (a practical study to develop smart city based on local wisdom). In *2016 International Conference on ICT For Smart Society (ICISS)*, pages 80–85, 2016. doi: 10.1109/ICTSS.2016.7792856.

[39] Smart Hospitals Powered by Intel and Intel's Ecosystem Partners, January 2023. URL https://www.intel.com/content/www/us/en/healthcare-it/resources/smart-hospital-business-brief.html. [Accessed: 1. Feb. 2023].

[40] Digital Hospital Whitepaper, January 2023. URL https://www.intel.com/content/www/us/en/healthcare-it/resources/digital-hospital-whitepaper.html. [Accessed: 1. Feb. 2023].

[41] Silvia Liberata Ullo and G. R. Sinha. Advances in iot and smart sensors for remote sensing and agriculture applications. *Remote Sensing*, 13(13):2585, Jul 2021. ISSN 2072-4292. doi: 10.3390/rs13132585. URL http://dx.doi.org/10.3390/rs13132585.

[42] Uferah Shafi, Rafia Mumtaz, José García-Nieto, Syed Ali Hassan, Syed Ali Raza Zaidi, and Naveed Iqbal. Precision agriculture techniques and practices: From considerations to applications. *Sensors*, 19(17):3796, 2019.

[43] Javier Rodríguez-Robles, Álvaro Martin, Sergio Martin, José A Ruipérez-Valiente, and Manuel Castro. Autonomous sensor network for rural agriculture environments, low cost, and energy self-charge. *Sustainability*, 12(15):5913, 2020.

[44] AA Raneesha Madushanki, Malka N Halgamuge, WAH Surangi Wirasagoda, and Syed Ali. Adoption of the internet of things (iot) in agriculture and smart farming towards urban greening: A review. *International Journal of Advanced Computer Science and Applications*, 10(4), 2019.

[45] Ioana Marcu, Carmen Voicu, Ana Maria Claudia Drăgulinescu, Octavian Fratu, George Suciu, Cristina Balaceanu, and Maria Madalina Andronache. Overview of iot basic platforms for precision agriculture. In *Future Access Enablers for Ubiquitous and Intelligent Infrastructures: 4th EAI International Conference, FABULOUS 2019, Sofia, Bulgaria, March 28-29, 2019, Proceedings 283*, pages 124–137. Springer, 2019.

[46] Gulfishan Mobin and Abhishek Roy. A literature review on cloud based smart transport system. In *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1245–1250, 2021. doi: 10.1109/ICOEI51242.2021.9452884.

[47] Hyeonjoo Kim et al. Analysis of how tesla creates core innovation capability. *International Journal of Business and Management*, 15(6):42–61, 2020.

[48] Adel Khelifi, Manar Abu Talib, Douae Nouichi, and Mohamed Salah Eltawil. Toward an Efficient Deployment of Open Source Software in the Internet of Vehicles Field. *Arabian Journal for Science and Engineering v44 n11 (201911): 8939-8961*, 2019. ISSN 2193-567X. doi: 10.1007/s13369-019-03870-2.

[49] Hamid Fekri Azgomi and Mo Jamshidi. A Brief Survey on Smart Community and Smart Transportation. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 932–939. IEEE, November 2018. doi: 10.1109/ICTAI.2018.00144.

[50] Fotios Zantalis, Grigorios Koulouras, Sotiris Karabetsos, and Dionisis Kandris. A review of machine learning and iot in smart transportation. *Future Internet*, 11(4), 2019. ISSN 1999-5903. doi: 10.3390/fi11040094. URL https://www.mdpi.com/1999-5903/11/4/94.

[51] Shady S. Refaat, Omar Ellabban, Sertac Bayhan, Haithem Abu-Rub, Frede Blaabjerg, and Miroslav M. Begovic. *Smart grid and enabling technologies*. Hoboken, NJ : Wiley, 2021., Hoboken, NJ, USA, 2021. ISBN 978-1-11942245-7. doi: 10.1002/9781119422464.

[52] Ibrahim Mashal. Smart grid reliability evaluation and assessment. *Kybernetes*, ahead-of-print(ahead-of-print), March 2022. doi: 10.1108/K-12-2020-0910.

[53] Giuseppe Fenza, Mariacristina Gallo, and Vincenzo Loia. Drift-aware methodology for anomaly detection in smart grid. *IEEE Access*, 7:9645–9657, 2019.

[54] Ilias Siniosoglou, Panagiotis Radoglou-Grammatikis, Georgios Efstathopoulos, Panagiotis Fouliras, and Panagiotis Sarigiannidis. A unified deep learning anomaly detection and classification approach for smart grid environments. *IEEE Transactions on Network and Service Management*, 18(2):1137–1151, 2021.

