SECURE AND INTELLIGENT RESOURCE ALLOCATION FOR LOW-LATENCY WIRELESS COMMUNICATIONS

By

Xin HAO

A THESIS SUBMITTED IN FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
FACULTY OF ENGINEERING

THE UNIVERSITY OF
SYDNEY

DECEMBER 2023

To my supporting mother,

my caring father,

and my sweet daughter.

Acknowledgements

My Ph.D. journey has been an incredible and transformative experience. I would like to express my deep gratitude to the many individuals who have supported and guided me throughout this remarkable journey.

I express my sincere gratitude to Prof. Yonghui Li, my lead supervisor, for graciously accepting me as his Ph.D. student and offering invaluable guidance. His generous assistance and insightful comments have profoundly enriched both my research and personal development throughout my candidature. Prof. Li consistently extends his support not only in the realm of research but also in navigating and adapting to the international learning environment. His unwavering patience and encouragement are evident whenever I seek assistance. I feel exceptionally fortunate and honored to have him as my supervisor.

I would also like to extend my heartfelt thanks to my auxiliary supervisor, Dr. Phee Lep Yeoh, for his valuable and responsible supervision. He has always respected my choices, exhibited remarkable patience, and provided unwavering support. Working with him makes me feel confident, accomplished, and relaxed even though the Ph.D. is usually particularly stressful and overwhelming. Through our discussions, mentorship has honed my skills in literature review, critical thinking, and logical writing. Beyond his technical expertise, his outlook on life has offered valuable life lessons that will undoubtedly shape my future. Furthermore, I also appreciate his funding to support me in attending the top conference in our area as well as visiting the Technical University of Munich and Southwest Jiaotong University. Lastly, I thank him for the helpful discussions for all my publications during my Ph.D. candidature, as well as for helping organize and revise this thesis. Without his comments, suggestions, and discussions, I could never achieve the academic publications and this thesis.

I am equally grateful for the guidance of my auxiliary supervisor, Dr. Changyang She, whose expertise in machine learning algorithms proved crucial to my research.

His patient and systematic approach were invaluable as I delved into the intricacies of machine learning. I extend my thanks to him not only for his assistance in revising journal papers but also for taking the time to explain the underlying logic. His support has been instrumental in shaping my academic journey. Additionally, I appreciate his thoughtful guidance in career planning during moments of confusion.

I extend my gratitude to Prof. Branka Vucetic for her invaluable contributions to my research journey. Her insightful comments, suggestions, and engaging discussions have significantly enhanced the depth of my work. I am particularly thankful for her support in accessing computing resources when needed. Prof. Vucetic's understanding, patience, and unwavering support have played a pivotal role in overcoming challenges and striving for excellence in my endeavors. Additionally, her assistance during conference attendance has been instrumental in expanding academic horizons.

My Ph.D. journey has been enriched by my new friends, colleagues and co-authors. I would like to thank Ayoob Salari, Chentao Yue, Guangchen Wang, Haiyao Yu, Jiazheng Chen, Jingjing Guo, Ke Wang, Kou Tian, Litianyi Zhang, Qianfei Zhou, Rui Wang, Tian Qiu, Wenjian Hu, Xuemeng Liu, Yao Yu, Yi Zhou, Yuhong Liu, and Zhaoquan Geng for their companionship and shared academic experiences.

I would like to thank my various sources of financial support. Firstly, I would like to thank the Faculty of Engineering Research Scholarship, which covers my tuition fees as well as living expenses. Secondly, the assistance from the Postgraduate Research Support Scheme in covering part of the conference attendance fees is greatly appreciated. Thirdly, I would like to express my gratitude to the Faculty of Engineering Career Advancement Award for their invaluable support in enabling me to pursue my research while actively seeking employment opportunities.

I extend my gratitude to the examiners of my thesis for their valuable contributions in improving its quality. Their insightful feedback and guidance have played a pivotal role in enhancing this work. Thank you for your dedication and expertise.

I save my deepest gratitude to my family members. To my mother, thank you for your unwavering support, for taking care of my baby daughter, for trusting in me, and for your constant encouragement and understanding. To my father, your silent care and support have meant the world to me. To my sweet daughter, thank you for your understanding and patience during my busy academic pursuits.

My Ph.D. achievement would not have been possible without the collective support, encouragement, and guidance from all those mentioned above. Thank you all from the bottom of my heart.

Statement of Originality

I hereby declare that the work presented in the thesis is the result of the original research carried out by myself, in collaboration with my supervisors, while enrolled in the School of Electrical and Computer Engineering at the University of Sydney as a candidate for the Doctor of Philosophy.

These studies were conducted under the supervision of Prof. Yonghui Li, Dr. Phee Lep Yeoh, and Dr. Changyang She. They have not been submitted for any other degree or award in any other university or educational institution.

Xin HAO
School of Electrical and Computer Engineering
The University of Sydney
December 2023

Acknowledgement of Authorship

I hereby certify that the work embodied in this thesis contains published papers or scholarly work of which I am a joint author. I have included as part of the thesis a written declaration.

Chapter 3 of this thesis is based on materials in the journal paper [J1] and the conference paper [C1] as listed in Section 1.4. My role involved creating the analytical framework, conducting all simulation experiments, interpreting the results, drafting and revising the papers, and writing the response letters to the reviewers

Chapter 4 of this thesis contains materials from the journal paper [J2] and the conference paper [C3] as listed in Section 1.4. My work in this chapter included creating the analytical framework, conducting all simulation experiments, interpreting the results, drafting and revising the papers, and writing the response letter to the reviewers.

Chapter 5 draws upon content from the journal paper [J3] and the conference paper [C2] as detailed in Section 1.4. My work in this chapter included creating the analytical framework, conducting all simulation experiments, interpreting the results, and drafting and revising the papers.

In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

Xin Hao, December, 2023

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Yonghui Li, December, 2023

Abstract

Low-latency wireless communications have attracted intense attention due to emerging network applications such as online gaming, video conferencing, autonomous driving, remote surgery, and virtual reality. A key challenge for achieving low-latency communications is to design optimal resource allocation frameworks that can satisfy multiple network and user requirements since storage, computing, and bandwidth resources are severely limited in real-world wireless communications. An important consideration for resource allocation is to satisfy diverse network and user security requirements due to a wide range of attacks that can jeopardize data security and service-provisioning of low-latency wireless communications. Furthermore, resource allocation needs to be highly adaptable to the dynamics of users and wireless networks in diverse network scenarios, including the Internet of Things (IoT), mobile edge computing (MEC), and network slicing (NS). In this thesis, in light of above challenges, we propose novel resource allocation frameworks that can effectively reduce latency whilst resisting communications and data security attacks, as well as intelligently adapting to the dynamics of users in a range of emerging wireless network scenarios.

First, we consider how to support low-latency communications in a hierarchical IoT network in the presence of data tampering attacks. To guarantee data security, we consider storing the IoT data in a novel decentralized blockchain structure that we have designed with in-built tampering-proof properties. Specifically, to solve the high computing complexities in conventional blockchains, we design a double blockchain (DBC) architecture composing an information blockchain (IBC) and a reputation blockchain (RBC). The IBC is a heavyweight blockchain that stores large amounts of IoT data in the resource-rich cloud layer, whilst the RBC is a lightweight blockchain that stores the reputation data of the IoT devices in the edge layer. To quickly identify malicious tampering of the IoT data stored in IBC, we propose a mapping protocol for the fast querying of their corresponding reputation data in RBC. To reduce the latency for data processing in the edge layer, we develop a blockchain node classification algorithm that assigns base stations (BSs) with different blockchain tasks according to their resource capacities. Stochastic network analysis is presented to validate the effectiveness of our DBC in a hierarchical IoT network. Simulation results demonstrate that the transmission and processing latencies are significantly reduced in DBC compared with a conventional single blockchain architecture.

Next, we focus on reducing latency in an MEC network, where the MEC service provider in the cloud layer supports multiple users by employing multiple BSs in the edge layer. To prevent data tampering attacks, we consider securely storing MEC user requests in a blockchain and designing a low-latency reputation-based proof-of-stake (RPoS) consensus protocol to select highly reliable blockchain-enabled BSs. Both time-varying user requests and a dynamic network requirement of maximum denial-of-service (DoS) probability are considered in the latency minimization optimization problem. We design a constrained deep reinforcement learning (DRL) algorithm to efficiently optimize the computation resource allocation for both the blockchain consensus protocol management and the MEC service-provisioning. To further reduce the latency for training the DRL algorithm, we apply aggregation-enabled feature engineering and transfer learning to handle the dynamics of users and the MEC network. Simulation results confirm that the proposed constrained DRL can achieve the minimum latency with a satisfactory DoS probability when the dynamics of both the DoS probability constraint and time-varying user requests are considered.

Finally, we consider a generalized network model supporting diverse quality-of-service (QoS) requirements for a dynamic number of users and non-stationary wireless channels. Our model is suitable for network slicing (NS) where low latency, high data security, and high data rate QoS requirements need to be satisfied with the same physical infrastructure. We design a graph neural network (GNN) that adapts to the changing wireless network topology with dynamic user numbers, and unsupervised learning is used to train the GNN. To handle policy mismatch caused by non-stationary wireless channels during the training and testing stages of the GNN, we propose a new hybrid-task meta-learning (HML) algorithm, which considers different channel distributions in the meta-training stage of the GNN. Compared with the state-of-the-art model-agnostic meta-learning (MAML) algorithm, our HML algorithm can significantly improve sampling efficiency by 73%. Numerical results show that our combination of GNN and meta-learning can successfully generalize the low-latency wireless network requirements, including high security, high data rate, and low-latency QoS with dynamic number of users and non-stationary wireless channels.

Table of Contents

List of Tables

List of Figures

List of Acronyms

BS	base station
CSI	channel state information
DBC	double blockchain
DNN	deep neural network
DDPG	deep deterministic policy gradient
DL	deep learning
DRL	deep reinforcement learning
DoS	denial-of-service
DQN	Deep Q-Network
eMBB	enhanced mobile broadband
5G	fifth-generation
FNN	fully-connected neural network
GNN	graph neural network
HML	hybrid-task meta-learning
IB	information block
IBC	information blockchain
IoT	Internet of Things
MAML	model-agnostic meta-learning
MDP	Markov decision process
MEC	mobile edge computing
MPNN	message passing neural network

NN	neural network
NS	network slicing
NP-hard	non-deterministic polynomial-time hard
RB	reputation block
RBC	reputation blockchain
6G	sixth-generation
SGA	stochastic gradient ascent
SGD	stochastic gradient descent
PD-DDPG	primal-dual deep deterministic policy gradient
PDF	probability distribution function
PMF	probability mass function
QoS	quality-of-service
SINR	signal-to-interference-plus-noise ratio
SNR	signal-to-noise ratio
T-NT	terrestrial and non-terrestrial
VR	virtual reality

List of Symbols and Notations

T_c	the channel coherence time
κ_{bc}	the coefficient of CPU cycles for blockchain management
κ_{sp}	the coefficient of CPU cycles for MEC service-provisioning
F	the computation capacity of a base station
$O(\cdot)$	the computational complexity
ϵ_u	the decoding error probability of the u-th user
γ_c	the discount factor of cost
γ_r	the discount factor of reward
T_s	the duration of one time slot
$\mathbb{E}[\cdot]$	the expectation operator
$f_{\mathrm{PDF}}(\cdot)$	the function of PDF
$f_{\mathrm{PMF}}(\cdot)$	the function of PMF
$f_{\mathrm{FNN}}(\cdot)$	the function of FNN
$f_{\mathrm{GNN}}(\cdot)$	the function of GNN
$f_{\mathtt{Concat}}(\cdot)$	the function of Concatenation
$f_{\mathtt{Readout}}(\cdot)$	the function of Readout
$f_{\mathtt{Softmax}}(\cdot)$	the function of `Softmax`
$\mathbb{1}\{\cdot\}$	the indicator function
λ_{fog}	the intensity/density of the entire fog node set
λ_{iot}	the intensity/density of the transmitting IoT devices
$\lambda_{\mathcal{L}}$	the Lagrangian dual variable

β_r	the learning rate of critic reward NN
β_c	the learning rate of critic cost NN
β_a	the learning rate of actor NN
β_θ	the learning rate of GNN
β_ϕ	the learning rate of meta-training
R	the long-term reward
C	the long-term cost
r	the instantaneous reward
c	the instantaneous cost
$Q(\cdot)$	the Q-function
P_u	the transmit power of the u-th node/BS
N_{block}	the number of blocks in one blockchain
L_{block}	the size of one block
L_{head}	the size of one block header
L_{body}	the size of one block body
L_h	the size of non-hash entry in the block body
L_b	the size of each entry in the block body
N_0	the single-sided noise spectral density
δ_u	the information leakage of the u-th user
$\Pr\{A\}$	the probability of the event A
$\Pr\{A\|B\}$	the conditional probability of event A under a given condition B
$\Gamma(\cdot)$	the gamma function.
ξ_i	the reputation of the i-th node
τ_i	the transmitting/processing time of the i-th node
$\hat{\alpha}$	the path loss exponent
α	the large-scale channel gain
g	the small-scale channel gain
N_{B}	the number of base stations
Δf	the minimum service rate can be allocated

Δw	the minimum size of bandwidth resource block
ϵ_{DoS}	the threshold of denial-of-service probability
ϵ_{SIR}	the threshold of SIR

Chapter 1

Introduction

1.1 Low-Latency Wireless Communications

Low-latency communication is a critical requirement for current and future generation wireless systems aimed at transmitting, receiving, and processing data with minimal end-to-end delays [Nasrallah *et al.*, 2019; Parvez *et al.*, 2018]. In recent years, intense attention has been paid to realizing low-latency wireless communications due to the increasing demand for real-time or near real-time services in newly-emerged applications, such as video conferencing, autonomous driving, remote surgery, online gaming, and virtual reality [Dang *et al.*, 2023; Shi *et al.*, 2023; Wu *et al.*, 2023]. One major challenge in implementing these low-latency communication services is to design efficient storage, computing, and bandwidth resource allocation policies for different network scenarios such as the Internet of Things (IoT), mobile edge computing (MEC), and network slicing (NS) [Deng *et al.*, 2022; Ding *et al.*, 2022; Xiao *et al.*, 2023; Zhou *et al.*, 2022].

As shown in Fig. 1.1, we consider that low-latency wireless communication networks can suffer from security-related performance degradation due to attacks from malicious base stations (BSs) or malicious users. For example, a malicious BS may

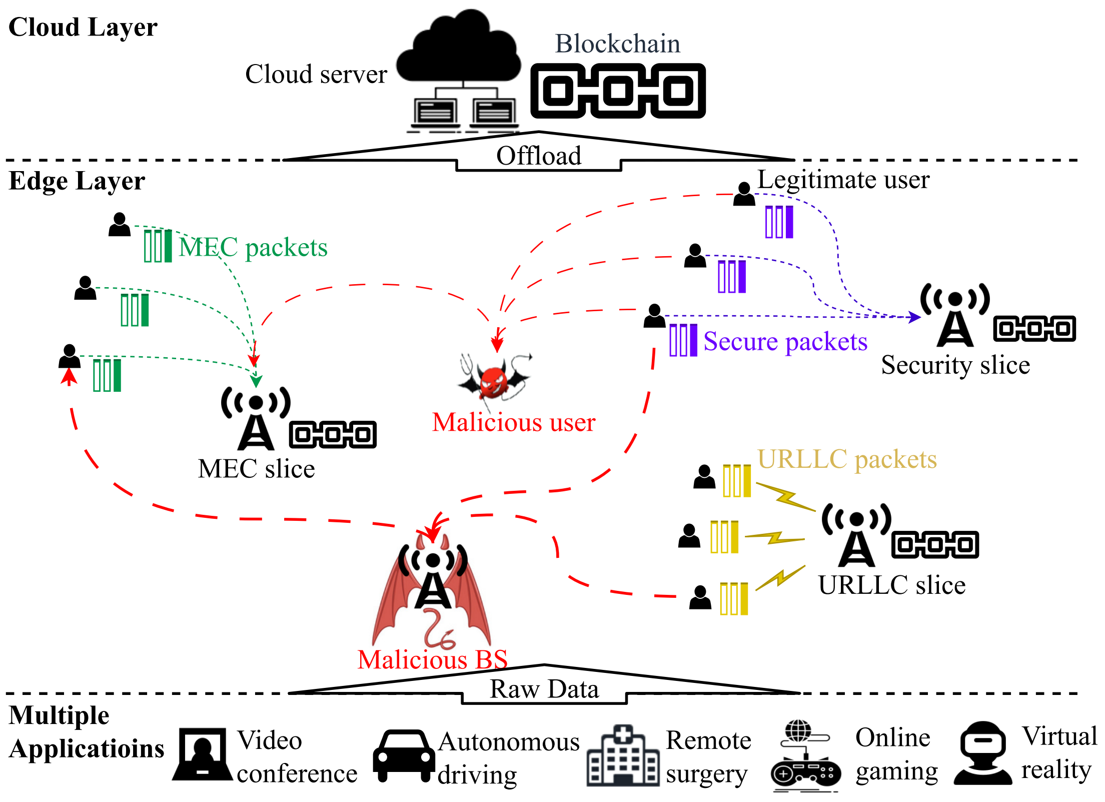


Figure 1.1: Architectural diagram for secure low-latency wireless communication networks including: a) Hierarchical cloud-edge IoT architecture; b) Scalable cloud-edge blockchains to prevent security attacks at edge layer; c) Network slicing supporting different service requirements, such as MEC, URLLC, and security. In this figure, the malicious BS is sending fake information to the user in the MEC slice, as well as eavesdropping the information transmitted in the security and URLLC slices. Simultaneously, the malicious user is sending the BS in the MEC slice some random feedback, which cannot truly reflect the MEC service-provisioning condition. Furthermore, this same malicious user is eavesdropping on the confidential information transmitted from the users to the BSs in the security and URLLC slices.

be transmitting interfering data to legitimate users in different network slices, whilst a malicious user could be actively modifying data sent to a legitimate BS or eavesdropping secure links from legitimate users. To prevent such attacks, we will design new hierarchical blockchain architectures to ensure a high level of data integrity both at the central cloud server as well as at the edge layer base stations. Our aim is to achieve secure low-latency wireless communications with efficient storage, computing, and bandwidth allocation based on highly dynamic requirements arising from both the networks and users.

1.1.1 Internet of Things

The increasing global demand for wireless-enabled smart devices has led to a significant expansion of the so-called Internet-of-Things (IoT) paradigm. As predicted by Cisco, there will be 500 billion wireless devices connected to the Internet by the year 2030 [Shafique *et al.*, 2020]. To accommodate such a vast number of IoT devices, cloud computing servers have been integrated into wireless communication networks, such as Google Cloud IoT, Cisco IoT Cloud Connect, and Microsoft Azure IoT Hub [Vaezi *et al.*, 2022], to efficiently handle resource-intensive network traffic. However, the number of the resource-rich cloud servers is limited and usually located far from the edge users, thus using conventional cloud server architectures to serve large numbers of wireless edge users can lead to lengthy delays in completing a single message transmission. This delay, which is referred to as communication latency, similar to the geographical distance between the satellites and end users [Dakic *et al.*, 2023; Homssi *et al.*, 2023], is largely due to the significant geographical distance between the cloud servers and edge users in wireless networks [Rodrigues *et al.*, 2020; Shakarami *et al.*, 2020; Spinelli and Mancuso, 2021]. While significant latency can be attributed to network congestion and queuing, exploring transmission latency is worthwhile in the hierarchical IoT context considered in this thesis.

Hierarchical cloud-edge network architectures have been considered to reduce latency by employing edge servers in addition to the powerful cloud servers [Garadi *et al.*, 2020; Pan J. and J. McElhannon, 2018]. Within the cloud-edge architecture, the edge-layer servers can process numerous lightweight requests from the wireless IoT edge users, thus reducing the latency associated with communications to the cloud server. Fig. 1.1 illustrates our cloud-edge IoT network scenario that supports low-latency wireless communications. In this IoT network, raw data is transmitted to the edge layer from user applications with low-latency requirements, such as video conferencing, autonomous driving, remote surgery, online gaming, and virtual reality. We will consider that the raw data can be classified into two types, namely lightweight data that is processed or stored at the edge servers, whilst heavyweight data is offloaded to the cloud layer for further processing and storage.

1.1.2 Mobile Edge Computing

The mobile edge computing (MEC) architecture is designed to provide highly reliable wireless links between the users and edge-layer MEC servers, which makes it well-suited to supporting low-latency wireless communications [Rodrigues *et al.*, 2020]. As shown in Fig. 1.1, to reduce the latency between the users and the cloud server, MEC-enabled BSs are employed to serve the edge users. A major challenge in MEC service-provisioning that we will consider is the potential for malicious BSs or malicious users to tamper with data stored at the MEC servers. Recent works have suggested that blockchain, which is a secure distributed data structure, could be deployed at MEC servers to prevent tampering attacks in MEC networks [Asheralieva *et al.*, 2020; Feng *et al.*, 2020b; Xiao *et al.*, 2020].

To enhance security, we consider in Fig. 1.1 that our proposed scalable blockchain is integrated into both the cloud and edge networks. The decentralized nature of our blockchain-secured MEC network can effectively eliminate the single-point-of-failure that commonly exists in conventional centralized communication systems, and substantially increases the complexity for attackers to compromise the MEC data services [Ranaweera *et al.*, 2021]. Our scalable blockchain allows edge servers to securely offload data to the cloud server to alleviate the resource consumption burden associated with conventional blockchains whilst still ensuring satisfactory low-latency MEC services [Rafique *et al.*, 2020].

1.1.3 Network Slicing

A third consideration in this thesis is the emerging paradigm of network slicing (NS) to address the significant demands placed on network operators to support diverse wireless services using a shared physical infrastructure [Wijethilaka and Liyanage, 2021]. NS is a key technology for future-generation wireless networks to allocate services into separate logical networks based on their distinct requirements or characteristics using software-defined networking rules. Each logical network, which is referred to as a slice, can be supported by the same physical network resulting in significant cost and efficiency savings for network operators [Afolabi *et al.*, 2018].

The network slice is considered to be different from one category to another by different quality-of-service (QoS) requirements. For example, the Security slice focuses on maximizing the secrecy rate, whilst the URLLC slice targets both latency and decoding error probability. To satisfy the different QoS requirements, we optimize the resource allocation in the network slices sharing unique physical infrastructure. In Fig. 1.1, we consider that the wireless network can simultaneously support three different types of slices: URLLC slice, MEC slice, and Security slice.

1.2 Key Challenges and Existing Solutions

The open nature of wireless channels raises increasing security concerns about data tampering and eavesdropping attacks. While allocating additional computing and bandwidth resources for security management can help to resist these attacks, the support of low-latency communications could be compromised due to reduced allocations from the limited total resource pool. Machine learning has been recently recognized as a powerful technology for improving resource allocation especially under dynamic constraints. Indeed, the design of secure and intelligent resource allocation algorithms to satisfy the dynamic requirements of users and networks in low-latency wireless communications remains a key challenge due to the limitations of conventional numerical optimization approaches such as dynamic programming and iterative algorithms.

1.2.1 Data Security

Data security plays a pivotal role in protecting low-latency applications. For example, in remote surgery, the tampered historical data can compromise the integrity of treatment decisions, and the data leakage of personal information poses concerns regarding privacy as well. There remain significant challenges in resisting potential tampering and eavesdropping attacks launched by malicious BSs and users in wireless networks.

Scalable Blockchains to Prevent Tampering Attacks

A tampering attack refers to unauthorized malicious actions that intentionally manipulate data in transmission or storage. Successful tampering attacks in wireless communication systems can lead to undesirable consequences, including denial-of-service (DoS), privacy violations, and financial loss. Conventional centralized communication systems manage data through a centralized controller, which is vulnerable to single-point-of-failure [Cao *et al.*, 2023]. Blockchain, a distributed ledger, has been proven to be an effective tool due to its tamper-proofing property [Liu *et al.*, 2023], which has been demonstrated by the success of *Bitcoin* as a secure digital currency [Nakamoto, 2008].

In recent years, blockchain has been considered for identifying and resisting tampering attacks in wireless communication networks. In [Yu *et al.*, 2020], the authors applied blockchain to resist tampering attacks on the reputation data in a wireless crowdsourcing IoT network. In [Feng *et al.*, 2020a], the authors proposed using blockchain to ensure the irreversibility of the offloading data in MEC networks. In [Zanzi *et al.*, 2020], the authors secured an NS network by employing a blockchain as a trusted broker between the infrastructure provider and network tenants. In [Liu *et al.*, 2020], the authors proposed using blockchain to secure the authentication process in a vehicular network, which is typically a highly dynamic environment due to traffic mobility.

Given the fact that latency can be significantly reduced by increasing communications and computing resources, a key challenge we will address in this thesis is how to efficiently balance the allocation of resources between the low-latency communication requirement and the blockchain consensus protocol design. This is particularly notable when integrating blockchain into low-latency wireless communications, such as IoT networks with constraints on storage, communication, and computing resources.