[55] Bruno Rossi, Stanislav Chren, Barbora Buhnova, and Tomas Pitner. Anomaly detection in smart grid data: An experience report. In *2016 ieee international conference on systems, man, and cybernetics (smc)*, pages 002313–002318. IEEE, 2016.

[56] Julius Jow, Yang Xiao, and Wenlin Han. A survey of intrusion detection systems in smart grid. *International Journal of Sensor Networks*, 23(3):170–186, 2017.

[57] Panagiotis I Radoglou-Grammatikis and Panagiotis G Sarigiannidis. Securing the smart grid: A comprehensive compilation of intrusion detection and prevention systems. *IEEE Access*, 7:46595–46620, 2019.

[58] Zubair A Baig and Abdul-Raoof Amoudi. An analysis of smart grid attacks and countermeasures. *J. Commun.*, 8(8):473–479, 2013.

[59] Xiaofang Xia, Yang Xiao, and Wei Liang. Sai: A suspicion assessment-based inspection algorithm to detect malicious users in smart grid. *IEEE Transactions on Information Forensics and Security*, 15:361–374, 2019.

[60] Pranay P Gaikwad, Jyotsna P Gabhane, and Snehal S Golait. A survey based on smart homes system using internet-of-things. In *2015 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*, pages 0330–0335. IEEE, 2015.

[61] Marie Chan, Eric Campo, Daniel Estève, and Jean-Yves Fourniols. Smart homes—current features and future perspectives. *Maturitas*, 64(2):90–97, 2009.

[62] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[63] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[64] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[65] Solmaz Niknam, Harpreet S. Dhillon, and Jeffrey H. Reed. Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine*, 58(6):46–51, 2020. doi: 10.1109/MCOM.001.1900461.

[66] Dinh C. Nguyen, Quoc-Viet Pham, Pubudu N. Pathirana, Ming Ding, Aruna Seneviratne, Zihuai Lin, Octavia Dobre, and Won-Joo Hwang. Federated learning for smart healthcare: A survey. *ACM Comput. Surv.*, 55(3), feb 2022. ISSN 0360-0300. doi: 10.1145/3501296. URL https://doi.org/10.1145/3501296.

[67] Waqar Ali, Rajesh Kumar, Zhiyi Deng, Yansong Wang, and Jie Shao. A federated learning approach for privacy protection in context-aware recommender systems. *Comput. J.*, 64:1016–1027, 2021.

[68] Subrato Bharati, M Mondal, Prajoy Podder, and VB Prasath. Federated learning: Applications, challenges and future scopes. *International Journal of Hybrid Intelligent Systems*, (Preprint):1–17, 2022.

[69] Jianwei Huang, Huaxiong Li, Ming Liu, Ling Ni, and Wei Zhang. Blockchain-based decentralized financial systems. *IEEE Transactions on Emerging Topics in Computing*, 5(2):273–284, 2017.

[70] Leonardo Batista, Lidiane França, and Thiago Silva. The role of digital currencies

in the financial industry: A study of bitcoin. *International Journal of Information Management*, 35(1):21–27, 2015.

[71] L Zhang, Y Xie, Y Zheng, and W Xue. The challenges and countermeasures of blockchain in finance and economics. *Journal of Business Economics and Management*, 21(5):888–901, 2020.

[72] Marko Budisic, Janez Zumer, and Blaz Rosko. Blockchain technology in finance: A review of the literature. *Journal of Business Economics and Management*, 20(3):569–581, 2019.

[73] Aditi Aggarwal et al. Blockchain for digital financial inclusion: A review of opportunities and challenges. *Journal of Financial Innovation*, 4(1):3, 2018.

[74] Simone Cimoli et al. The impact of blockchain technology on the financial industry: A systematic review. *Journal of Financial Stability*, 38:100794, 2019.

[75] Nabeel Al-Qirim et al. The future of blockchain in banking and financial services: An analytical survey. *Journal of Business Economics and Management*, 20(4):631–647, 2019.

[76] William H Aba and Marco M Musyoka. Blockchain in healthcare: Past, present, and future. *Journal of healthcare information management : JHIM*, 34(3):41–50, 2020.

[77] Mohammad R Gholami et al. Blockchain-based electronic health record: A review. *Journal of medical systems*, 43(11):484, 2019.

[78] Wei-Ting Kao et al. Blockchain-enabled secure health information exchange. *IEEE Journal of Biomedical and Health Informatics*, 22(6):1617–1625, 2018.

[79] Darach O'Ceallaigh et al. The potential of blockchain technology in healthcare: A systematic review. *Journal of medical systems*, 43(12):524, 2019.

[80] Hong-Ning Dai, Bing Li, Wei Liu, and Jia-Ning Chen. A blockchain-based framework for secure and transparent sharing of supply chain data. *IEEE Transactions on Industrial Informatics*, 13(5):2064–2072, 2017.

[81] Markus Taube, Matthias Froehlich, and Katharina Krombholz. Towards traceability in global supply chains with blockchain technology. *International Journal of Information Management*, 37:1–9, 2017.

[82] Guoyi Feng, Jiaxin Li, Wei Fan, and Mingsheng Song. The blockchain in supply chain management: A literature review and research directions. *International Journal of Information Management*, 40:1–12, 2019.

[83] Bharat Bhushan, Aditya Khamparia, K. Martin Sagayam, Sudhir Kumar Sharma, Mohd Abdul Ahad, and Narayan C. Debnath. Blockchain for smart cities: A review of architectures, integration trends and future research directions. *Sustainable Cities and Society*, 61:102360, 2020. ISSN 2210-6707. doi: https://doi.org/10.1016/j.scs.2020.102360. URL https://www.sciencedirect.com/science/article/pii/S2210670720305813.

[84] Xinyu Guo. Implementation of a blockchain-enabled federated learning model that supports security and privacy comparisons. In *2022 IEEE 5th International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pages 243–247. IEEE, 2022.

[85] Jiasi Weng, Jian Weng, Jilian Zhang, Ming Li, Yue Zhang, and Weiqi Luo. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2438–2455, 2019.

[86] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with het-

erogeneous resources in mobile edge. In *ICC 2019-2019 IEEE international conference on communications (ICC)*, pages 1–7. IEEE, 2019.

[87] Dinh C Nguyen, Ming Ding, Quoc-Viet Pham, Pubudu N Pathirana, Long Bao Le, Aruna Seneviratne, Jun Li, Dusit Niyato, and H Vincent Poor. Federated learning meets blockchain in edge computing: Opportunities and challenges. *IEEE Internet of Things Journal*, 8(16):12806–12825, 2021.

[88] Seyed Mojtaba Hosseini Bamakan, Amirhossein Motavali, and Alireza Babaei Bondarti. A survey of blockchain consensus algorithms performance evaluation criteria. *Expert Systems with Applications*, 154:113385, 2020.

[89] Y Supreet, P Vasudev, H Pavitra, Mouna Naravani, and DG Narayan. Performance evaluation of consensus algorithms in private blockchain networks. In *2020 International Conference on Advances in Computing, Communication & Materials (ICACCM)*, pages 449–453. IEEE, 2020.

[90] Tuyet Duong, Lei Fan, Jonathan Katz, Phuc Thai, and Hong-Sheng Zhou. 2-hop blockchain: Combining proof-of-work and proof-of-stake securely. In *European Symposium on Research in Computer Security*, 2020.

[91] Francesco Scicchitano, Angelica Liguori, Massimo Guarascio, Ettore Ritacco, and Giuseppe Manco. A deep learning approach for detecting security attacks on blockchain. In *ITASEC*, pages 212–222, 2020.

[92] Xinle Yang, Yang Chen, and Xiaohu Chen. Effective scheme against 51% attack on proof-of-work blockchain with history weighted information. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 261–265, 2019. doi: 10.1109/Blockchain.2019.00041.

[93] Loi Luu, Yaron Velner, Jason Teutsch, and Prateek Saxena. Smart pool: Practical decentralized pooled mining. In *USENIX Security Symposium*, pages 1409–1426, 2017.

[94] Bin Cao, Yixin Li, Lei Zhang, Long Zhang, Shahid Mumtaz, Zhenyu Zhou, and Mugen Peng. When internet of things meets blockchain: Challenges in distributed consensus. *IEEE Network*, 33(6):133–139, 2019. doi: 10.1109/MNET.2019.1900002.

[95] Hamid Malik, Ahsan Manzoor, Mika Ylianttila, and Madhusanka Liyanage. Performance analysis of blockchain based smart grids with ethereum and hyperledger implementations. In *2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–5, 2019. doi: 10.1109/ANTS47819.2019.9118072.

[96] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10, 2013. doi: 10.1109/P2P.2013.6688704.

[97] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[98] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[99] Chelsea C White III and Douglas J White. Markov decision processes. *European Journal of Operational Research*, 39(1):1–16, 1989.