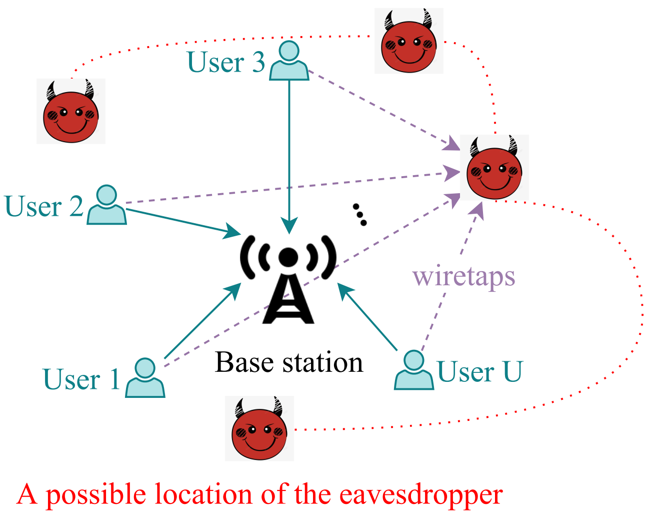


Figure 1.2: U legitimate users transmit confidential information to a BS, and a strong eavesdropper intends to eavesdrop the transmitted information.

We will investigate scalable cloud-edge blockchain architectures as a promising solution to satisfy wireless applications with low-latency requirements in the edge layer, whilst resisting data tampering attacks by storing different amounts of data in the cloud and edge-layer blockchains. In addition, we focus on developing a scalable blockchain consensus protocol to satisfy edge-layer low-latency requirements.

Machine Learning-Based Optimization to Prevent Eavesdropping Attacks

The eavesdropping attack is a well-known physical-layer security attack, which jeopardizes data security by eavesdropping the transmitted information [Karas *et al.*, 2016]. To quantify the effectiveness of security measures against eavesdropping attacks, the concept of secrecy rate comes into play [Gopala *et al.*, 2008]. Fig. 1.2 shows a wireless network with U randomly located legitimate users transmitting confidential information to the BS in the presence of a moving eavesdropper. In this network, we denote the data rate of the u-th legitimate user and the eavesdropping channel by $r_u^D = \log_2(1 + \mathrm{SNR}_u^D)$ and $r_u^e = \log_2(1 + \mathrm{SNR}_u^e)$, where SNR_u^D and SNR_u^e are the signal-to-noise ratios (SNRs) of the u-th legitimate and the eavesdropping channel, respectively. The secrecy rate of the u-th user is given by

$$r_u^S = \left[r_u^D - r_u^e\right]^+, \tag{1.2.1}$$

where $[x]^+ = \max\{0, x\}$.

To resist eavesdropping attacks, significant efforts have been devoted to maximizing the secrecy rate in wireless networks. The authors of [Kang *et al.*, 2019a; Li *et al.*, 2016; Zhou *et al.*, 2018] developed different optimization approaches to accurately evaluate and evaluate the secrecy rates. However, the high computational complexity associated with conventional numerical optimization algorithms poses challenges in real-world implementations. Compared with conventional numerical optimization algorithms like dynamic programming and iterative algorithms, machine learning algorithms have been applied to optimize the secrecy rate in wireless networks due to the low computational complexity advantage. In [He *et al.*, 2019], the authors showed that learning-based algorithms could obtain near-optimal solutions achieved by exhaustive search and perform in real-time and with negligible latency impact. In [Gao *et al.*, 2020], a deep neural network (DNN) was used to maximize the secrecy rate of a legitimate user and eavesdropper by jointly optimizing the power allocation, carrier frequency, transmit power, and waveform. In [Zhang *et al.*, 2021b], the authors applied a DNN to optimize the transmit power allocation aimed at maximizing the secrecy rate of a single user subject to an interference leakage threshold constraint. In [Sharma *et al.*, 2011], the authors used deep reinforcement learning (DRL) to maximize the average secrecy rate of multiple legitimate users by optimizing the downlink transmit power. In [Li *et al.*, 2022], the authors proposed an unsupervised learning algorithm to maximize the effective secrecy rate for the short packets with an extremely low-latency requirement of 1 ms.

However, existing machine learning algorithms for wireless communications rely on the assumption that the training and testing channels follow the same distribution, which usually does not hold due to the non-stationary nature of wireless channels. The low-latency requirement for the newly emerged applications further amplifies the challenges for eavesdropping attack-resistant. The problem of simultaneously achieving low latency and resisting the eavesdropping attack in non-stationary wireless networks remains an open issue.

1.2.2 Dynamics of Users

Wireless communication networks typically serve a dynamic number of users, and the user requests often change over time. These dynamics pose challenges in efficiently managing the physical resources in low-latency wireless communication.

Dynamic Numbers of Users

Conventional numerical optimization approaches, such as the iterative algorithm, are well-established solutions for identifying optimal resource allocation policies in dynamic wireless communication networks [Dong *et al.*, 2021]. However, low-latency optimization with dynamic numbers of users poses an important challenge in achieving algorithm efficiency for optimal resource allocation policies [Shen *et al.*, 2021]. Machine learning offers promising solutions for addressing this challenge with strategies such as offline training and online adaptation. In the following, we will briefly introduce novel machine learning concepts that we will apply in this thesis to address the challenge of low-latency resource optimization with dynamic numbers of users.

Time-Varying User Requests

The second critical challenge for user dynamics that we will address in this thesis is how to efficiently serve users with time-varying requests [Spinelli and Mancuso, 2021]. Dynamic resource allocation is a powerful approach for allocating resources to current user requests without negatively impacting future user requests [Chekired *et al.*, 2019; Wijethilaka and Liyanage, 2021]. A fundamental trade-off is that allocating more resources in the current time slot results in a smaller processing latency for these users, but a potential performance degradation due to insufficient resources for future users [Zappone *et al.*, 2019]. We will show that this trade-off can be managed by formulating the optimal resource allocation as a sequential decision-making problem [Chen *et al.*, 2022; Tang *et al.*, 2021].

Although deep reinforcement learning (DRL) is a promising tool for solving sequential decision-making problems, it can face challenges when dealing with continuous state and action spaces, as observed in previous works that employed deep Q networks (DQN) [Wang *et al.*, 2022a]. To overcome this limitation, unconstrained deep deterministic policy gradient (DDPG) has been applied to low-latency wireless communications, incorporating optimization objectives and constraints into the reward function [Wang *et al.*, 2023]. However, in the current 5G and future 6G wireless communication networks, time-varying network requirements, e.g., maximum denial-of-service (DoS) probability, maximum transmit power, and minimum data rate, are commonly included in addition to the low-latency requirement. Novel solutions are expected to support sequential decision-making problems considering multiple performance metrics.

1.2.3 Dynamics of Wireless Networks

In our low-latency optimization frameworks, we will also consider the dynamic requirements on network performance, particularly due to network slicing and the impact of non-stationary wireless channels changing over time. These two factors further increase the challenges in achieving an optimal resource allocation policy for low-latency wireless communications.

Changing Requirements on Network Performance

In addition to the time-varying user requests, requirements on network performance make it more challenging to achieve efficient storage, computing, and bandwidth allocation algorithms for low-latency wireless communications. To solve this issue, different technologies have been proposed in the existing literature. In the study by Xiong *et al.*, 2020, the authors used a conventional unconstrained DRL algorithm to minimize an objective function that combines user computing processing time and allocated computation resources in an MEC network. In [Alsenwi *et al.*, 2021], the authors maximized the data rate of enhanced mobile broadband (eMBB) users while satisfying the latency constraint on URLLC users by an unconstrained DRL algorithm. However, unconstrained DRL encounters difficulties in explicitly satisfying both time-varying user and dynamic network performance constraints, which poses a trade-off between the dynamic requirements of users and the network.

Recently, constrained DRL algorithms have been developed for dynamic resource allocation in low-latency wireless communications. In [Li *et al.*, 2021a], constrained DDPG was applied in a virtual reality network to minimize the video loss ratio while satisfying a constraint on processing latency. In [Xu *et al.*, 2021b], a constrained

DRL algorithm was used in a wireless network to maximize the data throughput whilst satisfying QoS constraints of energy and queue length. Nevertheless, how to resist tampering attacks in low-latency wireless networks with time-varying user and network requirements remains an open problem for further investigation.

Non-Stationary Wireless Channels

Wireless communications also face the challenge of non-stationary wireless channels. It is especially challenging in communication networks with low-latency requirements. This is because machine learning algorithms encounter difficulties in timely adjusting the resource allocation policies to the varying channel distributions in wireless networks in the training and testing stages. While scaling up neural networks can boost performance, it comes at the cost of increased storage and computational resources. Scalable GNNs offer a solution by adjusting their scale based on user demands. However, developing innovative strategies is essential to enhance GNN adaptability in low-latency communications with dynamic wireless channels.

In future-generation communication networks that need to address dynamic numbers of users and non-stationary wireless channels, there are typically multiple network performance requirements. In [Wang *et al.*, 2020], the achievable secrecy rate was maximized by optimizing the blocklength. In [Li *et al.*, 2022], the authors maximized the effective achievable secrecy rate by optimizing the power control policy. In [Alsenwi *et al.*, 2021], the authors maximized the Shannon capacity as well as the reliability constraint on short packets by jointly optimizing the bandwidth, power, and punctured time slots. In [Dong *et al.*, 2021], three different QoS requirements, long packets-based data rate, short packets-based data rate, and long packets-based latency, are simultaneously optimized in a cascaded neural network. To date, no general bandwidth allocation approach has been proposed to simultaneously satisfy the QoS requirements on data rate, latency, and security in both long and short coding blocklength regimes.

In [Guo and Yang, 2023], optimizations of data rate in long coding blocklength regimes are analyzed, and a deep neural network structure was proposed to reduce the training complexity and guarantee spectral efficiency. In [Guo *et al.*, 2019], the authors explored maximizing the sum Shannon capacity by optimizing the power and bandwidth for short blocklengths. To guarantee a QoS requirement, the authors of [Wu *et al.*, 2003] defined effective capacity, denoting the maximum constant arrival rate that a given service process can support. In [Tang and Zhang, 2007], the authors maximized the effective capacity subject to a given QoS requirement of latency. Security is also an important QoS requirement in wireless systems, where optimizing for secrecy rate is very challenging as related problems are often non-convex and not in closed forms [Yu *et al.*, 2016]. In [Yang *et al.*, 2021], the authors formulated an optimization problem maximizing the minimal secrecy rate of all the users, and solved the problem by transforming the objective function to be the weighted sum of the secrecy rate, secrecy rate violation probability, and Shannon capacity violation probability. In [Liu *et al.*, 2019a], the authors optimized the secrecy outage probability in the long blocklength regime. However, none of the previous works proposed a general approach for simultaneously solving the data rate, latency, and security requirements for long or short packets with the consideration of non-stationary wireless channels.

1.3 Research Problems and Contributions

In this thesis, we focus on *secure and intelligent resource allocation for low-latency wireless communications*. We propose three novel solutions, namely DBC, BC-DRL, and HML, to reduce the latency for IoT, MEC, and NS networks with guaranteed security levels. The research problems and the corresponding contributions are described in this section.

1.3.1 IoT Blockchains

In Chapter 3, we propose a decentralized double blockchain (DBC) architecture for scalable, lightweight, and secure IoT information and reputation management. The results in this chapter have been published in the IEEE Internet of Things Journal and in the 2021 IEEE WF-IoT conference.

The DBC is a private blockchain deployed on a cloud-fog communication network which is composed of the information blockchain (IBC) storing large amounts of IoT data in the cloud layer and a reputation blockchain (RBC) storing reputation data of the IoT devices in the near-terminal fog layer. To improve the efficiency of IBC and RBC generation and RBC storage in the fog layer, we propose a fog layer node classification algorithm to assign different tasks for fog nodes holding different amounts of resources. To improve security, we propose a mapping algorithm between the lightweight RBC and heavyweight IBC to securely validate the reputation data based on corresponding IoT device data, and quickly identify malicious tampering of the DBC. The locations of the fog layer nodes are modeled according to a random Poisson point process (PPP) over a given 2-D area to approximate the stochastic property of real-world wireless node deployments. Furthermore, we assume that the number of IoT devices transmitting to the fog nodes also follows a random Poisson distribution. Based on these models, we derive novel closed-form expressions for the storage size, transmission latency, and tampering time of the IoT fog nodes in our DBC architecture. The main contributions of this research problem are summarized as follow:

- We propose a DBC architecture and analyze the performance of the private DBC with an IBC storing large amounts of IoT device data in a cloud layer due to limited storage space in the fog layer, whilst an RBC stores lightweight IoT reputation data in the fog layer nodes to easily assess the trust levels of the IoT devices. A blockchain mapping algorithm between the RBC and IBC is proposed to prevent tampering attacks in the fog and cloud blockchain layers.

- We develop a DBC node classification algorithm to classify the fog nodes into full, light, and basic nodes according to their storage space and computing resource to improve the performance of the information and reputation blocks generation and RBC storage compared to a single blockchain scheme where all fog nodes compete to generate and store the blockchain.

- We consider a general wireless IoT communication network deployment, in which the locations of fog nodes and the number of transmitting IoT devices are randomly distributed according to stochastic Poisson distributions. Then, we theoretically analyze the network performance by deriving closed-form expressions for the IBC and RBC storage size, processing and transmission latency, and tampering time-based on our extended model. These new analytical expressions provide a novel framework to examine the impact of varying fog nodes and IoT device densities in IoT blockchains.

- We analyze the performance of a DBC node classification algorithm and a blockchain-based mapping algorithm between the RBC and IBC to minimize the storage requirements and processing latency of our DBC compared to a conventional single blockchain (SBC) scheme. The fog nodes are classified to perform different tasks and each information and reputation block pair follows a one-to-one mapping to improve the scalability, latency, and security performances.

1.3.2 Secure Deep Reinforcement Learning

In Chapter 4, we propose a blockchain-secured deep reinforcement learning (BC-DRL) optimization framework for data management and resource allocation in decentralized wireless mobile edge computing (MEC) networks. The results in this chapter have been accepted to appear in the IEEE Transactions on Communications and submitted to the 2024 IEEE ICC conference.

In our framework, we design a low-latency reputation-based proof-of-stake (RPoS) consensus protocol to select highly reliable blockchain-enabled BSs to securely store MEC user requests and prevent data tampering attacks. We formulate the MEC resource allocation optimization as a constrained Markov decision process that balances minimum processing latency and denial-of-service (DoS) probability. We use the MEC aggregated features as the DRL input to significantly reduce the high-dimensionality input of the remaining service processing time for individual MEC requests. Our designed constrained DRL effectively attains the optimal resource allocations that are adapted to the dynamic DoS requirements. We provide extensive simulation results and analysis to validate that our BC-DRL framework achieves higher security, reliability, and resource utilization efficiency than benchmark blockchain consensus protocols and MEC resource allocation algorithms. The main contributions of this research problem are summarized as follow:

- We propose a reputation-based proof-of-stake (RPoS) blockchain consensus protocol that significantly reduces block generation and validation time whilst maintaining a high level of security by randomly selecting the miner BS node from a subset of BSs with high reputations. Attacks from malicious BSs are prevented by isolating the BSs with low reputations, whilst attacks from malicious users are resisted by using Bayesian inference to evaluate all user feedback. The secure storage of users' MEC requests in blockchain-enabled BSs further mitigates tampering attacks targeting MEC service provisioning.
- We solve the dynamic resource allocation by formulating it as an MDP that minimizes processing latency while satisfying the constraint on DoS probability. This formulation optimizes the allocation of computation resources for both blockchain consensus and MEC service provisioning. We provide mathematical proofs demonstrating the equivalence of the original problem and the reformulated MDP problem. In addition, we establish that the reformulated MDP satisfies the Markovian property.
- We design a constrained DRL algorithm that can accommodate dynamic requirements on DoS probabilities. To improve the training efficiency, we propose

an aggregating mechanism to reduce the dimension of the features as the remaining processing time of all the requests. Transfer learning is utilized to update pre-trained parameters when the requirement on DoS probability changes, and empirical convergence analysis is provided.

1.3.3 Scalable and Transferable Graph Neural Network

In Chapter 5, we develop a deep learning-based bandwidth allocation policy that is: 1) scalable with the number of users and 2) transferable to different communication scenarios, such as non-stationary wireless channels, different quality-of-service (QoS) requirements, and dynamically available resources. The results in this chapter have been submitted to the IEEE Transactions on Wireless Communications and published in the 2023 IEEE ICC Workshops.

To support scalability, the bandwidth allocation policy is represented by a graph neural network (GNN), with which the number of training parameters does not change with the number of users. To enable the generalization of the GNN, we develop a hybrid-task meta-learning (HML) algorithm that trains the initial parameters of the GNN with different communication scenarios during meta-training. Next, during meta-testing, a few samples are used to fine-tune the GNN with unseen communication scenarios. Simulation results demonstrate that our HML approach can improve the initial performance by 8.79%, and sampling efficiency by 73%, compared with existing benchmarks. After fine-tuning, our near-optimal GNN-based policy can achieve close to the same reward with much lower inference complexity compared to the optimal policy obtained using iterative optimization. The main contributions of this research problem are summarized as follow:

- Our proposed GNN is designed to handle six diverse QoS requirements of data rate, latency, and security in each of the long and short coding blocklength regimes. This generalization is achieved by using feature engineering to translate the channel state information (CSI) and customized QoS requirement of individual users into the minimum required bandwidth.

- Based on the extracted feature of minimum required bandwidth, we design a GNN-based bandwidth allocation policy that is scalable to the number of users. To train the GNN, we apply an unsupervised learning method to maximize the sum reward of the users with different QoS requirements in a network-slicing architecture.

- The optimal bandwidth allocation policies are obtained based on an iterative optimization algorithm to obtain the performance limit of the GNN-based policy in terms of the sum reward. By analyzing the computational complexity, we show that the GNN has a much lower inference complexity compared with the iterative optimization algorithm that is optimal.

- Finally, we develop our generalized hybrid-task meta-learning (HML) algorithm that is transferable to different communication scenarios by using meta-training to train the initial parameters of the GNN. We note that only a few samples are required to fine-tune the parameters of the GNN in meta-testing which validates that our GNN-based policy initialized by HML can be efficiently transferred to previously unseen communication scenarios. Simulation results show that our GNN-based policy achieves near-optimal performance and HML significantly outperforms the three considered benchmarks of MAML, MTL transfer (multi-task learning based transfer learning), and random initialization.

1.4 List of Publications

The following is a list of publications in refereed journals and conference proceedings during my Ph.D. candidature.

Journal Articles

1. X. Hao, P. L. Yeoh, Z. Ji, Y. Yu, B. Vucetic, and Y. Li, “Stochastic analysis of double blockchain architecture in IoT communication networks,” *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9700–9711, Jun. 2022.

2. X. Hao, P. L. Yeoh, C. She, B. Vucetic, and Y. Li, “Secure deep reinforcement learning for dynamic resource allocation in wireless MEC networks”, early access in *IEEE Transactions on Communications.*

3. X. Hao, C. She, P. L. Yeoh, Y. Liu, B. Vucetic, and Y. Li, “Hybrid-task meta-learning: A graph neural network approach for scalable and transferable bandwidth allocation”, submitted to *IEEE Transactions on Wireless Communications.*

4. W. Hu, Y. Yu, X. Hao, P. L. Yeoh, L. Guo, and Y. Li, “A cost-effective multi-type data scheduling for blockchain in massive Internet of UAVs,” early access in *IEEE Internet of Things Journal.*

Conference Proceedings

1. X. Hao, P. L. Yeoh, T. Wu, Y. Yu, Y. Li, and B. Vucetic, "Scalable double blockchain architecture for IoT information and reputation management," in *Proceedings of 2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, New Orleans, LA, USA, pp. 171–176, 2021.

2. X. Hao, P. L. Yeoh, Y. Liu, C. She, B. Vucetic, and Y. Li, "Graph neural network-based bandwidth allocation for secure wireless communications," in *Proceedings of 2023 IEEE International Conference on Communications Workshops (ICC workshops)*, Rome, Italy, pp. 332–337, 2023.

3. X. Hao, P. L. Yeoh, C. She, Y. Yu, B. Vucetic, and Y. Li, "A constrained deep reinforcement learning optimization for reliable network slicing in a blockchain-secured low-latency wireless network," *Accepted by 2024 IEEE International Conference on Communications (ICC).*

Chapter 2

Preliminaries of Related Technologies

2.1 Blockchain

Blockchain consensus protocol design is the key technology that enables the tamper-proofing property of blockchain networks. The most well-known blockchain consensus protocol is the proof-of-work (PoW), which is used in `Bitcoin`. However, it has been shown in [Ling *et al.*, 2021; Xiong *et al.*, 2020] that a blockchain network with PoW consensus suffers from lengthy delays due to the competition amongst all blockchain nodes during the block generation process and the large amount of computations required for block validation by all nodes. To reduce the high computation overhead and processing latency for block generation, the practical Byzantine fault tolerance (PBFT) consensus was proposed to select one blockchain node for new block generation [Castro and Liskov, 1999]. However, PBFT consensus protocol still experiences high computation resource requirements and latency for block validation, because all the nodes need to validate the newly generated block. The proof-of-stake (PoS) consensus protocol is a promising solution to significantly reduce the computation overhead in blockchain networks by selecting a trusted subset of blockchain nodes to validate the newly generated block [Tschorsch and Scheuermann, 2016]. Nonetheless, the PoS consensus is more vulnerable to attacks compared with PoW and PBFT, since the highest stake miner node that is selected for block generation can be easily identified and targeted by attackers [Yang *et al.*, 2019].

In the hierarchical IoT and the decentralized MEC, blockchain technology enhances data security and trust by providing a tamper-proof and transparent record of transactions. It enables decentralized data exchanges among IoT devices and edge servers, reducing reliance on centralized intermediaries and mitigating risks of unauthorized access. By integrating blockchain for consensus mechanisms, hierarchical IoT and MEC systems ensure the integrity and confidentiality of data processing at the network edge. This approach fosters innovation and autonomy in data interactions while strengthening the resilience and reliability of wireless low-latency communications.

2.2 Unsupervised Learning

Supervised learning and unsupervised learning are the two most commonly used machine learning categories. Specifically, supervised learning is trained on labeled data of the optimal policy, whilst unsupervised learning focuses on finding patterns in the dataset without explicit labels or target outputs. When solving complicated problems, unsupervised learning has the following advantages compared with supervised learning [She *et al.*, 2021]: 1) Saving the computing power for collecting the labels; 2) Eliminating the system error caused by the non-linearity in labels; 3) Solving non-deterministic polynomial-time hard (NP-hard) problems without labeled training data.

The target of unsupervised learning algorithms is to find the optimal policy by finding a minimum loss value, and is given by

$$\theta^* = \arg\min f(\theta), \tag{2.2.1}$$

where θ^* represents the parameters of the neural network's optimal policy, and $f(\theta)$ is the loss function. Stochastic gradient descent (SGD) is a commonly used method to train in unsupervised learning algorithms, and is described as

$$\theta_{n+1} = \theta_n + \beta_\theta \nabla f(\theta_n), \tag{2.2.2}$$

where θ_n is the parameters of the neural network in the n-th iteration, and $\nabla f(\theta_n)$ is called the policy gradient in the n-th iteration.

Unsupervised learning has been widely applied to low-latency wireless communications. In [Sun and Yang, 2019b], the authors applied unsupervised learning to jointly optimize the transmit power and bandwidth for URLLC services. In [Filho *et al.*, 2021], the authors detected cyber-attacks on the cyber–physical systems with strict latency requirements by a promising unsupervised approach, generative adversarial networks. In [Jing *et al.*, 2020], the authors minimized the weighted sum of all the IoT users by the proposed joint unsupervised learning and DRL algorithm.

When applying unsupervised learning algorithms in wireless networks, it is essential to transform the variable optimization problem into a functional optimization problem. This involves defining an optimization function, where the "variable" itself is a function [Sun *et al.*, 2023]. For example, when we use the channel state information (CSI) to maximize the data rate by optimizing the bandwidth allocation, both the bandwidth allocation and the data rate are functions of the CSI. By transforming the variable optimization problem into a functional problem, the dataset is

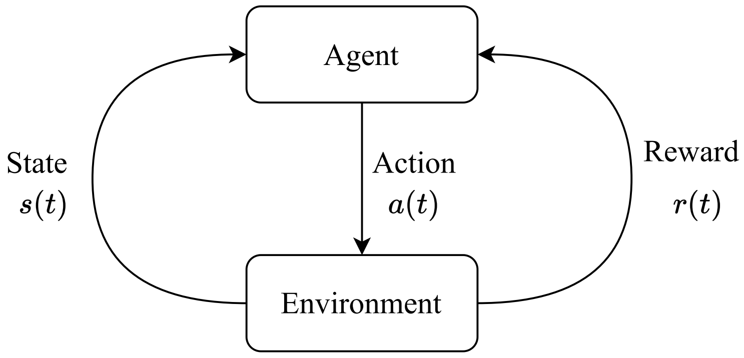


Figure 2.1: Agent-environment interaction loop for RL algorithms.

pre-structured before being fed into the neural network, thus simplifying the training process afterward.

The integration of unsupervised learning techniques presents a promising approach for bandwidth allocation in the Network Slicing (NS) scenario. Unsupervised learning algorithms offer the capability to extract valuable insights from unlabeled data without the need for explicit guidance or supervision. This is particularly advantageous in the NS context, where collecting labeled data can be challenging due to the diverse QoS requirements across different slices.

2.3 Deep Reinforcement Learning

In addition to supervised and unsupervised learning algorithms, reinforcement learning (RL) is designed for solving sequential decision-making problems, which takes into account not only the immediate consequences of each decision but also their long-term impact on achieving a specific goal. [Shakarami *et al.*, 2020]. As shown in Fig. 2.1, a classical RL contains two main characters, namely agent and environment. The agent is a decision-maker, which learns to act by trial and error in the unknown environment [Liang *et al.*, 2018]. From Fig. 2.1, we can observe the interaction of the agent and the environment in the t-th time slot: the agent observes the state of the environment, $s(t)$, then takes the action, $a(t)$, lastly perceives a reward, $r(t)$. The state, action, and reward are the three main elements in DRL, specifically,

1) *State*: is a representation of the current situation of the environment that the agent operates in. The state provides essential information that the agent uses to make decisions aimed at achieving a specific goal. In wireless communications, the commonly seen DRL states include CSI, available radio resources, and the amount of data traffic arrived.

2) *Action*: refers to the decision made by the agent, who observes the state to optimize the goal of a long-term objective. In wireless communications, the commonly seen DRL actions include communication and computing resource allocation policy, user scheduling scheme, and size of transmitted packets.

3) *Reward*: is a numerical signal provided to the agent by the environment after it takes a specific action in a given state. The target of DRL is to maximize the long-term reward, $R(t)$, given by

$$R(t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(t)\right], \tag{2.3.1}$$

where $\gamma \in (0, 1)$ is the discount factor. In wireless communications, the commonly seen DRL rewards include latency, data rate, and security-related metrics.

If we aim to use RL to solve a problem, the problem has to be a Markov decision process (MDP). This condition is met when the environment's state and reward at $t + 1$ depends only on the state and action representations at t, in which case the environment's dynamics can be defined by [Sutton and Barto, 2016],

$$p(s', r|s, a) = \Pr\{s(t + 1), r(t + 1)|s(t) = s, a(t) = a\}, \tag{2.3.2}$$

for all r, s', s, and a. When applying DRL to solve the MDP problem, the long-term reward, $R(t)$ given in eq. (2.3.1) could be approximated by the state-action function denoted by $Q(s(t), a(t))$ [Feriani and Hossain, 2021]. The optimal state-action function, $Q^*(s(t), a(t))$, is computed by solving the Bellman equation given by

$$Q^*(s(t), a(t)) = y(t) = r(t) + \gamma Q(s(t+1), a(t+1)). \tag{2.3.3}$$

We note that since the focus is on optimizing the long-term reward, it is possible that sometimes immediate action will not lead to an immediate increase in the instantaneous reward. This can occur because there are situations where sacrificing the instantaneous reward could potentially result in an enhancement of the long-term reward.

Classical RL algorithms traditionally operate in environments with low-dimensional state and action spaces. These algorithms rely on tabular representations of the environment's state-action space and are well-suited for solving problems with relatively small state and action spaces. However, in real-world applications such as wireless communications, the state and action spaces are often high-dimensional and continuous [She *et al.*, 2019], making traditional RL methods impractical.

Deep RL (DRL) addresses this limitation by leveraging deep neural networks to approximate complex value functions or policies directly from high-dimensional observations. Unlike classical RL, which relies on discrete state-action pairs, DRL algorithms can operate directly on raw sensory inputs, enabling them to handle high-dimensional state and action spaces efficiently [Dong *et al.*, 2019]. Additionally, DRL algorithms excel at learning hierarchical representations of the environment, allowing them to discover intricate patterns and dependencies within the data [Meng *et al.*, 2023].

DRL is suitable for the considered low-latency MEC scenario due to its inherent adaptability to process the sequential decision-making problems and ability to handle high-dimensional input spaces, which are essential for optimizing computing resource allocation in MEC-enabled low-latency wireless networks. In addition, DRL's capability to learn hierarchical representations and extract insights from raw sensory inputs further enhances its suitability for addressing the complex challenges associated with MEC computing resource allocation.

2.4 Constrained Deep Reinforcement Learning

Constrained DRL has been developed to transform the optimization problem with specific constraints to have an objective function with a weighted sum of the original objective function and constraints, where weights are optimized Lagrange multipliers [Liang *et al.*, 2018]. Specifically, constrained DRL aims to select a policy, $\mu(\cdot)$, that maximizes the long-term reward, $R(t)$, while satisfying the constraints on the long-term cost, $C(t)$. The optimization problem can be formulated as [Liang *et al.*, 2018]

$$\begin{aligned} \mu^* = \arg\max_{\mu(\cdot)} \quad & R_\mu(t) \\ \text{s.t.} \quad & C_\mu(t) \leq C_{\max}, \end{aligned} \tag{2.4.1}$$

where μ^* represents the optimal policy, and $C_{\max}$ is the threshold of the long-term cost. To solve problem (2.4.1), the Lagrangian relaxation procedure is employed in constrained DRL, and the Lagrangian function is given by

$$\mathcal{L}_t\left(\mu(\cdot), \lambda_{\mathcal{L}}\right) = R_\mu(t) - \lambda_{\mathcal{L}}\left(C_\mu(t) - C_{\max}\right), \tag{2.4.2}$$

where $\lambda_{\mathcal{L}}$ is the Lagrangian multiplier. We note that it is possible to extend the optimization problem from having a single constraint to considering multiple constraints by introducing additional Lagrangian multipliers for each new constraint.

Constrained DRL ensures computing resource allocation policies adhere to specific DoS probability constraints, offering flexibility for tailoring solutions to specific DoS requirements. This approach is particularly valuable in real-world MEC deployments. Therefore, constrained DRL enables the MEC system to strike a balance between minimizing the latency and satisfying DoS constraints considered in this thesis.

2.5 Graph Neural Network

Selecting a suitable neural network structure for solving the target problem is as important as choosing a proper machine learning algorithm, because the selection of the neural network structure greatly affects the training process of a machine learning algorithm. The neural networks include different kinds of structures, such as the fully-connected neural network (FNN), the convolutional neural network, the recurrent neural network, and the graph neural network (GNN). The commonly seen dynamic number of users leads to the dynamics of network connectivity. GNN demonstrates high efficiency in processing data with dynamic connectivity by utilizing graph representation of the data [Wu *et al.*, 2021b].

To apply GNN for training, the data must be transformed into a dedicated graph representation that adheres to the specified graph definitions. An example of a graph is shown in Fig. 2.2. This graph G consists of a set of vertices V_{G} and a set of edges E_{G}, where each edge connects a pair of vertices, representing the connections or relationships between different elements. Mathematically, this can be expressed as $G = (V_{\mathrm{G}}, E_{\mathrm{G}})$. We assume the graph has K vertices, where $V_{\mathrm{G}} = \{v_k; k \in \mathcal{K}\}$ is the set of vertices, and $E_{\mathrm{G}} = \{e_{k_1,k_2}; k_1, k_2 \in \mathcal{K}\}$ is the set of edges. Vertex v_2 is the

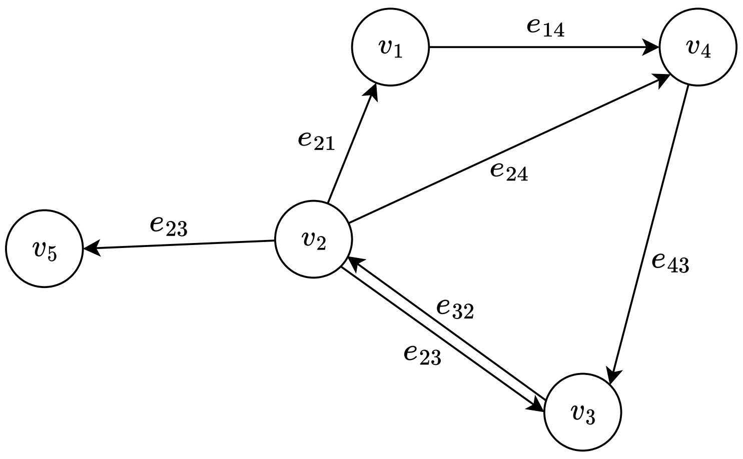


Figure 2.2: An example of a graph.

neighbor of all the other vertices.

GNNs compose diverse classes, including the message passing neural network (MPNN) [Gilmer *et al.*, 2017], the random edge graph neural network [Eisen *et al.*, 2019], and the graph attention network [Veličković *et al.*, 2018]. The effective performance of the GNN depends on the suitable choice of the GNN class for graph data representation. MPNN is particularly well-suited for wireless networks with a dynamic number of users due to its permutation equivariance property, which indicates MPNNs can update node representations based on changing interactions [Shen *et al.*, 2021]. MPNN has been widely applied to wireless communication networks, where the commonly seen vertices are BSs, users, wireless links, and transceiver pairs. In [Shen *et al.*, 2021], the authors solved large-scale radio resource management problems, where high computational efficiency is achieved by processing the aggregated and combined features of all the transceiver pairs instead of processing them one by one. In [Hao *et al.*, 2023], the authors achieved a scalable bandwidth allocation policy for an uplink non-stationary wireless communication system with a dynamic number of users by MPNN. In [Gu *et al.*, 2023], the authors optimized distributed power allocation by MPNN, where the transmit power of pilots represents messages from neighboring vertices and can be aggregated efficiently by evaluating the total interference power.

The three main steps of MPNNs are message passing, aggregation, and readout. These three steps are important for applying MPNN into wireless networks with a dynamic number of users, specifically,

1) Message Passing: MPNNs employ a message-passing mechanism that allows vertices to exchange information with their neighbors iteratively. Through passing the messages of the vertices inside the graph, the vertices can explore the structure of the wireless network. For example, in Fig. 2.2, the messages carried by vertex v_2 are passed to all the other vertices with one iteration of message passing. In wireless networks with a dynamic number of users, the message-passing mechanism is adaptable to the dynamics of the user numbers.

2) Aggregation: Aggregating information from neighbor vertices accommodates the dynamic changes in user number. The aggregated messages can be processed together, thus saving the energy of processing the diverse data.

3) Readout: The readout mechanism summarizes aggregated information to produce the graph-level representation. This is particularly valuable in wireless networks with dynamic user numbers as it facilitates informed decision-making by observing the network state evolution.

In the considered NS scenario, GNNs are utilized to adapt to the dynamic number of users efficiently. Unlike conventional FNNs, the GNN can offer scalability, enabling them to handle fluctuations in the number of users effectively. This scalability ensures robust bandwidth allocation strategies, accommodating the varying demands of users in low-latency wireless communication conditions.

2.6 Meta-Learning

Meta-learning, also known as learning to learn, can encourage the GNN to learn transferable graph representations that generalize well across different channel distributions. The authors of [Finn *et al.*, 2017] proposed a classic meta-learning algorithm, model-agnostic meta-learning (MAML), which can improve the adaptation ability of a neural network to new scenarios. MAML contains two main stages: meta-training and meta-testing, specifically,

1) Meta-Training: In this initial stage, a neural network learns from a range of datasets to acquire generalizable knowledge, enabling it to quickly adapt to new tasks during meta-testing.

2) Meta-Testing: In the meta-testing stage, the neural network's performance is evaluated on unseen datasets not used in meta-training. Using the knowledge it gained from meta-training, the neural network adjusts and performs well on new datasets, demonstrating its capacity to handle unfamiliar tasks.

Meta-learning has been applied to achieve timely resource allocation in non-stationary wireless networks. In [Huang *et al.*, 2021a], the authors utilized meta-learning to minimize the MEC latency by optimizing the computing resource allocation in a wireless network serving different combinations of wireless devices. In [Wang *et al.*, 2022b], the authors used meta-learning to quickly adapt to the user movement patterns in a VR network. In [Li *et al.*, 2021b], the authors proposed a meta-learning-based Wi-Fi impersonation detection algorithm, which learns knowledge from historical scenarios and quickly adapts to new scenarios with few-shot samples. In [Hu *et al.*, 2020], the authors designed a meta-learning algorithm that can help the drone base station to quickly adjust its trajectory to adjust to dynamic user requests.

Meta-learning is utilized in the considered NS scenario to enable rapid adaptation to changing network conditions and user requirements. By leveraging prior knowledge from related tasks, meta-learning enhances the agility and efficiency of resource allocation algorithms, allowing the system to effectively allocate bandwidth resources and meet diverse QoS requirements in a timely manner.

Chapter 3

Double Blockchain Architecture for IoT Communication Networks

In this chapter, we propose a DBC architecture for secure information and reputation data management in large-scale wireless IoT networks. Specifically, the DBC is a private blockchain deployed on a cloud-fog communication network which is composed of an information blockchain (IBC) storing large amounts of IoT data in the cloud layer and a reputation blockchain (RBC) storing reputation data of the IoT devices in the near-terminal fog layer. The locations of the fog layer nodes are modeled according to a random Poisson point process (PPP) over a given two-dimensional area to approximate the stochastic property of the considered stochastic wireless node deployment model. Furthermore, we assume that the number of IoT devices transmitting to the fog nodes also follows a random Poisson distribution. Based on these models, we derive novel closed-form expressions for the storage size, transmission latency, and tampering time of the IoT fog nodes in our DBC architecture. Numerical simulations highlight high storage scalability, low latency, and superior security of the DBC design, and provide insights into the performance gains for different fog node and IoT device densities.

3.1 Chapter Introduction

Blockchain technology has received significant attention in the last decade due to its benefits of data transparency, robustness, and security [Hao *et al.*, 2021]. More recently, blockchains have been considered for Internet-of-things (IoT) applications where huge amounts of data from IoT devices are securely hashed and stored in communication networks [Li *et al.*, 2020]. However, conventional blockchain architectures face the challenges of enormous storage space, high computation complexity, and extended system latency required to manage the entire blockchain, which cannot be reliably met by miniature IoT devices with inadequate storage resource, computing capability, and transmit power [Liu *et al.*, 2019b; Yao *et al.*, 2019]. As such, novel lightweight blockchain and communication architectures are required to improve the scalability, latency performance, and security of future wireless communication IoT blockchain applications.

Realistic wireless network models based on stochastic geometry have been previously considered to accurately characterize the massive randomly distributed geographic locations of fog nodes and the amount of IoT device data [Cui *et al.*, 2016; Haenggi, 2005; Haenggi, 2013]. The Poisson point process (PPP) is a flexible stochastic model for general wireless node distributions in a two dimensional plane. The PPP model is commonly used to characterize small shifts in the signal-to-interference-plus-noise-ratio (SINR) at each node [Shojaeifard *et al.*, 2015] and derive the expected throughput by integrating the SINR over the total network area [Guo and M. Haenggi, 2015; Gupta *et al.*, 2015]. In real-world applications, the amount of IoT data transmitted is also randomly determined by the IoT devices' individual schedules which can be modeled according to a Poisson distribution [Sun *et al.*, 2019a].

Given the large-scale nature of IoT networks, it is important to take scalability into consideration. In [Li *et al.*, 2017], the blockchain is divided into shorter segments known as shards to improve scalability, where a large amount of IoT data can be processed in multiple parallel shards. In [Huang *et al.*, 2021a], RepChain was proposed as a sharding-based double blockchain scheme that is executed in a single layer communication network. However, parallel processing with shards requires all nodes to compete and participate in blockchain generation and storage, which is challenging in IoT applications with varying number of nodes and communications resource constraints. Furthermore, single-layer IoT network structures have limited scalability in terms of storage space requirement compared to cloud-fog networks, where blockchain data could be stored in different layers based on the dynamic size and latency requirements of blockchain nodes.

Communications latency is a critical performance bottleneck for exchanging data in IoT blockchains [Lei *et al.*, 2020]. In a wireless IoT blockchain network, the time cost to generate and propagate a block, namely processing and transmission latency, are the key concerns because of the restricted computing resource and numerous nodes required to handle the blockchain. In [Wu *et al.*, 2021a], the processing latency is reduced by offloading computing tasks to edge or cloud processing nodes in an edge-cloud communications network. In [Yu *et al.*, 2021], the transmission latency is improved with a tree-based clustering algorithm, by which the blocks are propagated through different node clusters with a compressed tree depth. We note that these previous works assumed fixed distributions for the node locations. As such, the impact of randomly located nodes on cloud-fog IoT blockchain processing and transmission latency has not been considered in the open literature.

One of the key concerns of implementing blockchains in IoT applications is to ensure strong data integrity and manage trust amongst the distributed storage and processing nodes [Malik *et al.*, 2019]. To this end, private blockchain mechanisms [Ding *et al.*, 2020; Massar *et al.*, 2020] have been proposed for trusted IoT applications, in which node authentication is strongly enforced and their activity is actively monitored based on reputation evaluations. In [Li *et al.*, 2019a], the authors proposed to evaluate the reputation of each router in an IoT network based on router reports stored in an information blockchain to identify unreliable malicious nodes. The authors of [Kang *et al.*, 2019b] considered a reputation management scheme to ensure secure miner selection and prevent blockchain tampering. The reputations were evaluated based on past interactions with the miner candidates and recommendation data from other nodes. We note that mapping the relationship between the reputations and the information used for their evaluation could offer important benefits of traceability and improved security in IoT blockchains for trust management.

In this paper, to address the above challenges, we present detailed stochastic modeling and performance analysis of our double blockchain (DBC) architecture from [Hao *et al.*, 2021] which is composed of an information blockchain (IBC) and a reputation blockchain (RBC) that are processed and stored in a cloud-fog communications network. Each reputation block (RB) contains reputation values of the IoT devices that are evaluated based on their historical reputation stored in the previous RB and their current information stored in a corresponding mapped information block (IB). Our former work in [Hao *et al.*, 2021] proposed a DBC architecture assuming known fixed locations of fog nodes and a constant amount of IoT data. In this paper, to provide deeper insights into the scalability performance of the IoT blockchain for different node densities, we consider a more general network model with random fog node locations and random numbers of IoT devices uploading data, and derive new closed-form expressions based on random Poisson distributions to analyse the scalability, latency, and security performance of the IoT blockchain architecture. Furthermore, we provide a comprehensive analysis of the IoT blockchain system management in terms of the consensus protocol, smart contracts, potential attacks, and influences of varying node densities.

Numerical simulations are provided to assess the storage efficiency, processing latency, transmission latency, and blockchain tampering security of the extended DBC architecture. We highlight that the DBC architecture is well-suited to advanced IoT blockchain applications with large-scale network size and low-latency requirements.

3.2 System Model

This section describes the DBC model from [Hao *et al.*, 2021], and presents the communication model using a random node deployment for the fog blockchain layer networks and the number of transmitting IoT devices. Finally, the DBC consensus protocol is discussed.

3.2.1 DBC Architecture Model

The system model shown in Fig. 3.1 comprises the cloud blockchain layer, the fog blockchain layer, and the IoT device layer. The DBC architecture consists of two mapped blockchains, namely the IBC stored in the cloud blockchain layer and the RBC stored in the fog blockchain layer. Raw data collected by the IoT devices is

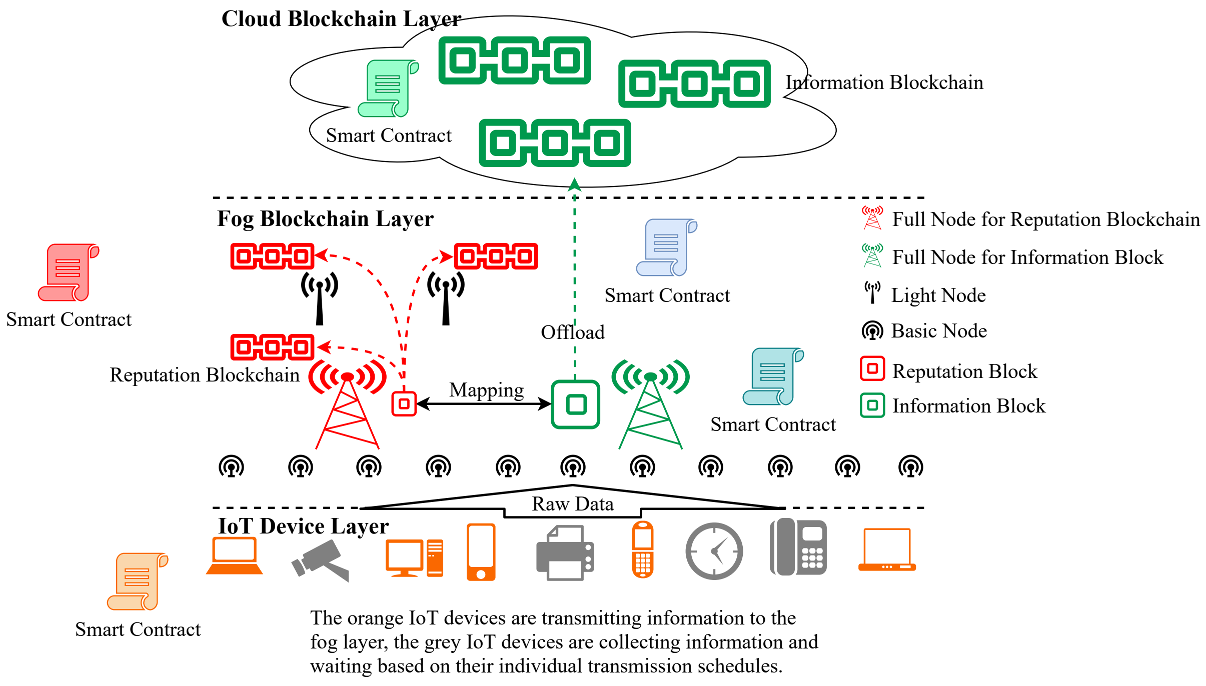


Figure 3.1: Scalable double blockchain architecture for IoT information and reputation management.

processed at the fog nodes to generate new IBs and RBs updated in the IBC and RBC, respectively. We consider that the short RBs are broadcasted in the near-terminal fog layer, while the long IBs are offloaded to the cloud layer.

The DBC node classification algorithm identifies suitable fog nodes that satisfy different resource requirements for the RBC and IBC. Our fog layer node classification algorithm identifies three classes for RBC and two classes for IBC. Specifically, we define full nodes, light nodes, and basic nodes for the RB, and full nodes and basic nodes for the IBC. For RBC, the full nodes are fog nodes with sufficient storage space and computing capacity to store and generate new reputation blocks. The light nodes have adequate storage space for storing the RBC but have limited computing capabilities to generate new reputation blocks. The basic nodes have limited resources and cannot participate in RBC storage or generation but could transfer data collected from IoT devices. For IBC, the full nodes are fog nodes with sufficient computing capacity to generate new information blocks but have limited storage space, whilst the basic nodes are the same as for the RBC. We do not classify any light nodes for the IBC because all new IBs are offloaded to the cloud layer for storage.

A smart contract is a self-executed software in the blockchain system by user-defined script, which can be used to ensure trusted information exchange in the blockchain without the need of a central server [Debe *et al.*, 2019]. The DBC approach [Hao *et al.*, 2021] is composed of a private IBC and a private RBC, which can be implemented using smart contracts for cloud blockchain storage management in the cloud layer and trust management of the fog nodes and IoT devices [Wang *et al.*, 2019] in the fog layer. In the DBC approach [Hao *et al.*, 2021], the fog nodes authentication is strongly enforced by the smart contract because all the fog nodes in the system are required to register via the smart contract. The behaviors of fog nodes and IoT devices in the network will be monitored by the smart contract, which can blacklist a malicious fog node or an IoT device if it tries to tamper the blockchain, or flag as suspicious based on its communication behavior.

Querying the reputation value in the RBC can be performed quickly due to their short block lengths. The one-to-one mapping algorithm [Hao *et al.*, 2021] between the RBC and IBC ensures that the smart contract can accurately verify the reputation values in the RBC by querying their corresponding information in the IBC. Based on the IBC and RBC values, the IoT devices can be carefully managed to prevent data security attacks and ensure that a high-level of trust is maintained in the IoT application.

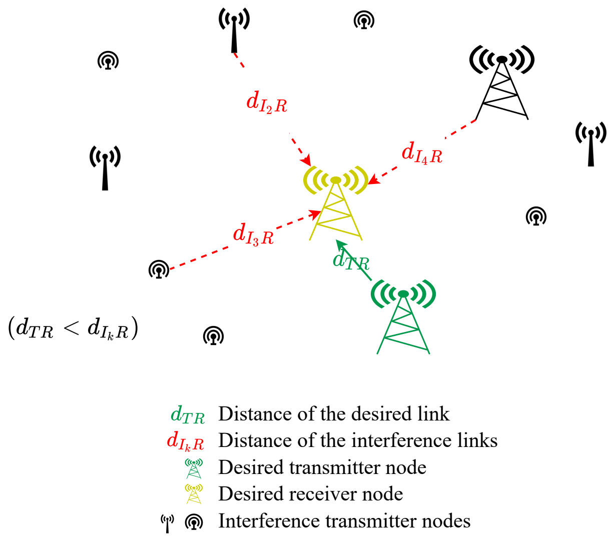

d_{I_2R}
d_{I_4R}
d_{I_3R}
d_{TR}
$(d_{TR} < d_{I_kR})$
d_{TR} Distance of the desired link
d_{I_kR} Distance of the interference links
Desired transmitter node
Desired receiver node
Interference transmitter nodes

Figure 3.2: Signal and interference transmission in the target area. The receiver node receives the desired signal from the nearest full or light node with distance d_{TR} and suffers interference from the other nodes with distance $d_{I_kR}(k > 1)$ in the considered area.