[100] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, and Jong Wook Kim. Q-learning algorithms: A comprehensive classification and applications. *IEEE Access*, 7: 133653–133667, 2019. doi: 10.1109/ACCESS.2019.2941229.

[101] Harm van Seijen, Hado van Hasselt, Shimon Whiteson, and Marco Wiering. A theoretical and empirical analysis of expected sarsa. In *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 177–184, 2009. doi: 10.1109/ADPRL.2009.4927542.

[102] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[103] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[104] Bin Cao, Zhenghui Zhang, Daquan Feng, Shengli Zhang, Lei Zhang, Mugen Peng, and Yun Li. Performance analysis and comparison of pow, pos and dag based blockchains. *Digital Communications and Networks*, 6(4):480–485, 2020. ISSN 2352-8648. doi: https://doi.org/10.1016/j.dcan.2019.12.001. URL https://www.sciencedirect.com/science/article/pii/S2352864819301476.

[105] Joseph C. Gardiner, Cathy J. Bradley, and Marianne Huebner. 28 the cost-effectiveness ratio in the analysis of health care programs. In *Bioenvironmental and Public Health Statistics*, volume 18 of *Handbook of Statistics*, pages 841–869. Elsevier, 2000. doi: https://doi.org/10.1016/S0169-7161(00)18030-7. URL https://www.sciencedirect.com/science/article/pii/S0169716100180307.

[106] Jose Arturo Garza-Reyes. From measuring overall equipment effectiveness (oee) to overall resource effectiveness (ore). *Journal of Quality in Maintenance Engineering*, 21(4):506–527, Jan 2015. ISSN 1355-2511. doi: 10.1108/JQME-03-2014-0014. URL https://doi.org/10.1108/JQME-03-2014-0014.

[107] Kentaroh Toyoda, Koji Machi, Yutaka Ohtake, and Allan N. Zhang. Function-level bottleneck analysis of private proof-of-authority ethereum blockchain. *IEEE Access*, 8:141611–141621, 2020. doi: 10.1109/ACCESS.2020.3011876.

[108] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[109] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

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

[111] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2015.

[112] RIC Message Router (RMR) - RIC Platform - Confluence, March 2023. URL https://wiki.o-ran-sc.org/pages/viewpage.action?pageId=3605041. [Accessed: 31. Mar. 2023].

[113] Prem C Consul and Gaurav C Jain. A generalization of the poisson distribution. *Technometrics*, 15(4):791–799, 1973.

Appendices

Appendix A

Simulator Code

A.1 Helper Functions

```
# This function reads the collected data
# from csvfiles and converts into a distribution
def getDistributions():
    # time_distribution = {}
    miners=['Miner1','Miner2','Miner3','Miner4','Miner5']
    diff = ['5mil','7mil','13mil','17mil','23mil']
    distribution = [] # d[miner][difficulty]
    for i in range(5):
        row = []
        for j in range(5):
            filename= str(miners[i])+'_'+str(diff[j])
            filepath = os.path.join('distributions',filename+'.txt')
            with open(filepath,'rb') as f:
                l = pickle.load(f)
            row.append(l)
        distribution.append(row)
    return distribution

# Probability of winning
# proportional to the compute of miners
p_dict = {'miner1':0.1,
```

```
22            'miner2':0.2,
23            'miner3':0.1,
24            'miner4':0.2,
25            'miner5':0.4}
26
27 # This function acts as a mock blockchain and gives the next
28 # next state based on the action taken
29 # Input
30 # miners => list of selected miners
31 # p_dict => probability dict proportional to compute of miners
32 # window => how many blocks to consider in an observation
33 # distribution => the time distribution for each miner per difficulty value
       # def render(self, mode='human'):
34     #     pass
35 # difficulty => 5mil=0,7mil=1.....
36 # return a flattened array
37 def chooseWinner(miners:list,p_dict=p_dict,window=7,distribution=
     getDistributions(),difficulty=1):
38     prob_array = [p_dict[m] for m in miners]
39     prob_array= [float(i)/sum(prob_array) for i in prob_array]
40     state = []
41     vect_dict={'miner1':([1,0,0,0,0],0),
42                'miner2':([0,1,0,0,0],1),
43                'miner3':([0,0,1,0,0],2),
44                'miner4':([0,0,0,1,0],3),
45                'miner5':([0,0,0,0,1],4)}
46
47     for i in range(window):
48         draw = choice(miners, 1,
49                       p=prob_array)
50         m=vect_dict[draw[0]][1]
```

```
51            state.append(vect_dict[draw[0]][0]+[np.random.choice(distribution[m][
                  difficulty])])
52
53        return state
```