3.2.2 Stochastic Communications and Interference Model

In this section, we present the communication model used to transmit the RBs in the fog blockchain layer. The full, light and basic nodes for RBC are assumed to be independently geographically distributed according to a PPP, where the corresponding node distribution intensities [1] per unit area are λ_{fr}, $\lambda_{\ell r}$ and λ_{br}. Since these three kinds of fog nodes are independent, we could conclude that the distribution of the entire fog node set $\mathcal{N}$ is also a PPP with intensity $\lambda = \lambda_{fr} + \lambda_{\ell r} + \lambda_{br}$. When a new block is propagated through the network, the intensity of the transmitting nodes is $\lambda_t = \lambda_{fr} + \lambda_{\ell r}$ since only the full and light nodes participate in the block propagation process. During this time, we consider that there exists interference from the other

[1]The deterministic counterpart of the random counting measure in a Poisson distribution is defined as the intensity measure, which is the expected number of points in the target set [Haenggi, 2013]. In a communications network, the node intensity is also interchangeably referred to as the node density [Gupta *et al.*, 2015].

nodes with intensity λ_{fog}.

As shown in Fig. 2, we assume that the nearest node from the transmitter is the desired receiver, and the signals sent from other users are treated as interference [Haenggi, 2013]. In the figure, the desired signal is transmitted with distance d_{TR}, while the interference signals from the other nodes in the considered area are transmitted with distance $d_{I_kR} > d_{TR}$. The distance between the fog nodes in a PPP network is characterized by the node intensity and probability distribution function (PDF) of the transmission distance. The corresponding PDF of the kth nearest transmitter node to the receiver node can be expressed as [Ji *et al.*, 2021]

$$f_{\mathrm{PDF}}(d_k) = \frac{2\left(\lambda\pi d_k^2\right)^k}{d_k\Gamma(k)} e^{-\lambda\pi d_k^2}, \tag{3.2.1}$$

where, d_k is the distance from the receiver node to the kth nearest transmitter node, λ_{fog} is the intensity of the considered fog node set, and $\Gamma(\cdot)$ is the gamma function.

The number of fog nodes in the PPP network can be characterized according to a probability mass function (PMF) given by

$$f_{\mathrm{PMF}} = \frac{e^{-\lambda}\lambda^n}{n\Gamma(n)}, \tag{3.2.2}$$

where n is the number of fog nodes. We consider large-scale fading, where the signal power at the receiver of fog node n_i is a random variable with the distribution of [Andrews *et al.*, 2011; Haenggi, 2013]

$$P_{s,n_i} = P_u d_{TR,n_i}^{-\hat{\alpha}}, \tag{3.2.3}$$

where P_u denotes the transmit power of the transmitter node, d_{TR,n_i} is the distance of the link from the transmitter to the receiver of node n_i, and $\hat{\alpha}$ is the distance-dependent path loss. We define the SINR_{n_i} as the signal-to-interference-plus-noise ratio of fog node n_i, which is calculated as [Andrews *et al.*, 2011]

$$\mathrm{SINR}_{n_i} = \frac{P_{s,n_i}}{I_{n_i} + \sigma^2} = \frac{P_u d_{TR,n_i}^{-\hat{\alpha}}}{\sum\limits_{d_{I_k R} \geq d_1} P_u d_{I_k R,n_i}^{-\hat{\alpha}} + \sigma^2}, \tag{3.2.4}$$

where P_{s,n_i} is the desired signal power of node n_i, d_{TR,n_i} is the distance from the transmitter to the receiver node n_i, which is the distance of the desired link. $d_{I_k R,n_i}$ is the distance from the kth interference to the receiver node n_i, as shown in Fig. 3.2.

The IoT devices collect data that is transmitted to the IBC full nodes. We assume that different IoT devices collect and upload information according to their individual schedules. As such, the number of IoT devices providing information at any given time follow a Poisson distribution, which can be characterized by a PMF as

$$P_{iot}(m) = \frac{e^{-\lambda_{iot}} \lambda_{iot}^m}{m\Gamma(m)}, \tag{3.2.5}$$

where, m is the number of transmitting IoT devices, λ_{iot} is the intensity of IoT devices.

3.2.3 DBC Consensus Protocol

The Proof of Authority (PoA) [Alrubei *et al.*, 2021] consensus protocol is often considered for IoT networks with private blockchains [Alrubei *et al.*, 2020] due to its computationally-efficient approach for adding a new block to the blockchain, and low-latency requirements as it does not require time-consuming confirmation rounds and validation by authorized nodes. The private DBC is well-suited to adopt the PoA consensus due to its lightweight and fast processing requirements, where the newly generated blocks are authorized automatically by the smart contract. We assume that the fog nodes are authenticated prior to joining the DBC network and the smart contract assigns suitable nodes in the network to perform validation before a newly generated block is added to the IBC or RBC.

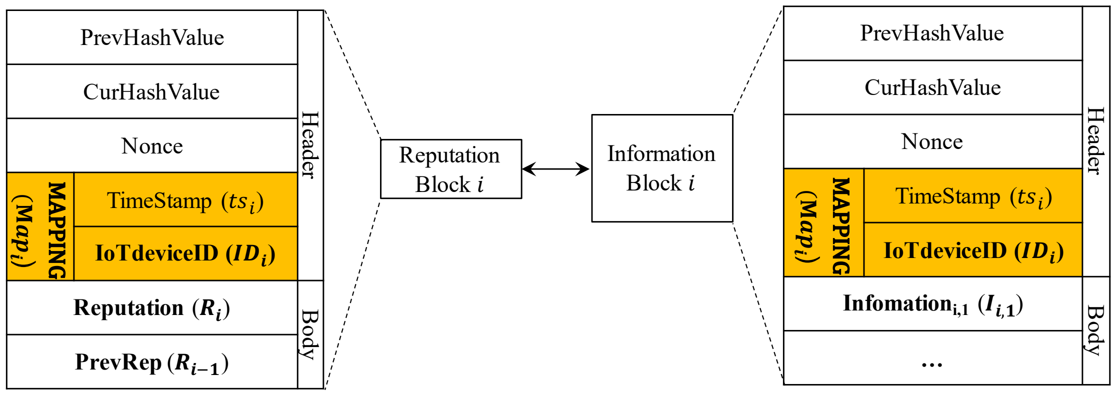


Figure 3.3: Block structure and mapping relationship.

3.3 DBC Architecture and Performance Analysis

In this section, we describe the DBC node classification, block structure, and mapping algorithm. Then, we design the storage size, the processing and transmission latency, and tampering time of the DBC architecture based on the PPP node deployment with the corresponding cost and performance tradeoff discussions. Since different applications may have different constraints for storage, latency and security, it is important to carefully consider the performance tradeoffs between the DBC and SBC architecture.

3.3.1 DBC Node Classification

For the DBC node classification algorithm [Hao *et al.*, 2021], let the set of fog nodes be $\mathcal{N} = \{n_1, n_2, ..., n_{N_{\text{fog}}}\}$, where N_{fog} is the total number of fog nodes. We denote S_i as the storage space and F_i as the computing resource of the ith fog node $n_i \in \mathcal{N}$. We design a node classification algorithm to determine the sets $\mathcal{N}_{fr}$, $\mathcal{N}_{\ell r}$ and $\mathcal{N}_{br}$, which represents the set of full, light and basic node set related to the RBC, while $\mathcal{N}_{fi}$ and $\mathcal{N}_{bi}$ represent the sets of full nodes and basic nodes related to the

Algorithm 1: Fog Layer DBC Node Classification

```
1  Initialize: $\mathcal{N}_{fi} = \mathcal{N}_{bi} = \mathcal{N}_{br} = \mathcal{N}_{fr} = \mathcal{N}_{\ell r} = \phi$, minimum space required for
     storing RBC $s_{\min_{RBC}}$, minimum computation required for generating new IB
     and RB $c_{\min_{IB}}$ and $c_{\min_{RB}}$;
2  for $n_i \in \mathcal{N}$ do
3      if $F_i > c_{\min_{IB}}$ then
4          $n_i \to \mathcal{N}_{fi}$;
5      else
6          $n_i \to \mathcal{N}_{bi}$;
7      end
8      if $S_i < s_{\min_{RBC}}$ then
9          $n_i \to \mathcal{N}_{br}$;
10     else if $F_i < c_{\min_{RB}}$ then
11         $n_i \to \mathcal{N}_{\ell r}$;
12     else
13         $n_i \to \mathcal{N}_{fr}$;
14     end
15 end
16 Output $\mathcal{N}_{fi}, \mathcal{N}_{bi}, \mathcal{N}_{br}, \mathcal{N}_{\ell r}, \mathcal{N}_{fr}$.
```

IB. N_{fr}, $N_{\ell r}$, N_{br}, N_{fi}, N_{bi} represent the corresponding node numbers, respectively. The algorithm first explores whether the target node is qualified to be a full node for generating an IB by comparing its computing resource with the designed threshold. Then, the same node will be classified according to the requirements of RBC. If the storage size of the target node is smaller than the threshold $s_{\min_{RBC}}$, it will be assigned to the basic fog node set for RBC $\mathcal{N}_{br}$. In contrast, if it has enough storage space, the subsequent step of the algorithm checks its computing resource to identify whether it is a full or light node for RBC.

We note that the computing resources required to generate the short RBs will be significantly lower than those for the long IBs, and we consider there are insufficient storage resources for the IBC in the fog layer. As such, the number of full nodes required for the RBC and IBC will not be the same. Furthermore, the algorithm classifies basic nodes, light nodes and full nodes for the RB, whilst only basic nodes and full nodes are considered for the IBC.

The DBC node classification algorithm determines the intensities of each fog node class and improves the scalability, latency, and security of the system by classifying the fog nodes based on the requirements of the IoT blockchain network.

3.3.2 DBC Block Structure and Mapping

In our DBC architecture, we consider a one-to-one mapping between the newly generated IB and RB to ensure that the reputation is constantly updated based on new IoT data. This is because a higher reputation updating frequency will result in faster identification of malicious behaviors by the IoT devices. Since we consider different IoT devices collect and transmit data to the fog layer according to their specific schedules, only the reputations of active devices will be updated. Thus, our one-to-one block mapping will result in different IoT device reputations stored in different RBs. The regular generation of new RBs will also result in a smaller block size.

Fig. 3.3 shows the block structure of the IB and RB which are composed of a header and a body. The header in both the ith IB and ith RB contains hash values of the $(i-1)$th and ith blocks, a nonce, a timestamp, a vector of ID values stored in $\boldsymbol{ID_i}$ representing the IoT devices that are transmitting information from the IoT device layer to the fog layer to generate the ith block pair, and a vector of mapping value stored in $\boldsymbol{Map_i}$ between the ith IB and ith RB. We note that the new RB is generated after the generation of its corresponding IB, because the RBC full node has to wait for the IBC full node to send the corresponding time stamp and device

ID data via high-speed backhaul links to ensure the RB is correctly mapped. Each block in the IBC stores information data from all the IoT devices whilst each block in the RBC stores the reputation values corresponding to the current and historical IoT data. The ith and $(i-1)$th reputations are stored in the vectors $\boldsymbol{\xi_i}$ and $\boldsymbol{\xi_{i-1}}$, respectively.

The reputation values in the current reputation block are evaluated based on the historical reputation data from the previous reputation block and the current information data collected from the IoT devices. Therefore, the vector of reputation values can be evaluated as

$$\boldsymbol{\xi_i} = f(\boldsymbol{\xi_{i-1}}, \boldsymbol{I_{i,1}}, \boldsymbol{I_{i,2}}, ...), \tag{3.3.1}$$

where $f(\cdot)$ is the function to calculate reputations, and $\boldsymbol{I_{i,1}}, \boldsymbol{I_{i,2}}, \ldots$ are the information data stored in IB i as shown in Fig. 3.3. Based on (3.3.1), the reputation values can be used to quickly identify the trust levels of all the IoT devices.

The mapping is evaluated by applying a hash function to the unique timestamp and a vector of IoT device IDs in the header of each block. Only the corresponding reputation block and information block will have the same mapping value. Taking hash-256 function as an example, the mapping value $\boldsymbol{Map_i}$ of the ith block pair can be evaluated as

$$\boldsymbol{Map_i} = \text{SHA256}\left(ts_i \mid \boldsymbol{ID_i}\right), \tag{3.3.2}$$

where $\mid$ is the concatenation of two sequences, and ts_i is the timestamp of the ith block as shown in Fig. 3.3.

3.3.3 DBC Average Storage Size

The average IBC and RBC storage size for randomly distributed fog nodes and transmitting IoT devices can be evaluated based on our block structure in Fig. 3 and PPP-based node deployment described in eq. (3.2.5), which results in

$$\begin{aligned}\mathbb{E}(S_{\text{IBC}}) &= N_{block}(2(256 + L_h) + \mathbb{E}(N_{iot})(2L_h + N_{\mathcal{I}}L_b)) \\ &= N_{block}(2(256 + L_h) + \sum_{m=0}^{\infty} mP_{iot}(m)(2L_h + N_{\mathcal{I}}L_b)) \\ &= N_{block}(\underbrace{2(256 + L_h) + \lambda_{iot}(2L_h + N_{\mathcal{I}}L_b)}_{\mathbb{E}(L_{IB})}),\end{aligned} \tag{3.3.3}$$

and

$$\begin{aligned}\mathbb{E}(S_{\text{RBC}}) &= N_{block}(2(256 + L_h) + \mathbb{E}(N_{iot})(2L_h + 2L_b)) \\ &= N_{block}(2(256 + L_h) + \sum_{m=0}^{\infty} mP_{iot}(m)(2L_h + 2L_b)) \\ &= N_{block}(\underbrace{2(256 + L_h) + 2\lambda_{iot}(L_h + L_b)}_{\mathbb{E}(L_{RB})}),\end{aligned} \tag{3.3.4}$$

respectively, where N_{block} is the number of blocks in each blockchain, and $\mathbb{E}(N_{iot})$ is the expectation of the number of IoT devices transmitting to the target full node, where λ_{iot} is the corresponding IoT device intensity. We denote $N_{\mathcal{I}}$ as the number of entries in each information block body, L_h is the size of the three non-hash entries in the block header, L_b is the size of each entry in the block body, $\mathbb{E}(L_{IB})$ is the average size of a single IB, and L_{RB} is the average size of a single RB. We can observe from (3.3.3) and (3.3.4) that the average storage sizes of IBC and RBC increase with intensities of IoT devices uploading information. Furthermore, based on the PMF in (3.2.2) the total storage size for the RBC in the fog blockchain layer can be evaluated as

$$\mathbb{E}(FS_{RBC}) = \mathbb{E}(N_t) \times S_{RBC} = \sum_{n_i=0}^{\infty} n_i \frac{e^{-\lambda_t}\lambda_t^{n_i}}{(n_i)!} S_{RBC} = \lambda_t S_{RBC}, \tag{3.3.5}$$

where, $\mathbb{E}(N_t)$ is the average number of fog nodes used to store the RBC, and λ_t is the corresponding intensity.

From the above analysis, we can conclude that the DBC architecture dramatically improves the scalability of the IoT network deployed in the fog layer. Since the storage resources of fog layer infrastructures such as the base station, the roadside unit, or the wireless access point are much more restricted and expensive than that of the cloud layer storage resources, it is important to improve scalability by reducing the storage requirement in the fog layer. Offloading the heavyweight IBC to the cloud layer can efficiently lower the storage resource requirements at the fog layer IoT nodes. It is worth noting that the IBC still needs high storage space requirements but with lower cost and restriction in the cloud layer. In addition, the DBC architecture requires total extra storage space cost of one more block header and mapping in each DB pair compared to the single blockchain architecture.

3.3.4 DBC Latency

Next, we evaluate the DBC average latency from the perspective of PPP-based deployments to determine the impact of the node density in real-world applications.

Average Processing Latency

The processing latency for our DBC block structure is the cumulative time required to generate a new block for the IBC and RBC and their corresponding mapping relationship. First, the full node for the information block uses the collected raw data from the IoT devices to generate a new information block. The information will then be transmitted to the nearest RBC full node to generate the corresponding reputation block together with the historical reputation values stored in the previous reputation block. Lastly, the mapping algorithm between this newly generated information and reputation block pair is executed. The processing time of IB is much higher than that of RB and mapping, because the IB data is much larger than the RB and mapping. As such, the average DBC processing time is mainly affected by the processing time of the IBC.

We can evaluate the times required to generate the ith IB, RB, and the mapping relationship shown in Fig. 3.3 as

$$\tau_{proc_{IB}}(i) = O_\tau \left(\text{SHA256}\left(L_{IB}(i)\right)\right), \tag{3.3.6}$$

$$\tau_{proc_{RB}}(i) = O_\tau \left(\text{SHA256}\left(L_{RB}(i)\right)\right), \tag{3.3.7}$$

and

$$\tau_{proc_{map}}(i) = O_\tau \left(\text{SHA256}\left(ts_i \mid \boldsymbol{ID_i}\right)\right), \tag{3.3.8}$$

respectively, where we assume that the SHA-256 hash function is used and $O_\tau(\text{SHA256})$ is the time to execute the hash function based on the size of the raw input data. Based on the size of the IB and RB defined in (3.3.3) and (3.3.4), respectively, we can observe that the processing time of the IB increases with both the number of the information entries N_I and the intensity of active IoT devices λ_{iot} in (3.3.3), whilst the processing time of the RB and mapping increase with λ_{iot}. Therefore, the processing time of the ith DBC block pair can be expressed as

$$\tau_{proc}(i) = \tau_{proc_{IB}}(i) + \tau_{proc_{RB}}(i) + \tau_{proc_{map}}(i). \tag{3.3.9}$$

We note that these processing latencies are measured through program execution time, which is determined by time flags embedded at the start and end of program execution related to eqs. (3.3.6)-(3.3.8).

Transmission Latency

In our DBC architecture, the transmission latency is the time required to transmit a newly generated RB by the full and light fog nodes for the RBC. We consider that the IBs are offloaded to the cloud layer by wired backhaul links, and therefore incurs minimal transmission time compared to the RBC which are transmitted via wireless links. The transmission latency is defined as the transmission time required to transmit a new RB to its nearest neighbor, which can be evaluated as [Li *et al.*, 2019b]

$$\tau_t = \frac{\mathbb{E}(L_{RB})}{C_s}, \tag{3.3.10}$$

where $\mathbb{E}(L_{RB})$ is the expectation of RB size given in (3.3.4), and $C_s(n_i, n_{i+1})$ is the communication throughput which can be expressed as [Haenggi, 2013]

$$C_s \triangleq p_{trans}\lambda_t\mathcal{P}_{suc}, \tag{3.3.11}$$

where each transmitter in the Poisson network decides to transmit with probability p_{trans}, λ_t is the intensity of the transmitting fog nodes, and $\mathcal{P}_{suc}$ is the transmission success probability, which is the probability that the received SINR at the RBC fog node is above a predefined threshold ϵ_{SIR}. Thus, the transmission success probability in a Rayleigh fading wireless network can be expressed as [Haenggi, 2013]

$$\begin{aligned}
\mathcal{P}_{suc} \triangleq& \mathbb{P}_{suc}(P_u > \theta(I_{n_i} + \sigma^2)) \\
=& \mathbb{E}(\exp(-\theta d^{\hat{\alpha}}(\frac{\sigma^2}{P_u} + I)) \mid d = d_1) \\
=& \exp(-\theta\frac{\sigma^2}{P_u}d_1^{\hat{\alpha}})\mathcal{P}_{suc}^{I} \\
=& \exp(-\varepsilon_1 d_1^{\hat{\alpha}})\mathbb{E}_I(\exp(-d^{\hat{\alpha}}I\theta) \mid d = d_1) \\
\overset{(a)}{=}& \exp\left(-\varepsilon_1 d_1^{\hat{\alpha}} - \pi\lambda\varepsilon_2 d_1^2\right),
\end{aligned} \tag{3.3.12}$$

where

$$\varepsilon_1 = \theta \frac{\sigma^2}{P_u}, \tag{3.3.13}$$

and

$$\varepsilon_2 = \theta^{\frac{2}{\hat{\alpha}}} \Gamma\left(1 + \frac{2}{\hat{\alpha}}\right) \Gamma\left(1 - \frac{2}{\hat{\alpha}}\right), \tag{3.3.14}$$

where (a) follows from averaging over the exponential distribution of P_{s,n_i}, and λ_{fog} is the intensity of the interference node set. Specifically, $\mathcal{P}_{suc}$ is the conditional expectation of success probability when $d = d_1$ in (3.3.12) since we consider the fog nodes always transmit to their nearest neighbors.

Substituting (3.3.11), (3.3.12), (3.3.13), and (3.3.14) into (3.3.10), we can derive the transmission latency as

$$\begin{aligned}\tau_t =& \frac{\mathbb{E}(L_{RB})\exp\left(\varepsilon_1 d_1^{\hat{\alpha}} + \pi\lambda\varepsilon_2 d_1^2\right)}{p_{trans}\lambda_t}\\ =& \frac{2(256 + L_h) + \lambda_{iot}(256 + L_h + 2L_b)}{p_{trans}\lambda_t}\\ & \times \exp\left(\theta\frac{\sigma^2}{P_u} d_1^{\hat{\alpha}} + \pi\lambda d_1^2 \theta^{\frac{2}{\hat{\alpha}}} \Gamma\left(1 + \frac{2}{\hat{\alpha}}\right) \Gamma\left(1 - \frac{2}{\hat{\alpha}}\right)\right).\end{aligned} \tag{3.3.15}$$

From (3.3.15), we conclude that the transmission latency can be decreased by reducing the intensity of the IoT device, λ_{iot}, and interference nodes, λ_{fog}, or increasing the intensity of the transmitting fog nodes, λ_t. As such, compared to a conventional single blockchain approach where the information and reputations are stored in all the fog layer nodes, our DBC architecture is expected to achieve a remarkably smaller transmission latency due to the small block size of the RBC. The expression in (3.3.15) also indicates that the transmission latency can be reduced by setting a lower SINR threshold ϵ_{SIR}, shorter deployment distance d_1, larger path loss exponent $\hat{\alpha}$, or higher transmit power P_u.

We note that the low average processing and transmission latency of our DBC is achieved at the cost of higher system complexity. This is because separately generating the IB, RB and mapping requires additional processing time compared with a conventional single blockchain. Additional processing overheads are required to execute the DBC node classification algorithm, while the transmission of RBC in the fog layer will require wireless channel resources which are efficiently minimized due to the small block size of the RB.

Impact Analysis of Processing and Transmission Latency

Eqs. (3.3.6)-(3.3.8) and eq. (3.3.15) highlight the pivotal role of processing and transmission latency in our considered hierarchical low-latency IoT network. While processing latency is significantly influenced by computing frequency, transmission latency is primarily determined by factors such as SIR threshold, distance, fog node intensity, and IoT device intensity. These dynamics underscore the complexity of balancing processing and transmission delays to optimize network performance.

3.3.5 Practical Consideration of PPP-Based Modeling

While PPP serves as a useful theoretical model, it is important to acknowledge its limitations in accurately representing the stochastic properties of real-world wireless node deployments. While PPP provides valuable insights into the spatial distribution of nodes in wireless networks, it may not fully capture the complexities and variations present in actual deployment scenarios. For example, in [Hourani *et al.*, 2019], the authors highlighted this point and strengthened the need for caution when applying PPP assumptions, particularly in practical contexts involving base stations (BS). Therefore, while PPP remains a valuable theoretical tool for analysis, it is essential to exercise discretion and not overstate its practical applicability, especially in scenarios involving BS deployment and network planning.

3.3.6 DBC Security

To evaluate the DBC security, we focus on the time required for a malicious fog node to launch a tampering attack on the RBC and the time required to identify the attack that has occurred. For instance, in the scenario where an attacker possesses the capability to manipulate stored data within a minute, the implementation of effective preventive measures becomes significantly challenging. Conversely, if the perpetrator's efforts necessitate an entire day to execute a successful tampering attack, it affords us an additional day to undertake operational strategies aimed at thwarting the assailant. We note that either a higher tampering time or a lower identification time both correspond to a higher security level.

To analyze the time required for tampering, we adopt the method described in ref [Yu *et al.*, 2020]. We utilize a similar blockchain consensus protocol in our low-latency IoT scenario, in which a malicious user needs to modify the copies of hash values of the target reputation block in more than half of the total number of RBC storage nodes and change the hash values in all the subsequent blocks [Yu *et al.*, 2020]. We assume that the malicious node attempts to attack any fog node since it does not know which nodes are full or light nodes, thus significantly decreasing the probability of a successful attack. We assume that the attacker lacks prior knowledge of the blockchain network. Thus, we define the security level based on the minimum time required for successful tampering. For instance, the attacker might expend additional effort attempting to tamper with earlier blocks, although such attempts would be futile in achieving a successful tampering attack. As such, the security level in terms of blockchain tampering time can be expressed as

$$\tau_{tamp_{min}} = \frac{\lambda_t}{2} \sum_{i=N_{block}-m_t}^{N_{block}} \mathbb{E}(\tau_{proc}^{Att}(i)), \tag{3.3.16}$$

where m_t is the index of the targeted block. From (3.3.16), we clearly see that the DBC security level increases with the reputation block size, the position of the target block in the RBC, and the intensity of full and light nodes for RBC. We note that $\tau_{proc}^{Att}(i)$ denotes the attacker's processing time for a single block. Tampering by the attacker necessitates extensive hash calculations, akin to the block generation process described in eq. (3.3.9). However, this processing time varies depending on the attacker's computing frequency.

As illustrated in Fig. 3.3, any modification to the hash value stored in the IB, RB, or mapping results in corresponding alterations in the other two. This suggests that detecting a tampering attack requires monitoring the shortest time taken for any change to manifest. Given that, for the same node, the processing latency of IB, RB, or mapping is determined by their respective sizes, and the mapping size is the smallest among these three. Hence, to identify the tampering attack, each fog node simply needs time to search and verify the hash values in their RB, which can be evaluated as

$$\tau_{iden} = \sum_{i=N_{block}-m_t}^{N_{block}} \mathbb{E}(\tau_{proc_{map(i)}}). \tag{3.3.17}$$

Since this process can be carried out independently at each fog node, the time required to identify the tampering attack is significantly smaller than the time required for the malicious node to successfully launch the attack given in (3.3.16), which guarantees a high system security.

3.4 Security Analysis

In this section, we discuss the DBC security in terms of storage and computing resources and three different attacks on the IoT devices and fog layer nodes. Finally, we discuss the practical tradeoffs in terms of the performance and security of our DBC architecture.

Our research aims to enhance IoT network security by strategically utilizing storage and computing resources. These resources were chosen based on their critical role in addressing the security challenge of tampering attacks faced by IoT networks, which usually lack storage and computing resources. In our approach, storage resources are leveraged for storing blockchain data on fog nodes, providing a tamper-proof and distributed ledger for securing IoT transactions and data exchanges. In addition, computing resources are utilized for fast and efficient processing of blockchain operations, including block generation and validation, at the network edge. By harnessing the capabilities of storage and computing resources, our research seeks to strengthen the security posture of IoT networks by ensuring data integrity, confidentiality, and resilience against cyber threats and attacks. This strategic deployment of resources is integral to our comprehensive approach towards enhancing IoT network security and protecting IoT devices and systems from emerging security risks and vulnerabilities.

3.4.1 Attack Analysis

Malicious IoT Device

Malicious IoT devices may provide malicious raw data and deliberately change the information stored in IBC. In our DBC architecture, the quality of the raw data can be compared with previously stored information in the IBC to identify malicious data from IoT devices and assign them with low reputations. Based on these reputations, the smart contract can quickly identify and isolate the malicious IoT devices to prevent further harmful impacts from the malicious data.

Information Hiding

Information hiding is a common attack in trust management systems. With information hiding, the malicious IoT device hides the information that have a negative impact on its reputation and only shares the positive ones [Ferrag *et al.*, 2019]. In our DBC, the current reputation value is calculated based on both the previous reputation value and the new information values. Thus, the reputation value of the information hiding IoT device can be decreased when it shares fewer information values compared to previous uploads. In addition, the IB and RB pair is strongly mapped. Therefore, if a specific IoT device uploads data less frequently compared to the other related devices, it can be quickly identified by the smart contract through mapping.

Eclipse Attack

A well-known blockchain attack aimed at the fog node is the eclipse attack [Heilman *et al.*, 2015]. In an eclipse attack, the attacker monopolizes all of the target node's incoming and outgoing connections, thus isolating the target node from the rest of the fog nodes in the network. The attacker can then filter the target node's view of the DBC, force the target node to waste computing power on obsolete views of the blockchain, or co-opt the target block's computing power for its own purposes. Even though the eclipse attack aims to tamper with only one fog node, it could potentially disrupt the whole blockchain network, such as in a Sybil attack [Douceur, 2002].

In the DBC, the eclipse attack can be prevented because DBC is a private blockchain in which the fog nodes are strictly authenticated. Furthermore, the IoT devices can upload their raw data to multiple fog nodes instead of a single node, which means the reputation values of the IoT devices are calculated by a large number of fog nodes in the network. Thus, even if one target node is successfully attacked by the eclipse attack, its irregular data can be identified by comparing it with other nearby nodes.

3.4.2 Security-Performance Tradeoffs

We note that managing two blockchains instead of one will require more computing resources due to the replicated processing of IBC, RBC, and the mapping algorithm between IBC and RBC. Comparing (3.3.15) and (3.3.16), we see that whilst increasing the reputation block size and number of RBC storage nodes improves security will result in a more secure system in the fog layer compared to the conventional SBC architecture, it also results in a higher transmission latency which means more time is needed to propagate a new reputation block to all the storage nodes. This highlights a fundamental trade-off between security and performance in designing our IoT reputation management system.

We also observe that due to the large size of the IB and its storage in the cloud layer, it will be extremely challenging to tamper the IBC. The higher security resulting from storing multiple copies of the RBC in large numbers of full and light nodes will result in higher storage resource consumption in the fog layer. This additional storage cost is minimized by ensuring that each new RB stores only the most recent reputation data for low-latency access by the smart contract. Furthermore, even if the RBC is successfully tampered, our mapping algorithm can be applied to quickly identify the tampering and re-evaluate the reputation values based on the information stored in

Table 3.1: Chapter 3 Simulation Parameters

Parameter	**Value**
Number of blocks in one blockchain N_{block}	$1,000$
Number of information entries in each IB $N_{\mathcal{I}}$	128
Size of non-hash entries in the block header L_h	8 Bytes
Size of each entry in the block bodyL_b	8 Bytes
Minimum space required for storing RBC $s_{\min_{RBC}}$	6 GB
Minimum computation required to generate IB $c_{\min_{IB}}$	9 GHz
Minimum computation required to generate RB $c_{\min_{RB}}$	6 GHz
Intensity of the entire fog node set λ_{fog}	1×10^3 [Sun *et al.*, 2019a]
Intensity of the transmitting IoT devices λ_{iot}	1×10^6 [Sun *et al.*, 2019a]
Transmission probability p_{trans}	0.5
Transmit power of fog node P_u	20 dBm [Sun *et al.*, 2019a]
Noise power to fog node σ^2	-104 dBm
SIR threshold ϵ_{SIR}	0 dB
Path loss exponent $\hat{\alpha}$	2.5 [Sun *et al.*, 2019a]
Distance of the desired link d_1	10 m

the corresponding information blocks to isolate or blacklist the detected malicious nodes.

3.5 Performance Evaluation

In this section, we present numerical simulations to evaluate the storage efficiency, processing and transmission latency, and security of the DBC architecture using a Matlab simulation environment based on randomly located fog nodes and the number of transmitting IoT devices. The simulation scenario is set as shown in Section 3.2, where the cloud server and the fog nodes are located in the cloud and fog layers, respectively. In the fog layer, we assume that the locations of the fog nodes are modeled according to a random PPP over a given two-dimensional area, and their

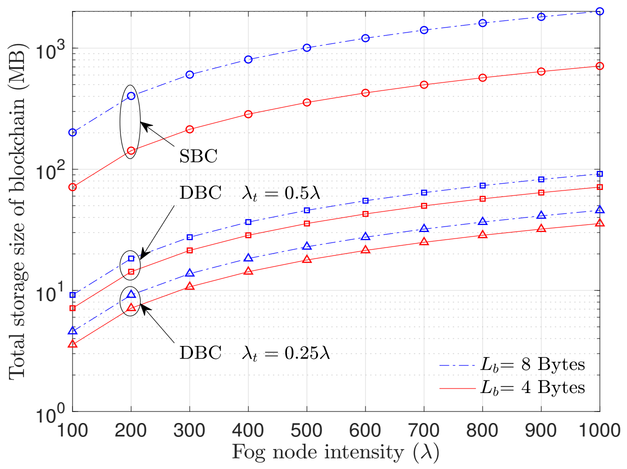


Figure 3.4: Blockchain storage size of SBC and DBC versus fog node density (per km^2).

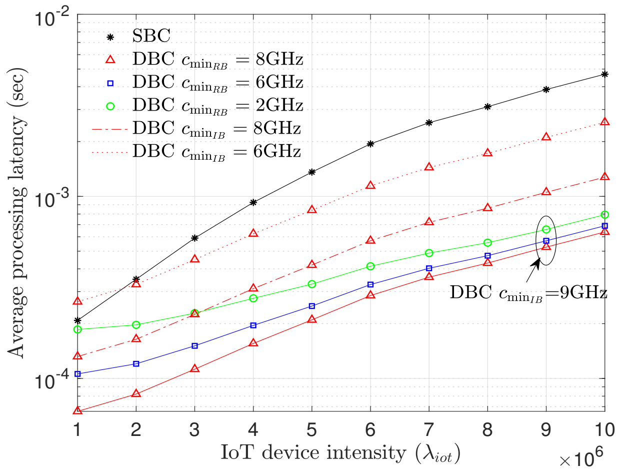
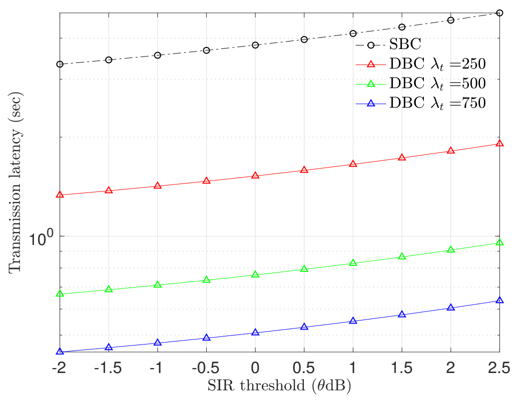


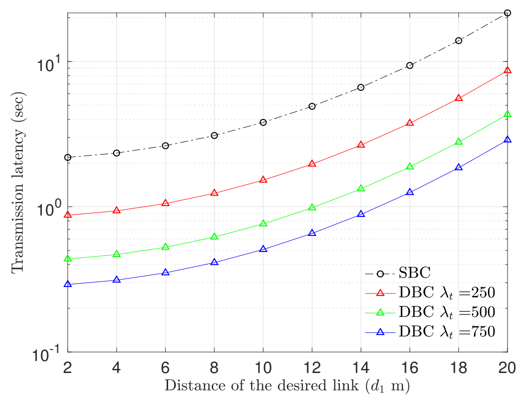
Figure 3.5: Average processing latency versus IoT device density (per km^2).

storage space and computing resources follow uniform distributions of $S_i \sim \mathcal{U}[2, 10]$ GB and $F_i \sim \mathcal{U}[2, 10]$ GHz, respectively. Unless otherwise mentioned, the simulation parameters are summarized in Table 3.1.

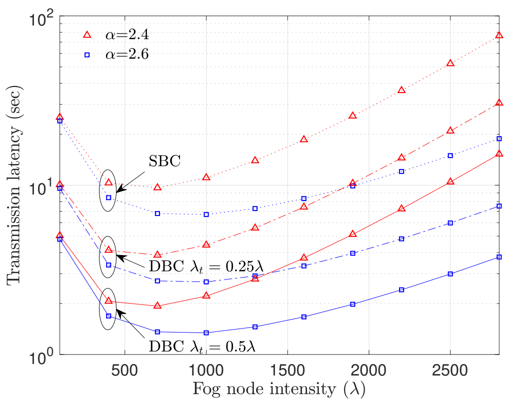
Fig. 3.4 plots the blockchain storage size derived in (3.3.5) versus the fog node intensity of the SBC and DBC. We consider different transmission node intensities,



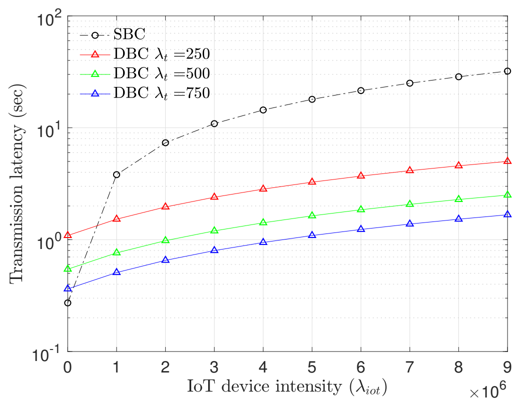
(a) Transmission latency versus the SIR threshold.



(b) Transmission latency versus the distance of the desired link.



(c) Transmission latency versus fog node density (per km^2).



(d) Transmission latency versus IoT device density (per km^2).

Figure 3.6: Transmission latency for different key design parameters.

and different block entry size L_b. This figure shows that the storage space of both the single and double blockchains increases with the intensity of fog nodes, and the storage size of double blockchain RBC is always smaller than that of the SBC with the same λ_t and L_b. We also observe that the storage size of DBC increases slower than SBC as L_b increases. This is because the DBC node classification algorithm only stores the lightweight RB in the transmission fog nodes compared to the SBC that stores both information and reputation values in all fog nodes. As expected, a smaller λ_t corresponds to a smaller storage size for the RBC, since fewer transmission nodes will be selected amongst the fog nodes. For example, when $\lambda = 10^3$ with 8 Bytes L_b, the SBC requires a total storage size of approximately 2GB whilst the DBC only requires 92MB and 46MB when λ_t is 500 and 250, respectively.

In Fig. 3.5, we plot the average processing time to generate a new block from (3.3.9) versus the IoT device density for different $c_{\min_{IB}}$ and $c_{\min_{RB}}$. The processing time is measured by the being and end flags of block generation program. We observe from this figure that both higher $c_{\min_{IB}}$ and $c_{\min_{RB}}$ lead to a higher average processing time, because a higher computation threshold means a smaller number of nodes will participate in the IB and RB generation process. This figure also indicates that $c_{\min_{RB}}$ influence the processing latency more significantly with a lower IoT device intensity. This is because a lower IoT device intensity corresponds to a smaller IB size, which leads to less contribution to the entire DBC process latency as noted in (3.3.9). This figure further shows that SBC can outperform DBC at low IoT device intensities of around 1×10^6. This is because the additional time to process the RB and mapping in DBC causes comparable latency as IB when the IB block size is very small, which indicates that the DBC processing latency is more suitable for large-scale IoT networks. As expected, when the IoT device intensity is large, the DBC is primarily determined by the IB threshold $c_{\min_{IB}}$. For example, increasing the RB and IB computation thresholds from 6GHz to 8GHz when λ_{iot} is 1×10^7, results in decreasing the processing latency by approximately 7% and 50%, respectively.

Fig. 3.6 plots the transmission latency for five key design parameters characterized in eq. (3.3.15). Fig. 3.6(a) shows that the transmission latency increases with increasing SIR threshold ϵ_{SIR} because increasing the SIR threshold corresponds to decreasing the transmission success probability shown in (3.3.12). Fig. 3.6(b) demonstrates the transmission latency increases with the growing distance of the desired link d_1 because a longer desired link distance will result in more severe fading, path loss, and interference experienced by the desired link. Fig. 3.6(d) indicates that the transmission latency increases with higher IoT device intensity λ_{iot} since a higher IoT device intensity corresponds a larger RB size to be transmitted by each node. Interestingly, Fig. 3.6(c) shows that the transmission latency decreases with increasing path loss exponent $\hat{\alpha}$. This is because a higher path loss exponent can effectively reduce the transmission latency by decreasing the interference at each receiver due to the interference-limited nature of the fog network.

Another interesting observation in Fig. 3.6(c) is that the transmission latency first decreases as λ_{fog} increases since deploying more nodes corresponds to a higher communication throughput as shown in (3.3.11). As the intensity keeps increasing, the transmission latency starts to increase. This is because the interference is more severe as the network gets denser. This figure further shows that a higher path loss exponent leads to higher tolerance of interference intensity since the total interference decreases with increasing $\hat{\alpha}$. Similar to Fig. 3.5, we see in Fig. 3.6(d) that when the intensity of IoT devices is very low, SBC can outperform DBC. This is because a low IoT device intensity corresponds to less raw data and comparable block sizes in the SBC and DBC. However, as IoT device intensity increases, the SBC block sizes stored in the fog nodes will be larger than the DBC resulting in a lower transmission latency for the DBC.

Fig. 3.7 plots the minimum time required to tamper the DBC from (3.3.16) for

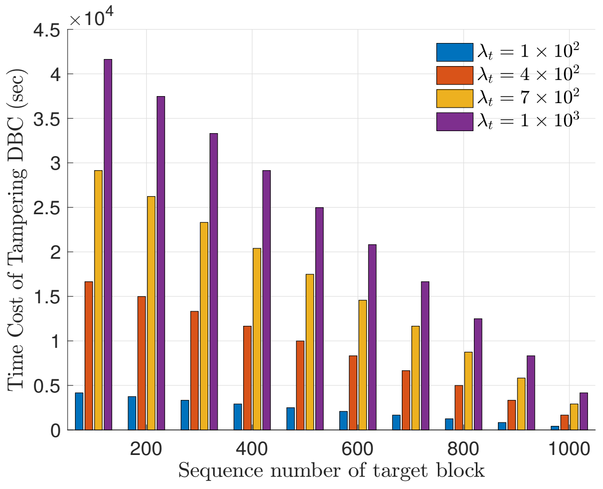


Figure 3.7: Minimum time to tamper the DBC versus the targeted block sequence number with different RBC transmission node intensities (λ_t).

different targeted block sequence numbers and different intensity of the RBC transmitting nodes. We consider a dense IoT network scenario in which the intensity of the IoT device at $\lambda_{iot} = 5 \times 10^9$ nodes per square kilometer. The figure highlights that our DBC architecture is highly secure against tampering attacks. For example, we see that the time required for the tampering attack can be up to 11.6 hours when $\lambda_t = 1 \times 10^3$, which is much higher than the time for single blockchain solutions such as in [Yu *et al.*, 2020], which indicated a tampering time of approximately 6 hours. This shows that a large number of storage nodes in the fog layer makes it difficult for a malicious node to successfully tamper more than half of all blockchain copies, which highlights the effectiveness of our DBC architecture to prevent tampering attacks.

In Fig. 3.8, we highlight a fundamental trade-off between the transmission latency and the ability to resist a tamper of the DBC architecture. A lower transmission latency and a higher ability to resist the tampering attack is desirable. However, as shown in Fig. 3.8, when the transmission latency increases, the tampering attack time

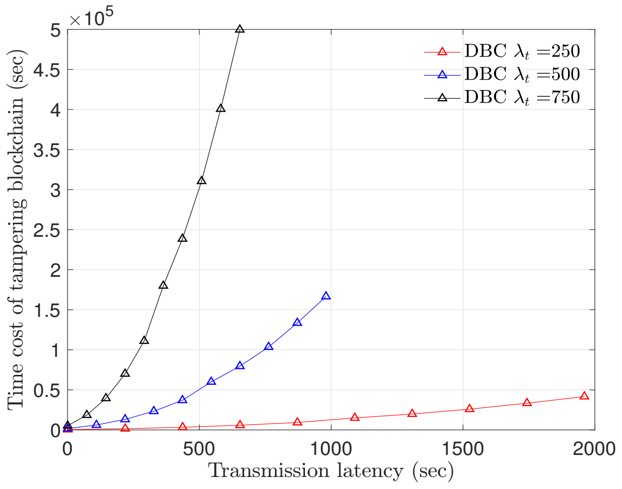


Figure 3.8: Trade-off between DBC transmission latency and tampering time.

also increases. This figure further shows that deploying more transmitting nodes is an effective solution to decrease the transmission latency and improve the security level. Of course, deploying more nodes is expensive and may not always be feasible. The figure shows that the network deployment should be carefully planned to ensure that a minimum transmission latency and maximum ability to resist tampering can be achieved for the given network density scenario.

3.6 Chapter Summary

We presented detailed analysis and stochastic modeling of our scalable DBC architecture [Hao *et al.*, 2021] for IoT information and reputation management. We consider a communication network in which the fog node locations and transmitting IoT device numbers are randomly distributed to reflect real-world networks. The DBC was composed of a lightweight RBC containing IoT reputation values which are stored in a selected number of fog layer nodes and a heavyweight IBC containing large amounts of IoT data stored in the cloud layer. We apply a DBC node classification algorithm to identify suitable fog layer nodes to generate the IBC and RBC, and store the RBC based on their resource constraints. A mapping relationship between RBC and IBC is embedded in the DBC architecture, which helps to quickly prevent and identify malicious tampering attacks on the blockchain. Numerical analysis and simulation results show that our stochastic DBC architecture can provide a highly scalable, low-latency and secure blockchain performance for IoT information and reputation management in large-scale heterogeneous communication networks with realistic fog node deployments and random numbers of transmitting IoT devices.

Chapter 4

Secure DRL for Dynamic Resource Allocation in Wireless MEC Networks

This chapter proposes a blockchain-secured DRL (BC-DRL) optimization framework for data management and computing resource allocation in decentralized wireless MEC networks. In our framework, we design a low-latency reputation-based proof-of-stake (RPoS) consensus protocol to select highly reliable blockchain-enabled BSs to securely store MEC user requests and prevent data tampering attacks. We formulate the MEC computing resource allocation optimization as a constrained MDP balancing processing latency and DoS probability. We use the MEC aggregated features as the DRL input to significantly reduce the high-dimensionality input of the remaining service processing time for individual MEC requests. Our designed constrained DRL effectively attains the optimal computing resource allocations that are adapted to the dynamic DoS requirements. We provide extensive simulation results and analysis to validate that our BC-DRL framework achieves higher security, reliability, and resource utilization efficiency than benchmark blockchain consensus protocols and MEC computing resource allocation algorithms.

4.1 Chapter Introduction

Security is a major concern in mobile edge computing (MEC) service provisioning given the prevalence of data tampering attacks leading to disruptive denial-of-service (DoS) in decentralized wireless networks [Lin *et al.*, 2023; Liu *et al.*, 2020]. Blockchain-based data management has been recently considered to prevent data tampering attacks and ensure the integrity of MEC user requests in wireless networks [Zhang *et al.*, 2021a]. In blockchain-secured MEC networks, the base stations (BSs) are also blockchain nodes which store critical data at all nodes in the blockchain network according to a given consensus protocol. The most well-known blockchain consensus protocol is proof-of-work (PoW) [Nakamoto, 2008], which is not suitable for wireless MEC networks with limited BS computing resources and strict latency requirements [Pokhrel *et al.*, 2020]. In [Ling *et al.*, 2021; Xiong *et al.*, 2020], it was shown that the PoW consensus incurred lengthy delays due to high computations for block validation and requiring all nodes to compete in the block generation.

Significant research efforts have focused on reducing the high computation overhead and processing latency for blockchain consensus [Asheralieva *et al.*, 2020; Castro and Liskov, 1999; Hao *et al.*, 2020; Kang *et al.*, 2019b; Tschorsch and Scheuermann, 2016; Xiao *et al.*, 2020; Yang *et al.*, 2019]. Among them, a popular approach is the practical Byzantine fault tolerance (PBFT) consensus protocol, which reduces the block generation time by selecting one blockchain node to generate a new block [Castro and Liskov, 1999]. PBFT consensus still has high computation and latency for block validation, because all nodes need to validate the generated block. Proof-of-stake (PoS) consensus is another protocol that can reduce computation and latency overheads by selecting a trusted subset of blockchain nodes to validate the generated block [Tschorsch and Scheuermann, 2016]. However, PoS consensus is vulnerable to attacks since the highest stake miner node that is selected for block generation can be easily targeted by attackers [Yang *et al.*, 2019]. Clearly, there is a pressing need to design a novel blockchain consensus that can significantly reduce the high computation overheads without jeopardizing the security level.

Apart from security, a further challenge in dynamic MEC networks is to efficiently allocate the computing resources in each time slot [Zappone *et al.*, 2019], since allocating more computing resources in the current time slot results in a smaller processing latency for these users, but a potentially higher DoS probability due to insufficient resources for future users. This fundamental trade-off between processing latency and DoS probability in MEC networks can be managed by formulating the optimal resource allocation as a sequential decision-making problem [Chen *et al.*, 2022; Tang *et al.*, 2021]. Dynamic programming is a traditional model-based approach used for resource allocation in sequential decision-making problems [Busoniu *et al.*, 2010]. However, it is challenging to apply in large-scale problems due to the exponential growth of state and action spaces [Taylor, 1994]. To overcome this challenge, deep reinforcement learning (DRL) algorithms have been employed [Wang *et al.*, 2022a; Wang *et al.*, 2023], and constrained DRL is an effective solution to address the explicit requirements on constraints by reformulating the original optimization as a constrained Markov decision process (MDP) [She *et al.*, 2021]. Recently, there is an urgent need to consider security constraints in DRL with the emergence of blockchain-secured MEC networks [Xu *et al.*, 2023a].

Numerical examples are provided to demonstrate the high-security, high-reliability, resource-saving, and low-latency advantages of our BC-DRL solution. We present detailed analysis and performance comparisons with existing PoS consensus protocol and DRL-based resource allocation algorithms, giving insights for implementing future secure blockchain and DRL-empowered dynamic resource allocations.