A.2 BlockChain Envirnment

```
1  import gym
2  from gym import spaces
3  import random
4  import numpy as np
5
6  class BlockChain(gym.Env):
7      """Custom Environment that follows gym interface"""
8      # Constraints
9      T1 = 0
10     T2 = 20
11     PHI=0.3
12     # Fairness Memory
13     q1 = deque([0.1]*10)
14     q2 = deque([0.1]*10)
15     q3 = deque([0.1]*10)
16     q4 = deque([0.1]*10)
17     q5 = deque([0.1]*10)
18     q_list = [q1,q2,q3,q4,q5]
19
20     data1 = [1,1,1,0,0]
21     data2 = [1,1,1,1,0]
22     data3 = [1,1,1,1,1]
```

```
23      d1 =list(set(permutations(data1)))
24      d2 = list(set(permutations(data2)))
25      d3 = list(set(permutations(data3)))
26      d = d1+d2+d3
27      d_final = []
28      difficulty = [5000000,7000000,13000000,17000000,23000000]
29      for i in (difficulty):
30          for l in d:
31              d_final.append(list(l)+[i])
32      action_dict = {i:val for i,val in enumerate(d_final)}
33
34      def make_row(self):
35          x1 = [1,0,0,0,0]
36          random.shuffle(x1)
37          y = random.randint(0,60)
38          return (x1+[y])
39
40      def __init__(self):
41          super(BlockChain, self).__init__()
42          # Define action and observation space
43          # They must be gym.spaces objects
44          # Example when using discrete actions:
45          self.action_space = spaces.Discrete(80)
46          # Example for using image as input (channel-first; channel-last also
                works):
47          self.observation_space = spaces.Box(low=0, high=500,
48                                              shape=(42,), dtype=np.float64)
49          # self.observation_space = spaces.Discrete(42)
50
51      def step(self, action):
52          rev_dict={'miner1':[1,0,0,0,0],
```

```
53                  'miner2':[0,1,0,0,0],
54                  'miner3':[0,0,1,0,0],
55                  'miner4':[0,0,0,1,0],
56                  'miner5':[0,0,0,0,1]}
57          selected_miners = []
58          for i,val in enumerate(self.action_dict[int(action)][:5]):
59              if val==1:
60                  selected_miners.append('miner'+str(i+1))
61          dif_dict = {5000000: 0, 7000000: 1, 13000000: 2, 17000000: 3,
                23000000: 4}
62          state = chooseWinner(miners=selected_miners,difficulty=dif_dict[self.
                action_dict[int(action)][5]])
63          self.observation = np.array(state).flatten()
64          reward = 0
65          fairness_count = 0
66          state_proportion = np.round(np.sum(state,axis=0)[:5]/len(state),2)
67          for i,q in enumerate(self.q_list):
68              ref_mean = np.mean(q)
69              q.popleft()
70              q.append(state_proportion[i])
71              new_mean = np.round(np.mean(q),2)
72              # Calculate reward
73              if ref_mean<=self.PHI:
74                  if new_mean<=self.PHI:
75                      reward+=40
76                      fairness_count+=1
77                  elif new_mean>self.PHI:
78                      reward-=180
79              else:
80                  if (new_mean-self.PHI)>(ref_mean-self.PHI):
81                      reward-=180
```

```
82                  elif new_mean<self.PHI:
83                      reward+=60
84                      fairness_count+=1
85                  else:
86                      reward+=30
87
88          count = 0
89          # count_time = 0
90          for i in range(len(state)):
91              if state[i][len(state[i])-1] < self.T1 or state[i][len(state[i])
                    -1] > self.T2:
92                  reward-=280
93              else:
94                  reward+=40
95                  count+=1
96
97          # Terminated or not
98          terminated = False
99          if count==len(state) and fairness_count == 5:
100             reward+=10000
101             terminated=True
102
103         self.reward = reward
104         self.info = {}
105         return self.observation, self.reward, self.done, self.info
106
107     def reset(self):
108         self.done = False
109         mat = []
110         for i in range(7):
111             mat.append(self.make_row())
```

```
112         observation =  np.array(mat).flatten()
113         return observation  # reward, done, info can't be included
114 
115     def render(self, mode='human'):     # def render(self, mode='human'):
116         percentages = []
117         for q in self.q_list:
118             # print(np.round(np.mean(q),2))
119             percentages.append(np.round(np.mean(q),2))
120         return percentages,self.reward
121 
122     def close (self):
123         print('Shutting down the blockchain')
```