4.2 Related Works

Most MEC blockchain research has focused on enhancing the security of resource-efficient PoS-based consensus protocols. The authors of [Yang *et al.*, 2019] proposed a secure network management scheme by designing a PoS-based consensus with random node selection in each block generation. To prevent data tampering attacks by the edge node, the authors of [Xiao *et al.*, 2020] employed a joint PoW and PoS consensus protocol storing the reputations of edge devices in the blockchain. In [Asheralieva *et al.*, 2020; Hao *et al.*, 2020; Kang *et al.*, 2019b], it was shown that carefully evaluating the reputation values to select highly reliable blockchain nodes for blockchain management can effectively reduce the computation overhead and resist potential blockchain attacks. Since reputation is a long-term evaluation metric, the authors in [Hao *et al.*, 2020] identified malicious users by evaluating their reputations based on current user data and historical reputations stored in a blockchain. To ensure secure miner selection, the authors in [Kang *et al.*, 2019b] used a multiweight model considering past interactions with other vehicles to evaluate trusted reputations of blockchain nodes. In [Asheralieva *et al.*, 2020], the reputations of MEC BSs acting as blockchain nodes are evaluated based on feedback from both their users and other MEC BSs.

Some recent research efforts have focused on optimizing resource allocation in blockchain-secured MEC networks [Feng *et al.*, 2020b; Guo *et al.*, 2018; Zuo *et al.*, 2021]. In [Feng *et al.*, 2020b], an iterative optimization approach was used to minimize the weighted sum of MEC energy consumption and blockchain latency in an MEC system. In [Zuo *et al.*, 2021], the authors analyzed a PoW-based consensus protocol and used a game-theoretic optimization framework to solve the target non-cooperative resource allocation. Reinforcement learning offers a promising approach to address the challenge of sequential decision-making, thereby presenting potential applications in the realm of computing resource allocation within MEC networks. To further improve the efficiency of resource allocation, more recent research has proposed to apply secure DRL, capable of handling high dimensional input of the neural network, in MEC service provisioning. In [Guo *et al.*, 2018], the authors used an unconstrained DRL approach to maximize the weighted sum of blockchain throughput and the reciprocal of MEC delay. However, unconstrained DRL may encounter difficulties in explicitly satisfying dynamic constraints, which can be addressed by transforming the problem into the dual domain to optimize the weights between the objective and constraints [Liang *et al.*, 2018]. In [Li *et al.*, 2021a], constrained DRL was applied in a virtual reality network, where the weight between the video loss ratio and processing latency is optimized. To further improve the training efficiency of DRL, researchers have explored methods to improve training efficiency, such as choosing low dimension features to reduce the complexity of the optimization problem [Dong *et al.*, 2019], and applying transfer learning of pre-trained parameters when new MEC devices join the network [Shuai *et al.*, 2023]. How to improve security and training efficiency for DRL in dynamic MEC networks still remains an open problem for further investigation.

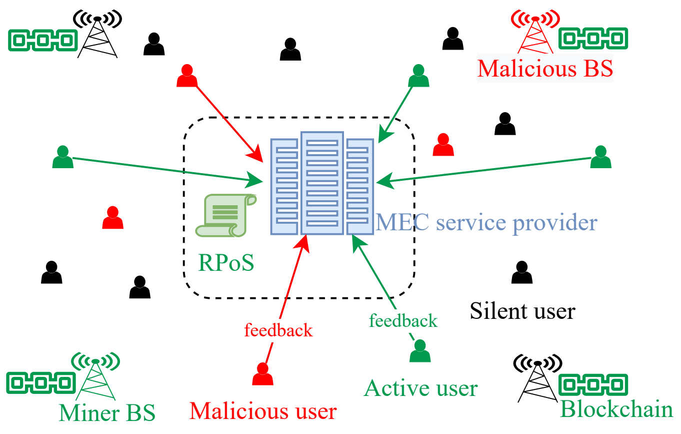


Figure 4.1: Our RPoS blockchain consensus selects trusted BSs for MEC service provisioning and blockchain management using feedback from all users to prevent BS denial-of-service (DoS) attacks from both malicious BSs and users.

4.3 System Model

In our model, an MEC service provider employs N_{B} BSs with overlapping coverage to satisfy the time-varying requests for computing resources from multiple users with DoS probability constraints for the BSs. In each time slot, we assume each user either stays silent or sends a request with a random workload to the service provider. The workload is defined as the required CPU cycles to complete a user's task. Thus, the number of user requests and the CPU cycles required can be modeled as two independent arrival processes. In each time slot, one BS is selected by the MEC service provider to process the received MEC requests from the users. Depending on the available resources, the BS applies the DRL algorithm to allocate or deny resources to these requests in the current time slot. The RPoS consensus is used to securely store the user requests and select BSs to serve the users.

4.3.1 Attack Model

Fig. 4.1 depicts the considered decentralized wireless MEC network in a specific time slot. The green and black BSs are trusted BSs that participate in blockchain management and MEC service provisioning in the current time slot, while the red BSs are untrusted. The green and black users are non-malicious users sending truthful feedback of the BS service provisioning, while the red users are malicious users. We assume that the majority of BSs and users are non-malicious, and user feedback is used to evaluate the BS reputation and DoS probability. Our user feedback-based approach helps the service provider to independently identify malicious BSs and is different from previous wireless blockchain consensus designs, where the BS is responsible for evaluating the reputations of the blockchain nodes.

While conventional networks commonly utilize diverse techniques, such as certificate exchange and authentication, to deter unauthorized access and mitigate intrusion attacks, the assurance of consistently successful prevention remains elusive. It is not guaranteed that the intrusion attack could always be successfully prevented. For example, the attacker could be an existing BS or user who has already got authorization in the blockchain network. Consequently, our analysis of attack models, specifically tampering attacks from malicious BSs and users, is oriented towards mitigating subsequent ramifications, particularly those stemming from tampering attacks. These models are detailed as follows.

Miner BS Attacks

We consider an attacker aims to launch a blockchain attack on the miner BS node by monopolizing all the incoming and outgoing connections from the miner BS to

Table 4.1: Possible Feedback From Individual User

Case	1	2	3	4	5
Active user with request	Yes	Yes	Yes	Yes	No
Service provided	Yes	No	No	Yes	Yes/No
Feedback from user	1	0	1	0	0/1
The user feedback is malicious	No	No	Yes	Yes	Yes

the surrounding BSs [Heilman *et al.*, 2015]. In existing PoS protocols, an attacker can identify the miner BS once it has successfully deciphered the stake evaluation mechanism since the highest stake miner is always selected [Yang *et al.*, 2019]. Our proposed RPoS mitigates this attack by randomly selecting the miner BS from a subset of high reputation BSs to provide services to the requesting users.

Malicious User Feedback Attacks

We consider that each user sends feedback indicating whether their requests in the current time slot were served or not. The green users send feedback indicating whether their requests were served in the previous time slot, whilst the black users had no requests and do not send any feedback. The red users are malicious users with or without requests, which are aiming to disrupt the BS reputation evaluation by sending untruthful feedback. Table 4.1 summarizes all possible user feedback in different cases.

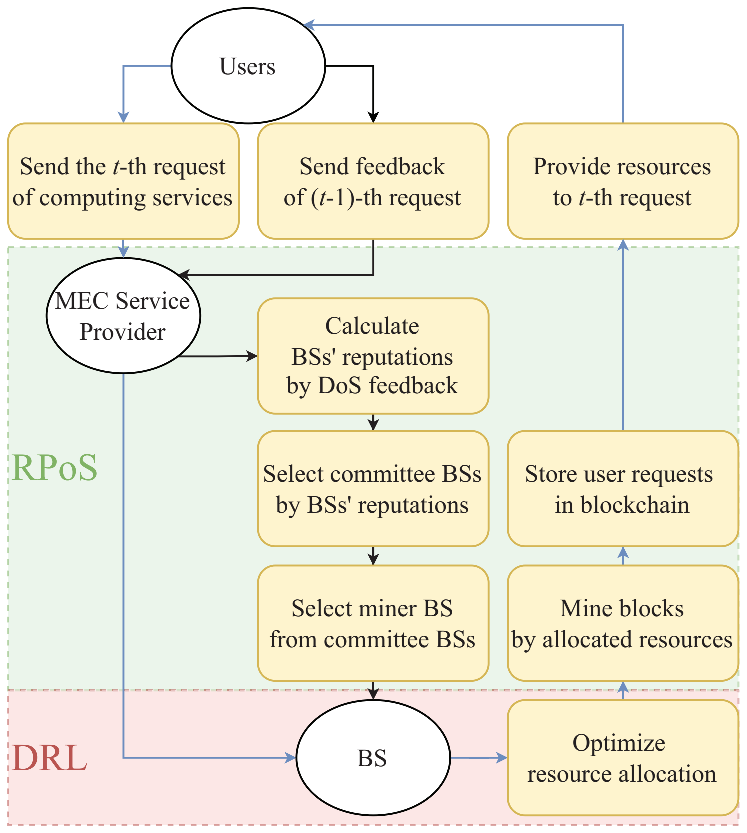

Users
Send the t-th request of computing services
Send feedback of (t-1)-th request
Provide resources to t-th request
MEC Service Provider
Calculate BSs' reputations by DoS feedback
RPoS
Select committee BSs by BSs' reputations
Store user requests in blockchain
Select miner BS from committee BSs
Mine blocks by allocated resources
DRL
BS
Optimize resource allocation

Figure 4.2: Blockchain-secured deep reinforcement learning (BC-DRL) framework for efficient and secure computing resource allocation.

4.3.2 BC-DRL Solution

Fig. 4.2 shows the decentralized architecture of our BC-DRL solution, which includes three entities: 1) Users, 2) MEC service provider managing the RPoS consensus, and 3) BSs implementing the DRL-based computing resource allocation algorithm. Their respective actions are detailed as follows.

Users

In each time slot, a random number of users send computing service requests to the MEC service provider. The users may also send feedback to the service provider indicating if their requests in the previous time slot were satisfied.

MEC Service Provider

First, the feedback received from the users is utilized to evaluate the BSs' reputations by Bayesian inference. Then, to improve the resource utilization efficiency without jeopardizing the system security level, trusted committee BSs with high reputations are selected to manage the blockchain network. Furthermore, an optimal BS is selected as the miner BS providing computing services to the users in each time slot. Lastly, all the user requests are packaged in a new block and stored in the blockchain to prevent data tampering attacks from changing the user requests and enhance the reliability of the DRL-based resource allocation.

BS

Implements the DRL algorithm optimizing computation resource allocation for the selected miner BS to mine the new block and serve user requests.

4.4 RPoS-Based Blockchain Management

In this section, we introduce the BS reputation evaluation to select trusted BSs. Next, we outline the RPoS consensus protocol and derive the CPU cycles for block generation and commitment which is used to evaluate the blockchain processing latency. Lastly, we analyze the tampering attack-resistant ability and discuss the tradeoff between security and resource consumption.

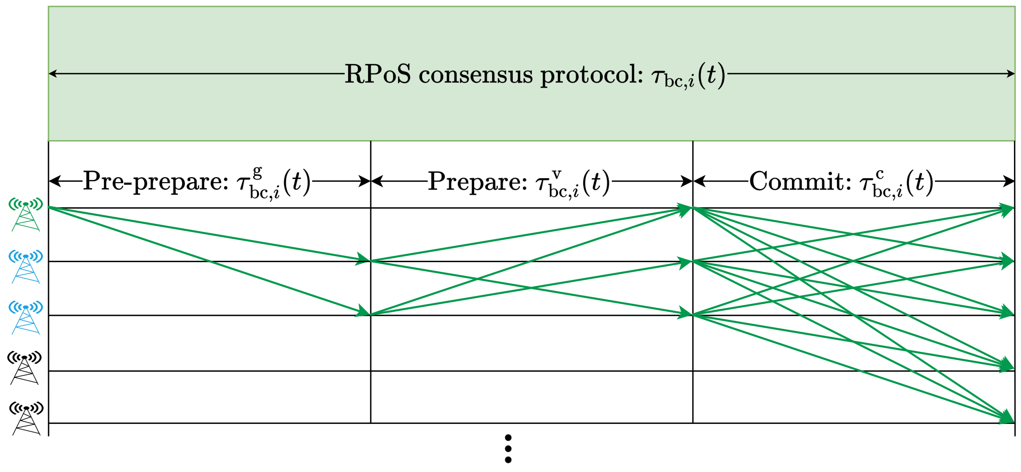

RPoS consensus protocol: $\tau_{\text{bc},i}(t)$
Pre-prepare: $\tau^{\text{g}}_{\text{bc},i}(t)$
Prepare: $\tau^{\text{v}}_{\text{bc},i}(t)$
Commit: $\tau^{\text{c}}_{\text{bc},i}(t)$

Figure 4.3: RPoS consensus for generating, validating, and committing a new block where the green BS is the miner BS, the blue BSs are the validator BSs, and the black BSs are the remaining BSs in the network.

4.4.1 BS Reputation Evaluation for Blockchain Consensus

The BS reputation evaluation of our RPoS consensus is based on Bayesian inference of the aggregated user feedback. We denote $e_i(t)$ as the event that the requests in the t-th time slot are served by the i-th BS, and $\Pr\{e_i(t)\}$ as the prior probability of event $e_i(t)$. Similarly, we denote $\overline{e}_i(t)$ as the complementary event of $e_i(t)$ that requests are denied by the i-th BS. Hence, we can obtain that $\Pr\{\overline{e}_i(t)\} = 1 - \Pr\{e_i(t)\}$.

User Feedback Mechanism

We define $\mathcal{K}(t)$ as the set of users sending feedback in the t-th time slot, and denote $d_{i,k}(t)$ as the feedback from the k-th user. Specifically, $d_{i,k}(t) = 0$ indicates that the k-th user is served by the i-th BS, and $d_{i,k}(t) = 1$ indicates that the service is denied.

Bayesian-Based DoS Inference Evaluation

Since malicious users may send feedback without previously making any requests, it is necessary to evaluate the probability that the BS has successfully served the user requests, which is also known as the DoS inference. The received feedback from the users in the t-th time slot is denoted by $\mathcal{D}_{i,\mathcal{K}}(t) = \{d_{i,1}(t), d_{i,2}(t), \cdots, d_{i,K}(t)\}$. Given the observation of $\mathcal{D}_{i,\mathcal{K}}(t)$, we obtain the DoS inference as the probability that the user requests are served by the i-th BS in the t-th time slot by Bayesian inference as [Raya *et al.*, 2008; Yang *et al.*, 2019]

$$I_i(t) \triangleq \Pr\{e_i(t)|\mathcal{D}_{i,\mathcal{K}}(t)\} = \frac{\Pr\{e_i(t)\}\Pr\{\mathcal{D}_{i,\mathcal{K}}(t)|e_i(t)\}}{\Pr\{\mathcal{D}_{i,\mathcal{K}}(t)\}}, \tag{4.4.1}$$

where $\Pr\{\mathcal{D}_{i,\mathcal{K}}(t)\} = \Pr\{e_i(t)\}\Pr\{\mathcal{D}_{i,\mathcal{K}}(t)|e_i(t)\} + \Pr\{\overline{e}_i(t)\}\Pr\{\mathcal{D}_{i,\mathcal{K}}(t)|\overline{e}_i(t)\}$ is the prior probability of $\mathcal{D}_{i,\mathcal{K}}(t)$. We have $\Pr\{\mathcal{D}_{i,\mathcal{K}}(t)|e_i(t)\} = \prod_{k\in\mathcal{K}(t)}\Pr\{d_{i,k}(t)|e_i(t)\}$ and $\Pr\{\mathcal{D}_{i,\mathcal{K}}(t)|\overline{e}_i(t)\} = \prod_{k\in\mathcal{K}(t)}\Pr\{d_{i,k}(t)|\overline{e}_i(t)\}$ because different users generate feedback independently. We denote $\Pr\{d_{i,k}(t)|e_i(t)\}$ and $\Pr\{d_{i,k}(t)|\overline{e}_i(t)\}$ as conditional probabilities that the requests are served under the conditions of $e_i(t)$ and $\overline{e}_i(t)$, respectively.

BS Reputation Evaluation

The reputation in the t-th time slot is updated according to

$$\xi_i(t) = \begin{cases} \xi_i(t-1), & \text{if } \mathcal{D}_{i,\mathcal{K}}(t) = \varnothing \\ \xi_i^{\mathrm{a}}(t), & \text{if } \mathcal{D}_{i,\mathcal{K}}(t) \neq \varnothing \end{cases} \tag{4.4.2}$$

where $\xi_i^{\mathrm{a}}(t)$ represents the updated reputation when user feedback exists. We note that eq. (4.4.2) indicates the i-th BS in the t-th time slot keeps the same value as that in the $(t-1)$-th time slot if there is no feedback received in the t-th time slot, otherwise evolves to an updated value, $\xi_i^a(t)$, which is a weighted sum of the current

Bayesian inference and the historical reputations, i.e.,

$$\xi_i^{\mathrm{a}}(t) = \vartheta_{\mathrm{I}} I_i(t) + (1 - \vartheta_{\mathrm{I}})\xi_i^{\mathrm{h}}(t), \tag{4.4.3}$$

where ϑ_{I} is defined as the weight coefficient of reputation inference indicating the weight of the Bayesian inference, and $\xi_i^{\mathrm{h}}(t) = \frac{1}{\tau_\xi} \sum_{t'=t-1}^{t-\tau_\xi} \beta_\xi(t-t')\xi_i(t')$ is the expected influence of historical reputations in the past τ_ξ time slots [Hao *et al.*, 2021], where $\beta_\xi(t-t')$ is the discount factor of the historical reputations in the $(t-t')$-th time slot (e.g. $\beta_\xi(t) = e^{-t}$ [Malik *et al.*, 2019], $\beta_\xi(t) = (1/2)^t$, $\beta_\xi(t) = t^{-1}$).

4.4.2 Proposed RPoS Consensus Protocol

Fig. 4.3 shows the three main steps of the RPoS consensus protocol for generating, validating, and committing a new block to the blockchain. Specifically, in the pre-prepare step, the MEC service provider assigns one BS from the committee as the miner BS to generate a new block by computing a unique hash signature based on the data prepared for packaging in the block. Next, in the prepare step, the newly generated block is validated by all the other BSs in the committee, known as validator BSs. To do so, each validator BS computes their signatures for comparison with the hash signatures generated by all the other committee BSs. Lastly, in the commit step, the new block is stored in all the BSs in the network. We define $\mathcal{N}_{\mathrm{M}}(t)$ and $\mathcal{N}_{\mathrm{V}}(t)$ as the sets of the committee miner BS and validator BSs in the t-th time slot.

Next, we detail the proposed RPoS consensus protocol, which involves the trusted committee BS and miner BS selection process using our BS reputations evaluated in eq. (4.4.3).

Committee BSs Selection

In our BC-DRL framework, we select a subset of BSs with reputations higher than a threshold to participate in the proposed RPoS consensus protocol. We denote $\xi_\eta(t)$ as the threshold reputation value, which can be evaluated as

$$\xi_\eta(t) = \eta\bar{\xi}(t) = \frac{\eta}{N_\mathrm{B}}\sum\nolimits_{i\in\mathcal{N}_\mathrm{B}} \xi_i(t), \tag{4.4.4}$$

where $\overline{\xi}(t)$ is defined as the average reputation of the overall BSs in the network in the t-th time slot, and η is defined as the weight coefficient of the reputation threshold. We note that η is a configurable parameter that directly impacts the reputation requirement. Specifically, a larger value of η corresponds to a higher reputation requirement for the committee BSs. Consequently, a higher reputation requirement implies a higher security level, as the committee members are expected to possess a correspondingly higher reputation value than the required threshold. However, it's essential to acknowledge that setting a higher reputation requirement may result in fewer BSs joining the committee. This reduction in the number of participating BSs could have adverse effects, such as reduced redundancy in the decentralized MEC blockchain network. Therefore, the selection of η should be carefully considered to strike a balance between security requirements and network performance. In practice, the optimal value of η would depend on various factors, including the specific application scenarios, network topology, available computing resources, and desired security objectives in the low-latency MEC network.

Based on (4.4.4), the number of BSs elected to the blockchain committee in the t-th time slot be calculated by

$$N_\mathrm{C}(t) = \sum\nolimits_{i\in\mathcal{N}_\mathrm{B}} \mathbb{1}\{\xi_i(t) \geq \xi_\eta(t)\}, \tag{4.4.5}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function, which equals to one if $\xi_i(t) \geq \xi_\eta(t)$, and equals to zero otherwise. The subset of BSs with reputations higher than $\xi_\eta(t)$ is referred to as the blockchain committee in BC-DRL.

Miner BS Selection

In RPoS, we consider that the miner BS is randomly selected from the subset of trusted committee BSs in each time slot to mitigate against miner BS attacks. As such, the probability of an attacker identifying the miner BS is

$$p_{\mathrm{M}}(t) \triangleq \Pr\{\mathcal{N}_{\mathrm{A}}(t) = \mathcal{N}_{\mathrm{M}}(t)\} = \frac{N_{\mathrm{M}}(t)}{\mathbb{E}[N_{\mathrm{C}}(t)]}, \tag{4.4.6}$$

where $\mathcal{N}_{\mathrm{A}}(t)$ indicates the set of BSs under attack, and $\mathbb{E}[\cdot]$ denotes the expectation operation. We observe that the probability of successful attacks on the miner BS decreases with increasing size of the blockchain mining committee, $N_{\mathrm{C}}(t)$.

4.4.3 CPU Cycles Requirement and Latency

CPU Cycles Requirement Required by Miner BS

We can observe from Fig. 4.3, the miner BS performs computations in the pre-prepare and commit steps. If the i-th BS is assigned as a miner in the t-th time slot, the required CPU cycles for block generation and validation is given by

$$\begin{aligned} f_{\mathrm{bc},i}(t) &= f^{\mathrm{g}}_{\mathrm{bc},i}(t) + f^{\mathrm{c}}_{\mathrm{bc},i}(t) \\ &= \mathbb{1}\{i \in \mathcal{N}_{\mathrm{M}}(t)\}\kappa_{\mathrm{bc}} L_{block}(t)(1 + N_{\mathrm{V}}(t)), \text{ (CPU cycles)} \end{aligned} \tag{4.4.7}$$

where $f^{\mathrm{g}}_{\mathrm{bc},i}(t)$ and $f^{\mathrm{c}}_{\mathrm{bc},i}(t)$ are the CPU cycles required by the pre-prepare and commit steps, respectively. The miner BS needs to calculate the hash value for its unique signature to be added to the new block in the pre-prepare step, and also calculates all the signatures of committee validator BSs in the block in the commit step.

We note that the CPU cycles required by the miner BS are proportional to the real-time block size, which is defined by

$$L_{block}(t) = L_{head} + L_{body}(t), \quad \text{(bytes)} \tag{4.4.8}$$

where L_{head} is the size of the block header, which has a fixed size. The t-th block header stores the hash value of the $(t-1)$-th block. The blockchain structure ensures the data in each block is resistant to tampering attacks. The size of the block body is given by $L_{body}(t) = L_{\mathrm{c}}\lambda(t)$ (bytes), which is proportional to the number of user requests in the t-th time slot. The coefficient L_{c} is a constant, and $\lambda(t)$ is the number of requests generated by all the users in the t-th time slot.

Blockchain Processing Latency

Based on eqs. (4.4.7) and (4.4.8), the blockchain processing latency for the RPoS consensus is given by

$$\tau_{\mathrm{bc},i}(t) = \tau^{\mathrm{g}}_{\mathrm{bc},i}(t) + \tau^{\mathrm{v}}_{\mathrm{bc},i}(t) + \tau^{\mathrm{c}}_{\mathrm{bc},i}(t), \quad \text{(slots)} \tag{4.4.9}$$

where $\tau^{\mathrm{g}}_{\mathrm{bc},i}(t)$, $\tau^{\mathrm{v}}_{\mathrm{bc},i}(t)$ and $\tau^{\mathrm{c}}_{\mathrm{bc},i}(t)$ are the processing latency introduced by the pre-prepare, prepare, and commit steps, respectively. The detailed derivation of $\tau_{\mathrm{bc},i}(t)$ in (4.4.9) which includes the block computing and wireless transmission time amongst all the BSs is given in Appendix A.1.

4.4.4 Security Analysis

The ability of BC-DRL to resist tampering attacks is a crucial aspect of secure MEC service provisioning. To analyze the security performance, we evaluate the minimum time required to tamper the blockchain, which is defined as [Hao *et al.*, 2020]

$$\tau_{\mathrm{tam}}(t) = \frac{N_{\mathrm{B}}}{2}\mathbb{E}_i[\tau_{\mathrm{bc},i}(t)], \quad i \in N_{\mathrm{C}}(t), \tag{4.4.10}$$

where the expectation is taken over the selected committee members in the t-th time slot. This equation reveals that the MEC service provisioning with blockchain has a very high tampering attack-resistant ability, as an attacker would need to compromise at least half of the BSs in the network to succeed in tampering with the data. This robust security feature ensures the integrity and reliability of BC-DRL for MEC services.

Based on our analysis, integrating blockchain into MEC introduces additional latency and computing resources. However, these trade-offs are justified by the benefits, such as tamper attack resistance and steady service provisioning. For example, the RPoS consensus protocol is resource-efficient, distinguishing it from existing blockchain consensus protocols. The dynamic DoS probability constraint ensures security and stability, effectively mitigating potential attacks.

4.5 Constrained DRL Resource Optimization

In this section, we present our MEC computation resource allocation optimization to minimize the overall processing latency for the blockchain and MEC service provisioning subject to constraints on the BS DoS probability. We formulate the optimization problem as a constrained MDP, which is solved using a constrained DRL algorithm. To improve the training efficiency, we reduce the dimension of the neural network's inputs and apply transfer learning to handle changes in the constraints on DoS probability. Lastly, we give the complexity analysis of our proposed constrained

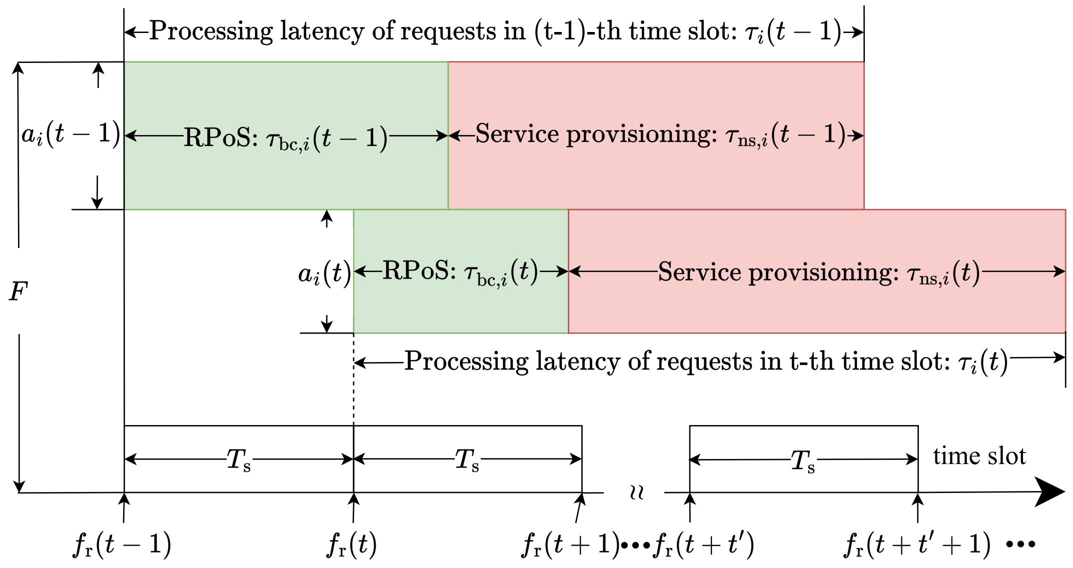

Processing latency of requests in (t-1)-th time slot: $\tau_i(t-1)$
$a_i(t-1)$
RPoS: $\tau_{\mathrm{bc},i}(t-1)$
Service provisioning: $\tau_{\mathrm{ns},i}(t-1)$
F
$a_i(t)$
RPoS: $\tau_{\mathrm{bc},i}(t)$
Service provisioning: $\tau_{\mathrm{ns},i}(t)$
Processing latency of requests in t-th time slot: $\tau_i(t)$
T_s
T_s
T_s
time slot
$f_r(t-1)$
$f_r(t)$
$f_r(t+1)$•••$f_r(t+t')$
$f_r(t+t'+1)$ •••

Figure 4.4: Example of allocated service rates, $a_i(t)$, and the overall processing latency, $\tau_i(t) = \tau_{\mathrm{bc},i}(t) + \tau_{\mathrm{sp},i}(t)$, where F is the total computation capacity of each BS, T_s is the duration of one time slot, and $f_{\mathrm{r}}(t)$ is the total number of requested CPU cycles in the t-th time slot.

DRL algorithm.

4.5.1 BC-DRL Optimization

We aim to minimize the overall processing latency for our BC-DRL framework whilst guaranteeing a given BS DoS probability constraint. The optimization problem is formulated as

$$\begin{aligned} \min_{a_i(t)} \quad & \mathbb{E}[\tau_i(t)] \\ \text{s.t.} \quad & \mathbb{E}[c_i(t)] \leq \epsilon_{\mathrm{DoS}}, \end{aligned} \tag{4.5.1}$$

where $\tau_i(t)$ is expressed in (4.5.4) indicating the overall processing latency for the i-th BS, and $c_i(t)$ expressed in (4.5.5) is the BS DoS probability for the i-th BS indicating that the assigned miner BS does not allocate any resources in the t-th time slot. To analyze the affecting factors of this optimization problem, we can observe from eq. (4.5.1) and Fig. 4.4 that, for any of the miner BS, the overall processing latency depends on the number of requested CPU cycles $f_{\mathrm{r}}(t)$ in each time slot with duration T_{s} and the allocated service rate $a_i(t)$ (CPU cycles/slot). Given the fixed maximum computation capacity of a BS, there is a trade-off between $\tau_i(t)$ and $c_i(t)$. Based on the time-varying $f_{\mathrm{r}}(t)$, we optimize the allocated service rate, $a_i(t)$, to minimize the average processing latency subject to the average DoS probability constraint. In addition, when examining the decentralized blockchain-secured MEC network as a whole, the processing latency and DoS probability of a particular BS vary depending on its frequency of being selected as the miner BS. This relationship is reflected in eq. (4.4.6). As discussed in Section 4.4.2, a larger value of η corresponds to a smaller committee size and a higher frequency of MEC service-provisioning by each committee member meeting the reputation requirement.

Overall Processing Latency

We consider the service arrival process of each user request follows an independent and identically distributed (i.i.d) Bernoulli process. As such, the total number of user requests in each time slot follows a Poisson distribution,and the total number of CPU cycles required to satisfy all user requests in the t-th time slot can be evaluated as

$$f_{\mathrm{r}}(t) = \kappa_{\mathrm{sp}} \cdot \lambda(t) \sum_{u_r=1}^{\lambda(t)} L_u(t), \quad \text{(CPU cycles)} \tag{4.5.2}$$

where κ_1 (CPU cycles/byte) is the coefficient of CPU cycles required for service provisioning, $\lambda(t)$ is the Poisson distribution parameter representing the average number of user requests, and $L_u(t)$ (bytes/request) is the package size of the u-th requesting user. Since the miner BS is designed to provide services to the users, the total CPU cycles required for the service provision BS can be expressed as

$$f_{\mathrm{sp},i}(t) = \mathbb{1}\{i \in \mathcal{N}_{\mathrm{M}}(t)\} \cdot f_{\mathrm{r}}(t). \quad \text{(CPU cycles)} \tag{4.5.3}$$

The overall processing latency corresponds to the summation latency of the processing the blockchain and the MEC service by the miner BS indexed by i, and is given by

$$\tau_i(t) = \tau_{\mathrm{bc},i}(t) + \tau_{\mathrm{sp},i}(t), \quad \text{(slots)} \tag{4.5.4}$$

where $\tau_{\mathrm{bc},i}(t)$ is blockchain processing latency given in eq. (4.4.9) and $\tau_{\mathrm{sp},i}(t) = f_{\mathrm{sp},i}(t)/a_i(t)$ is the processing latency of the MEC service provision, where $a_i(t)$ (CPU cycles/slot) is the service rates allocated in the t-th time slot by the miner BS, referred to the i-th BS.

BS DoS Probability

The instantaneous BS DoS indicator of the i-th BS in the t-th time slot is defined as

$$c_i(t) = \mathbb{1}\{a_i(t) = 0\}, \tag{4.5.5}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function, which equals one when no resources are allocated in the t-th time slot by the i-th BS (i.e., $a_i(t) = 0$), and equals zero otherwise .

4.5.2 Design of Constrained MDP

As shown in Fig. 4.4, the processing latency can be larger than one time slot, the service rates allocated in the current time slot affect the available service rates and the DoS probability in the future time slots, which makes problem (4.5.1) a sequential decision-making problem. Since DRL is well-suited for solving Markovian problems, we resort to reformulating problem (4.5.1) as a constrained MDP.

We first define the action, state, instantaneous reward and cost, and long-term reward and cost.

Action

The action to be taken in the t-th time slot is the service rate of the i-th BS, $a_i(t)$ (CPU cycles/slot), shown in problem (4.5.1). We denote Δf and F as the minimum and maximum service rates that can be allocated. If the requests are denied, $a_i(t) = 0$. Otherwise, $a_i(t)$ could be any value between Δf and F. Thus, the action space can be described as $\mathcal{A} = \{0\} \cup \mathcal{F}$ (CPU cycles/slot), where $\mathcal{F} = [\Delta f, F]$.

State

The required CPU cycles of the requests within each time slot are bounded by $f_{\mathrm{r,max}} = \max\{f_{\mathrm{r}}(t)\}$. Given the minimum service rate, Δf, the maximum processing latency can be expressed as

$$T_{\max} = \frac{f_{\mathrm{r,max}}}{\Delta f}. \tag{4.5.6}$$

We design the state of the i-th BS includes the states of all the service rates allocated in the past $T_{\max}$ time slots and can be described as

$$\begin{aligned} \boldsymbol{s}_i(t) &= \{\hat{\boldsymbol{s}}_i(t,t'), t' \in [t - T_{\max}, t]\} \\ &= \{\hat{\boldsymbol{s}}_i(t, t - T_{\max}), \cdots, \hat{\boldsymbol{s}}_i(t-1, t), \hat{\boldsymbol{s}}_i(t,t)\}, \end{aligned} \tag{4.5.7}$$

where $\hat{\boldsymbol{s}}_i(t,t') = [\hat{\tau}_i(t,t'), \hat{a}_i(t,t')]$ is the state of the service rates allocated in the t'-th time slot, which is composed of the remaining processing latency and the service rates allocated in the t'-th time slot (See Appendix A.2). We denote t as the current time slot, and t' as the time slot that the service rate allocated, respectively. We note that both t and t' are integers, and $0 \le t' \le t$. Therefore, the state space can be described as $\mathcal{S} = \{\boldsymbol{s}_i(t), t \in [0, T_{\max}]\}$.

Instantaneous Reward and Cost

The instantaneous reward is defined as

$$r_i(t) = \begin{cases} 0, & \text{if } a_i(t) = 0 \\ -r_i^{\mathrm{a}}(t), & \text{if } a_i(t) \neq 0 \end{cases} \tag{4.5.8}$$

where $r_i^{\mathrm{a}}(t) = \tau_i(t)/\tau_{\max}$ is the normalized processing latency when service rate is allocated, and $\tau_{\max}$ is the maximum value of $\tau_i(t)$ in (4.5.4). We denote $\overline{L}_{\mathrm{b}} = L_{head} + L_{\mathrm{c}}\bar{\lambda}_{req}$ as the average size of the blocks, $\bar{\lambda}_{req}$ as the average number of arrived requests, and $\overline{L}_{u_{\mathrm{r}}}$ as the average size of the requests in each time slot.

The instantaneous cost function is the BS DoS indicator function in the t-th time slot which is defined in (4.5.5).

Long-Term Reward and Cost

Given a policy $\mu(\hat{s}_i(t))$, the long-term discounted reward, representing the normalized processing latency when computing resources are allocated, is defined as

$$R_{i,\mu}(t) = \mathbb{E}_\mu \left[\sum_{\hat{t}=t}^{\infty} \gamma_r^{\hat{t}-t} r_i(t) \right], \tag{4.5.9}$$

where γ_r is the reward discount factor. The long-term discounted cost, representing the long-term DoS probability, is defined as

$$C_{i,\mu}(t) = \mathbb{E}_\mu \left[\sum_{\hat{t}=t}^{\infty} \gamma_c^{\hat{t}-t} c_i(t) \right] = \frac{\mathbb{E}_\mu[c_i(t)]}{1-\gamma_c}, \tag{4.5.10}$$

where γ_c is the cost discount factor. To guarantee the requirement on the DoS probability, the long-term cost should satisfy the following constraint

$$C_{i,\mu}(t) \leq \mathcal{E}_{\max}, \tag{4.5.11}$$

where $\mathcal{E}_{\max} = \epsilon_{\mathrm{DoS}}/(1-\gamma_c)$ is the maximum long-term DoS probability, and ϵ_{DoS} denotes the required threshold of the instantaneous DoS probability.

4.5.3 Reformulated Constrained MDP Problem

Based on the above constrained MDP design parameters, we can reformulate problem (4.5.1) (See Appendix A.3 for the proof of equivalence of problems (4.5.1) and (4.5.12).) as a constrained MDP given by

$$\begin{aligned} \max_{\mu(\cdot)} \quad & R_{i,\mu}(t) \\ \text{s.t.} \quad & C_{i,\mu}(t) \leq \mathcal{E}_{\max}. \end{aligned} \tag{4.5.12}$$

To solve problem (4.5.12), we utilize a constrained DRL algorithm in which the policy, $\mu(\cdot)$, and the dual variable, $\lambda_{\mathcal{L}}$, are updated iteratively. The Lagrangian function of problem (4.5.12) is given by

$$\mathcal{L}_{i,t}\left(\mu(\cdot), \lambda_{\mathcal{L}}\right) = R_{i,\mu}(t) - \lambda_{\mathcal{L}}\left(C_{i,\mu}(t) - \mathcal{E}_{\max}\right), \tag{4.5.13}$$

where $\lambda_{\mathcal{L}}$ is the Lagrangian dual variable. Problem (4.5.12) can be converted to the following unconstrained problem

$$(\mu^*(\cdot), \lambda_{\mathcal{L}}^*) = \arg\min_{\lambda_{\mathcal{L}} \geq 0} \max_{\mu(\cdot)} \mathcal{L}_{i,t}(\mu(\cdot), \lambda_{\mathcal{L}}), \tag{4.5.14}$$

where $\mu^*(\cdot)$ and $\lambda_{\mathcal{L}}^*$ indicate the optimal policy and the optimal Lagrangian dual variable, respectively. To apply the constrained DRL algorithm, we further verify that the formulated problem in eq. (4.5.12) satisfies the Markov property [Sutton and Barto, 2016] (See proof in Appendix A.4).

4.5.4 Service Rate Allocation

To solve the coupled MDP problem formulated in eq. (4.5.12), we decouple this problem by utilizing the PD-DDPG algorithm to find the optimal primal-dual solution [Liang *et al.*, 2018]. We define the reward critic Q-network as $Q_R(\widetilde{\boldsymbol{s}}, a|\theta_R)$, the cost critic Q-network as $Q_C(\widetilde{\boldsymbol{s}}, a|\theta_C)$, the actor network as $\mu(\widetilde{\boldsymbol{s}}|\theta_\mu)$, respectively. The corresponding target critic networks are $Q'_R(\widetilde{\boldsymbol{s}}, a|\theta'_R)$, $Q'_C(\widetilde{\boldsymbol{s}}, a|\theta'_C)$, and $\mu'(\widetilde{\boldsymbol{s}}|\theta'_\mu)$. The experience replay memory buffer as $\mathcal{B}$. The specific algorithm is detailed in Algorithm 2[1].

Algorithm 2: Dynamic Resource Allocation Algorithm

```
1  Randomly initialize $Q_R(\widetilde{\boldsymbol{s}}, a|\theta_R)$, $Q_C(\widetilde{\boldsymbol{s}}, a|\theta_C)$, and $\mu(\widetilde{\boldsymbol{s}}|\theta_\mu)$. Initialize
     $\theta'_R \leftarrow \theta_R$, $\theta'_C \leftarrow \theta_C$, and $\theta'_\mu \leftarrow \theta_\mu$, $\lambda_\mathcal{L}$, $\epsilon_{\text{DoS}}$, $\mathcal{B}$, $\gamma_r$, and $\gamma_c$.
2  Select action based on the current policy $\theta_\mu$ and the exploration noise $\mathcal{N}_a(t)$
     as: $a(t) = \mu(\widetilde{\boldsymbol{s}}(t)|\theta_\mu) + \mathcal{N}_a(t)$.
3  Execute action $a_i(t)$ then observe reward $r(t)$, cost $c(t)$, and new state
     $\widetilde{\boldsymbol{s}}(t+1)$.
4  Store transition $\langle \widetilde{\boldsymbol{s}}(t), a(t), r(t), c(t), \widetilde{\boldsymbol{s}}(t+1)\rangle$ into $\mathcal{B}$.
5  Randomly sample a mini-batch of $M$ transition from $\mathcal{B}$:
     $\{\langle \widetilde{\boldsymbol{s}}_m, a_m, r_m, c_m, \widetilde{\boldsymbol{s}}_{m+1}\rangle, m = 1, 2, \cdots, M\}$.
6  Set Q-targets for reward and cost temporal difference (TD):
     $y_m^R = r_m + \gamma_r Q'_R\left(\widetilde{\boldsymbol{s}}_{m+1}, \mu'(\widetilde{\boldsymbol{s}}_{m+1}|\theta'_\mu)|\theta'_R\right)$,
     $y_m^C = c_m + \gamma_c Q'_C\left(\widetilde{\boldsymbol{s}}_{m+1}, \mu'(\widetilde{\boldsymbol{s}}_{m+1}|\theta'_\mu)|\theta'_C\right)$.
7  Update reward and cost critic Q-networks by minimizing the losses denoted
     by mean squared TD errors:
     $\Delta_R = \frac{1}{M}\sum_m \left(y_m^R - Q_R(\widetilde{\boldsymbol{s}}_m, a_m|\theta_R)\right)^2, \quad \Delta_C = \frac{1}{M}\sum_m \left(y_m^C - Q_C(\widetilde{\boldsymbol{s}}_m, a_m|\theta_C)\right)^2.$
8  Update the actor policy $\theta_\mu$, in the primal domain, with sampled policy
     gradient descent:
     $\nabla_{\theta_\mu}\mathcal{L} = \frac{1}{M}\sum_m \nabla_{\theta_\mu}(Q_R(\widetilde{\boldsymbol{s}}_m, \mu(\widetilde{\boldsymbol{s}}_m|\theta_\mu)|\theta_R) - \lambda_\mathcal{L} Q_C(\widetilde{\boldsymbol{s}}_m, \mu(\widetilde{\boldsymbol{s}}_m|\theta_\mu)|\theta_C))$.
9  Calculate the gradient of dual variable $\lambda_\mathcal{L}$:
     $\nabla_{\lambda_\mathcal{L}}\mathcal{L} = \frac{1}{M}\sum_m \left(Q_C(\widetilde{\boldsymbol{s}}_m, \mu(\widetilde{\boldsymbol{s}}_m|\theta_\mu)|\theta_C) - \mathcal{E}_{\max}\right)$.
10 Update the dual variable, $\lambda_\mathcal{L}$, in the dual domain, with sampled dual
     gradient ascent: $\lambda_\mathcal{L} \leftarrow \max\{0, \lambda_\mathcal{L} + \beta_{\lambda_\mathcal{L}} \nabla_{\lambda_\mathcal{L}}\mathcal{L}(\theta_\mu, \lambda_\mathcal{L})\}$.
11 Update target networks with $\varphi$:
     $\theta'_R \leftarrow \varphi\theta_R + (1-\varphi)\theta'_R, \quad \theta'_C \leftarrow \varphi\theta_C + (1-\varphi)\theta'_C, \quad \theta'_\mu \leftarrow \varphi\theta_\mu + (1-\varphi)\theta'_\mu$.
```

We consider that the agent follows a deterministic policy denoted by $\mu : a(t) =$

[1]Since the steps always refer to the i-th BS, we do not explicitly denote "i" in this algorithm.

$\mu(\boldsymbol{s}(t)|\theta_\mu)$. It is a neural network that determines the service rates allocation based on the state in each time slot, where θ_μ represents the parameter of the neural network.

To improve the training efficiency of the constrained DRL algorithm, we propose to reduce the dimension of the state space of the constrained MDP defined in eq. (4.5.7) by extracting key features to achieve a lower dimension state for training. Specifically, we extract remaining processing latency, $\hat{\tau}_i(t,t')$, of all the allocated service rates in the i-th BS to be a normalized sum, i.e.,

$$\rho_i(t) = \frac{\tau_{i,\min}(t)}{\tau_{\max}} = \frac{\frac{1}{F}\left(\sum_{t'=t-T_{\max}}^{t} \hat{\tau}_i(t,t')\right)}{\tau_{\max}}, \tag{4.5.15}$$

where $\tau_{i,\min}(t)$ is the remaining processing latency of all the allocated service rates if they are processed with the maximum service rate, F. Based on eq. (4.5.15), the lower dimension state is re-designed as

$$\widetilde{\boldsymbol{s}}_i(t) = \left\{\frac{f_{i,\mathrm{a}}(t)}{F}, \rho_i(t)\right\}, \tag{4.5.16}$$

where $f_{i,\mathrm{a}}(t)$ is the available service rates of the i-th BS in the t-th time slot.

We further apply transfer learning to reduce the training time due to changes in the DoS probability constraints. To do so, we reuse the parameters of well-trained neural networks as the initial parameters for the target neural network.

4.5.5 Computational Complexity Analysis

The computational complexity of the proposed constrained DDPG composes the inference complexity of three neural networks denoted by $Q_R(\widetilde{\boldsymbol{s}}, a|\theta_R)$, $Q_C(\widetilde{\boldsymbol{s}}, a|\theta_C)$, and $\mu(\widetilde{\boldsymbol{s}}|\theta_\mu)$, respectively. Thus, the computational complexity of the proposed constrained DDPG algorithm is

$$O_{\mathrm{PRO}} = O(N_{\mathrm{R}} + N_{\mathrm{C}} + N_\mu), \tag{4.5.17}$$

where N_{R}, N_{C}, and N_{μ} are the number of multiplications required to process the three neural networks, and are given by $N_{\mathrm{R}} = \sum_{L_{\mathrm{R}}=1}^{L_{\mathrm{R}}} n_{L_{\mathrm{R}}} n_{L_{\mathrm{R}}+1}$, $N_{\mathrm{C}} = \sum_{L_{\mathrm{C}}=1}^{L_{\mathrm{C}}} n_{L_{\mathrm{C}}} n_{L_{\mathrm{C}}+1}$, and $N_{\mu} = \sum_{L_{\mu}=1}^{L_{\mu}} n_{L_{\mu}} n_{L_{\mu}+1} + \Omega_{\mathrm{Sig}}$, respectively, where L_{R}, L_{C}, and L_{μ} denote the number of layers in the neural networks; $n_{L_{\mathrm{R}}}$, $n_{L_{\mathrm{C}}}$, and $n_{L_{\mu}}$ are the number of neurons in the L_{R}-th, L_{C}-th, and L_{μ}-th layer, and Ω_{Sig} represents the number of multiplications required to compute the `Sigmoid` function. We note that the computation complexity of `ReLU` function is ignored since it's very low compared with the other operations.

Another commonly used method for solving sequential decision-making problems is dynamic programming, whose computational complexity is given by

$$O_{\mathrm{DP}} = O(|\mathcal{A}| \times |\mathcal{S}| \times N_{\mathrm{iter}}), \tag{4.5.18}$$

where $|\mathcal{A}|$ and $|\mathcal{S}|$ represent the size of the action and state spaces given in Section 4.5.2, and N_{iter} is the number of iterations required for the dynamic programming algorithm to converge. Since our problem involves continuous action space, where the action can take any value ranging from f to F or equal 0, $|\mathcal{A}|$ approaches infinity, resulting in O_{DP} also approaching infinity. Consequently, conventional dynamic programming is not a feasible option for solving our problem shown in eq. (4.5.18).

4.6 Simulations and Empirical Convergence Analysis

4.6.1 Simulation Setup

We present numerical simulations to evaluate the performance and analyze the empirical convergence of our proposed BC-DRL solution using Google TensorFlow embedded in a Python platform. We consider a total of 10 BSs that are initialized

Table 4.2: Chapter 4 Key Simulation Parameters

Simulation parameters	**Values**
Number of overall BSs in the network N_{B}	10
Computation capacity of the BSs F	1.6 G CPU cycles/slot
Minimum service rate can be allocated Δf	0.01 G CPU cycles/slot
Request arrival rate λ_{req}	Poisson(10^3)
Size of request by the u-th user $L_u(t)$	Uniform$(1, 10)$ KB
Coefficient size of user requests L_{c}	8 bytes
Prior probability of event $e_i(t)$	0.8
Weight coefficient of reputation threshold η	1
Coefficient of reputation ϑ_{I}	0.2
Coefficient of CPU cycles for blockchain κ_{bc}	0.001 G [Guo *et al.*, 2018]
Coefficient of CPU cycles for services κ_{sp}	330 [Dong *et al.*, 2019]
Threshold of DoS probability ϵ_{DoS}	2 %

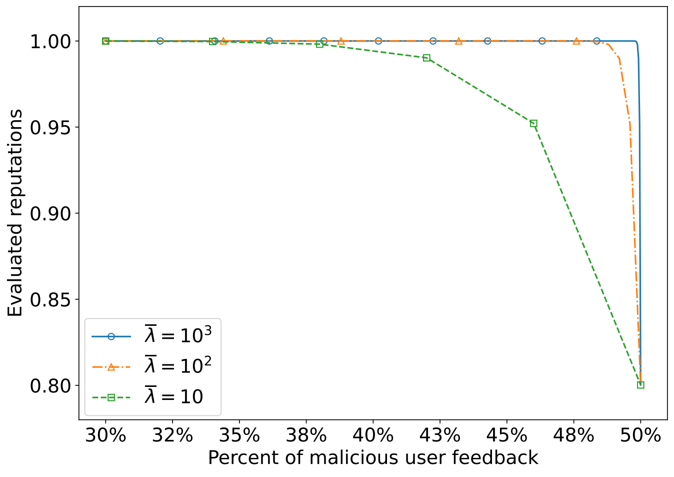
as non-malicious and have the maximum reputations equal to one. The transmission rates among all the BSs are assumed to be W=10Gbps [Sun *et al.*, 2021]. The action exploration noise in our DRL simulations follows an Ornstein-Uhlenbeck process [Lillicrap *et al.*, 2016]. We note that the service rates of the committee BSs are all considered equal to $a_i(t)$ in the t-th time slot, since the serving BS is randomly assigned from the committee in each time slot. Unless otherwise mentioned, the simulation parameters are summarized in Table 4.2[2].

4.6.2 Simulation Results

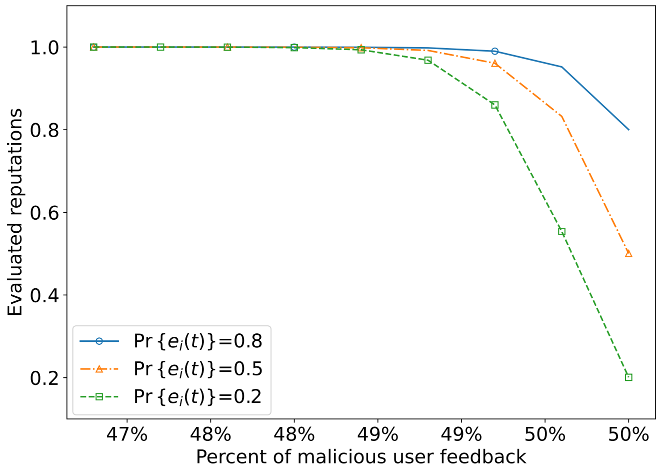
RPoS-Based Blockchain Management

We present simulations of the reputation evaluation and performance of our RPoS consensus protocol.

[2]For simulation simplicity, we set $\eta = 1$ in our simulations. However, exploring the effects of scalable η configurations could provide valuable insights.



(a) Reputations evaluated with different $\bar{\lambda}_{req}$ when $\Pr\{e_i(t)\} = 0.8$.



(b) Reputations evaluated with different $\Pr\{e_i(t)\}$, when $\bar{\lambda}_{req} = 100$.

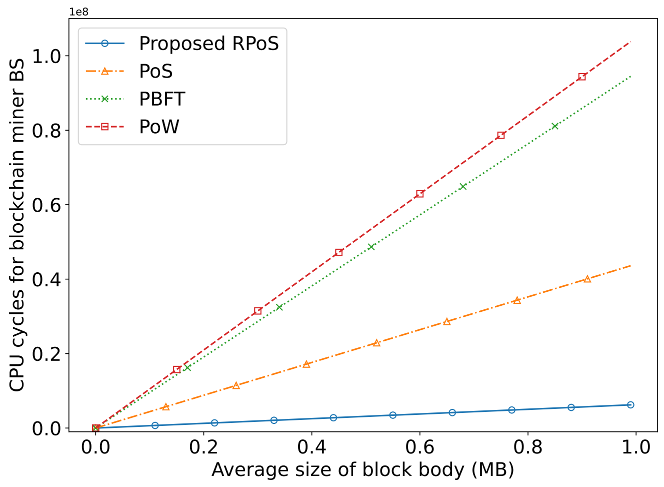
Figure 4.5: Evaluated BS reputations under malicious user feedback attacks.

Fig. 4.5 explores the impact brought by malicious user feedback attacks discussed in Section 4.3.1 by evaluating the reputations evaluated based on Bayesian inference when the percentage of feedback from malicious users increases. We see in Figs. 4.5 (a) and (b) that the evaluated reputations decrease with increasing malicious user feedback. Fig. 4.5(a) shows that our Bayesian-based reputation evaluation with a larger user request rate, $\bar{\lambda}_{req}$, can accurately maintain the BS reputations to be equal to 1 even with a high percentage of malicious user feedback that is close to 50%. In Fig. 4.5(b), we see that a higher prior probability of requests being served by the BSs, $\Pr\{e_i(t)\}$, corresponds to a higher resistance ability to malicious user feedback attacks due to a higher confidence in the reputation evaluation of the BSs.

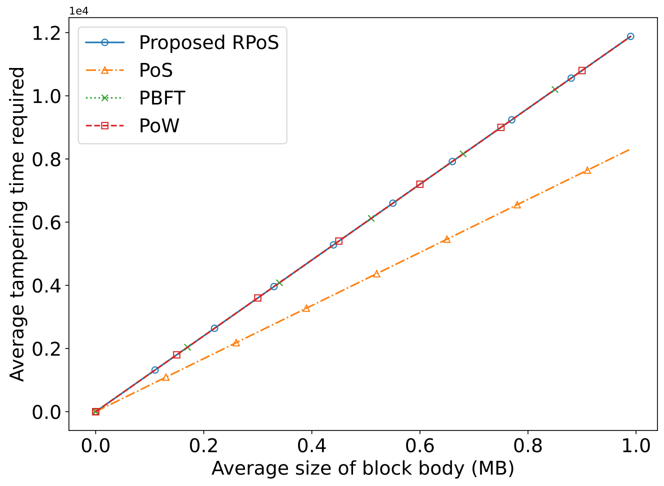
Fig. 4.6 shows the computation resource consumption and tampering resistant ability of different consensus when there are three malicious BSs. We can observe from Fig. 4.6(a) that the proposed RPoS requires significantly lower CPU cycles compared to other benchmark consensus protocols. Fig. 4.6(b) shows the required time to tamper the blockchain. This observation, combined with the findings presented in Fig. 4.6(a), confirms that PoS is vulnerable to tampering attacks despite its resource efficiency. In contrast, the proposed RPoS consensus exhibits robust resistance to tampering attacks, comparable to the security levels achieved by PoW, while significantly reducing computation resource requirements. We can also observe from Fig. 4.6 that both the CPU cycles for blockchain miner BS and tampering time increase linearly with the increasing average size of the block body. This linear relationship is attributed to the proportionality of these metrics to the block size, primarily determined by the block body size.

DRL-Based Resource Allocation

We compare the performance of our proposed algorithm with other benchmark dynamic resource allocation algorithms from the perspective of minimizing the processing latency and DoS probability, and improving the training efficiency. A user feedback value of “1” either indicates a BS refusing to provide services or malicious



(a) CPU cycles required by the blockchain miner BS. A lower CPU cycle requirement corresponds to a higher resource efficiency and lower BS DoS probability.



(b) Tampering time for the optimized parameters of the neural networks with different system management scenarios.

Figure 4.6: Computation resource consumption and tampering resistant ability for different blockchain consensus protocols.

users sending incorrect feedback when service rates are allocated. Unless otherwise mentioned, the hyper-parameters are summarized in Table 4.3.

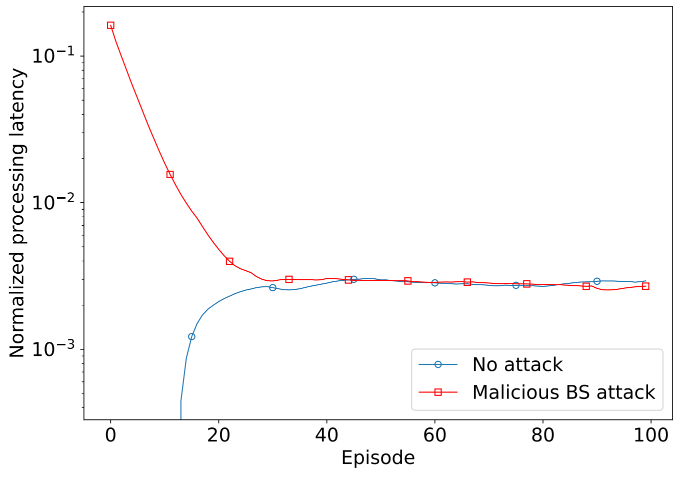
Fig. 4.7 shows the training results of BC-DRL with and without malicious BS

Table 4.3: Hyper-Parameters of Constrained DRL Algorithm

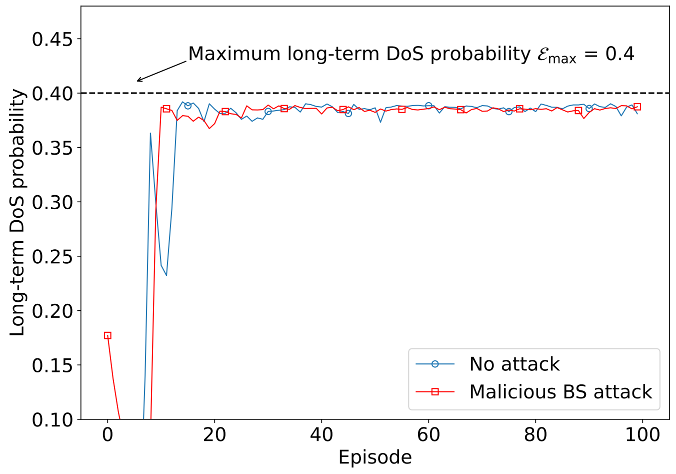
Simulation parameters	**Values**
Discount factors of reward and cost γ_r , γ_c	0.95
Learning rates of critic NNs β_r, β_c	5×10^{-4}
Learning rate of actor NN β_a	2×10^{-4}
Learning rate of dual variable $\beta_{\lambda_{\mathcal{L}}}$	0.1
Mini-batch size M	512
Target NNs updating rate φ	5×10^{-3}
Max replay buffer size	2×10^{5}
Number of steps in each episode N_{step}	1000

attacks. To investigate the impact of malicious BS attacks on processing latency and DoS probability, we include three malicious BSs in our simulation. Each BS randomly denies service requests from users following the Bernoulli process. We can observe that both the processing latency and the DoS probability converged to steady values after approximately 15 episodes. We observe that while BC-DRL achieves approximately the same reward and cost performance for both scenarios of with and without malicious BSs, the dual variables for the optimization converge to different values in each scenario to accurately balance the trade-off between processing latency and DoS probability.

Fig. 4.8 shows the training results of our proposed BC-DRL solution and a benchmark PD-DDPG resource allocation algorithm with PoS consensus. The comparison between the RPoS and PoS consensus protocols shows that both protocols can satisfy the constraint requirements. However, the RPoS outperforms PoS in terms of achieving a lower processing latency. The superiority of RPoS can be attributed to its ability to allocate more computing resources to provide MEC services. This advantage is due to the random selection of the BS from the committee in the RPoS



(a) Normalized processing latency when resources are allocated.

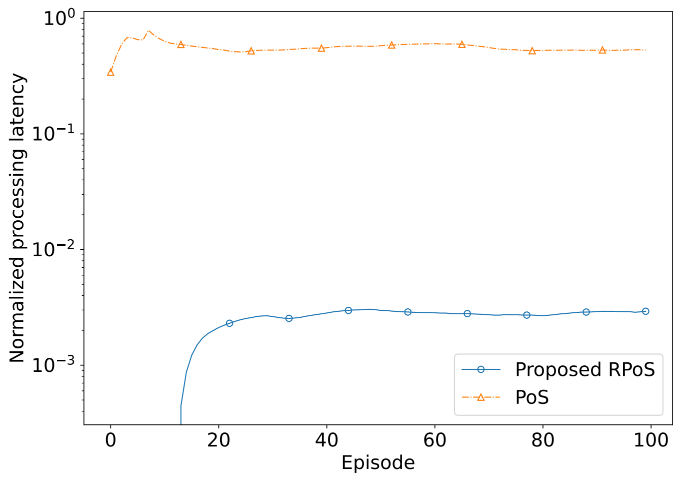


(b) Average DoS probability.

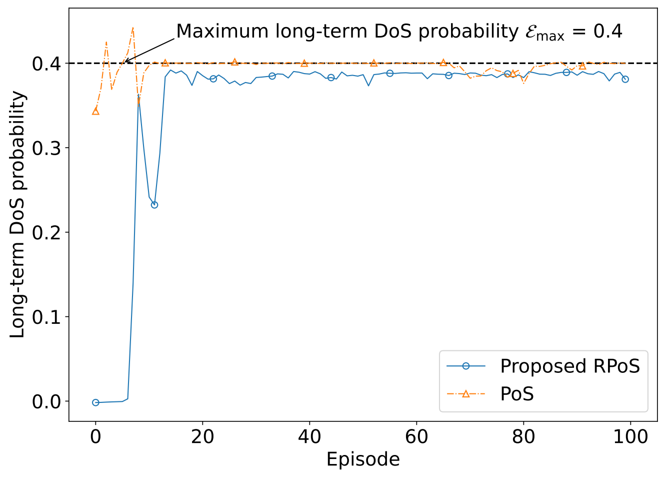
Figure 4.7: Performances of BC-DRL solution with and without malicious BS attacks.

consensus protocol, allowing for a more balanced distribution of resources among the participating BSs. In contrast, the PoS protocol consistently chooses the same BS, which may result in uneven resource allocation and higher processing latency.

In Fig. 4.9, we compare our constrained DDPG DRL solution with benchmark



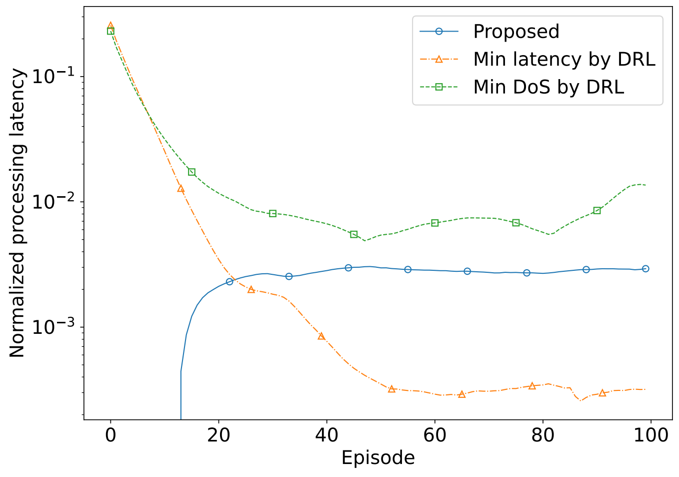
(a) Normalized processing latency when resources are allocated.



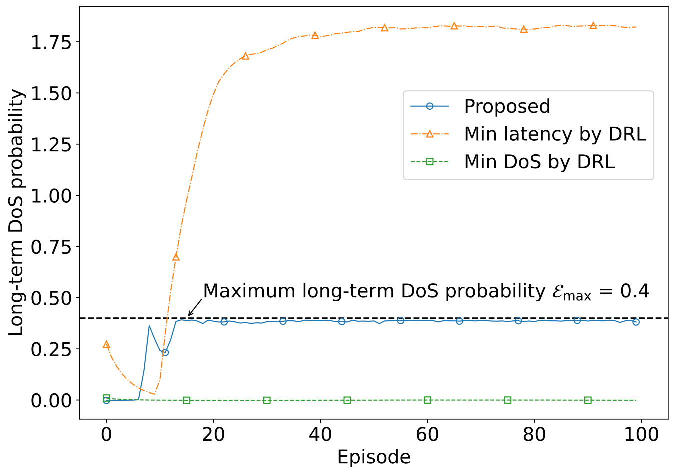
(b) Average DoS probability in the long term.

Figure 4.8: Performance of processing latency and DoS probability, when resource allocated by the proposed constrained DDPG algorithm.

unconstrained DDPG DRL solutions aimed at minimizing either the processing latency or DoS probability. Figs. 4.9(a) and 4.9(b) highlight a fundamental performance trade-off where the min latency solution leads to an intolerable DoS probability, whilst



(a) Normalized processing latency when resources are allocated.



(b) Average DoS probability in the long-term.

Figure 4.9: Processing latency and DoS probability when resources are allocated with different DRL algorithms. Our proposed algorithm achieves minimum processing latency with a satisfactory DoS probability.

the min DoS probability solution results in a lengthy processing latency and an unnecessarily low DoS probability. In contrast, our proposed constrained DRL solution can achieve a significantly reduced processing latency while maintaining a satisfactory DoS probability, as determined by the specified maximum long-term cost.

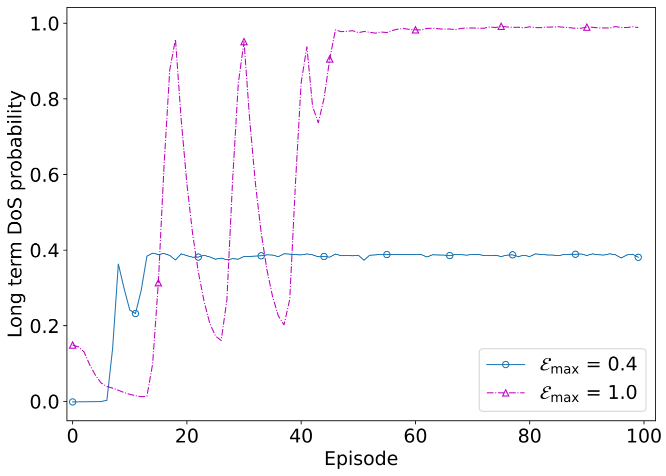


Figure 4.10: Training results of long-term DoS probability achieved by random initialization.

4.6.3 Empirical Convergence Analysis

Empirical convergence analysis of our proposed constrained DRL is provided since the convergence of the proposed DRL is highly affected by the iterative updates and interactions with the MEC network. Specifically, we focus on analyzing the convergence ability with dynamic values of the DoS probability constraint, $\mathcal{E}_{\max}$.

Fig. 4.10 shows the training results of long-term DoS probability achieved by random initialization when the constraints on the DoS probability equal 0.4 and 1.0, respectively. This figure shows that both DoS probabilities converge to stable values with the increasing number of training episodes, indicating that the proposed algorithm is learning and optimizing the resource allocation efficiently. We can also observe from this figure that it takes approximately 15 episodes to converge when the constraint on the DoS probability is 0.4, whilst the number of episodes before convergence increases to approximately 47 when $\mathcal{E}_{\max}$ increases to 1.0. This phenomenon indicates that the difference in the constraint on DoS probability significantly affects

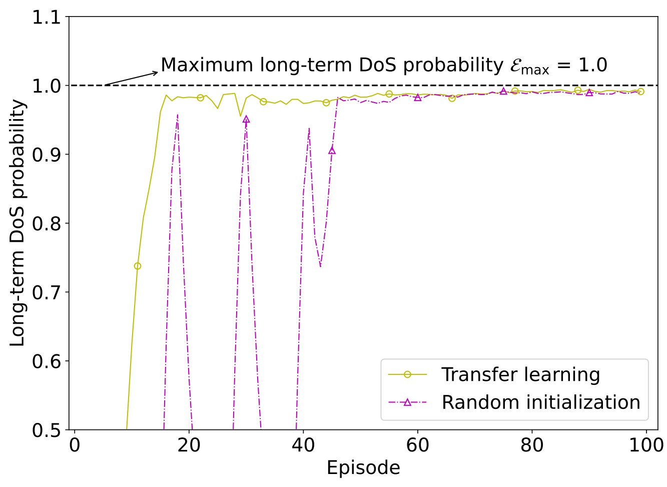


Figure 4.11: Average DoS probability with random initialization and transfer learning when DoS probability constraint in the long-term equals one.

the convergence speed of the proposed algorithm.

Fig. 4.11 plots the training results of transfer learning and random initialization when the maximum long-term DoS probability, $\mathcal{E}_{\max}$, equals 1. For transfer learning, we initialize the neural network with the well-optimized parameters of the neural network when $\mathcal{E}_{\max} = 0.4$. For the random initialization benchmark, we train the neural network from scratch. We can observe from Fig. 4.11 that using transfer learning helps the DRL training to converge at approximately 16 episodes, whilst it takes approximately 47 episodes for random initialization to converge to approximately the same value. This is because when the required constraint on DoS probability changes to a new value, the newly updated optimization problem is still related to the previous scenario. Therefore, some of the hidden features that have been well-trained in the previous scenario are still effective to be applied in the new scenario, which further reduces the required training epochs for convergence.

Fig. 4.12 presents the trade-off between the processing latency and the long-term

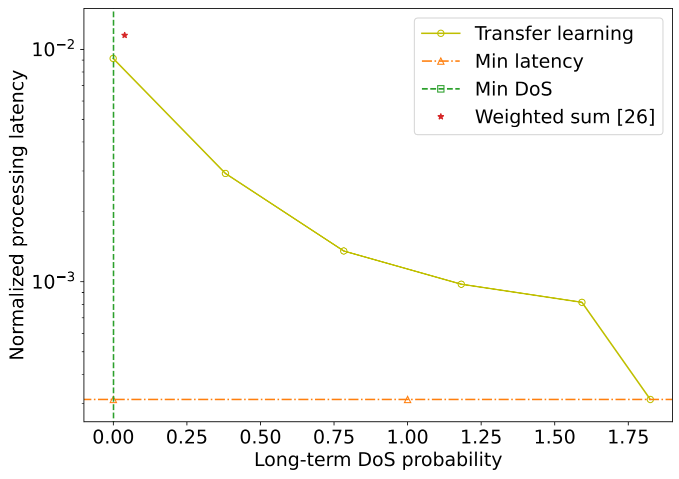


Figure 4.12: The trade-offs between the processing latency and the long-term DoS probability.

DoS probability achieved through transfer learning. We include a benchmark obtained from a prior study [Guo *et al.*, 2018], where the authors applied unconstrained DRL to maximize the weighted sum of the blockchain throughput and the reciprocal of MEC delay, with equal weighting coefficients of 0.5. It is important to note that the weighted sum benchmark relies on manually selected weighting coefficients and lacks the inherent capability to handle the dynamic constraint. In contrast, our constrained DRL can dynamically update the processing latency in response to changes in the DoS probability constraint. Interestingly, the trade-off figure achieved by transfer learning exhibits some non-smooth behavior. This phenomenon highlights that the converged values achieved by a DRL algorithm are significantly affected by the initialization values.

4.7 Chapter Summary

We developed a blockchain-secured deep reinforcement learning (BC-DRL) framework for efficient resource allocation in dynamic environments. The BC-DRL framework introduced a low-latency reputation-based proof-of-stake (RPoS) blockchain consensus protocol to select trusted base stations (BSs) and resist attacks from both BSs and users. We formulated the resource optimization problem as a Markov decision process (MDP) that balances processing latency and BS DoS probability. To address the challenge of high-dimensional inputs, we designed a constrained deep reinforcement learning (DRL) algorithm to solve the formulated constrained MDP. Numerical experiments and analysis showed that the proposed BC-DRL solution requires approximately 2.5 times less CPU cycles compared with PoW, and can find the optimized resource allocation policy while satisfying the given quality-of-service constraints compared with existing unconstrained DRL-based resource allocation policies.

Chapter 5

Hybrid-Task Meta-Learning: A GNN for Scalable and Transferable Bandwidth Allocation

In this chapter, we develop a deep learning-based bandwidth allocation policy that is: 1) scalable with the number of users and 2) transferable to different communication scenarios, such as non-stationary wireless channels, different QoS requirements, and dynamically available resources. To support scalability, the bandwidth allocation policy is represented by a GNN, with which the number of training parameters does not change with the number of users. To enable the generalization of the GNN, we develop a hybrid-task meta-learning (HML) algorithm that trains the initial parameters of the GNN with different communication scenarios during meta-training. Next, during meta-testing, a few samples are used to fine-tune the GNN with unseen communication scenarios. Simulation results demonstrate that our HML approach can improve the initial performance by 8.79%*, and sampling efficiency by* 73%*, compared with existing benchmarks. After fine-tuning, our near-optimal GNN-based policy can achieve close to the same reward with much lower inference complexity compared to the optimal policy obtained using iterative optimization.*

5.1 Chapter Introduction

Throughout the rapid evolution of wireless communication systems, the spectral efficiency, which is the amount of information that can be transmitted over a given bandwidth while maintaining a certain quality of service (QoS) level, still remains one of the most critical performance metrics for future sixth-generation (6G) wireless communications [Chowdhury *et al.*, 2020; Hao *et al.*, 2023]. To maximize spectrum efficiency, low-complexity bandwidth allocation solutions are critical for real-time decision-making within each transmission time interval (TTI) that could be shorter than one millisecond in current fifth-generation (5G) wireless communications. Furthermore, the number of users requesting bandwidth in each TTI is stochastic [Gu *et al.*, 2023; Guo and Yang, 2022], each user may have different QoS requirements [Han *et al.*, 2020; Mohammady *et al.*, 2014; Zanzi *et al.*, 2021], and wireless channels are non-stationary [Yuan *et al.*, 2021; Zhang *et al.*, 2022], making it difficult to develop a low-complexity bandwidth allocation policy that is scalable with the number of users and can satisfy a diverse range of communication scenarios.

Existing iterative optimization algorithms can obtain optimal bandwidth allocation policies, but their computational complexity is generally too high to be implemented in real time [Dong *et al.*, 2021; Lee *et al.*, 2023; Xu *et al.*, 2023b]. To reduce the computational complexity, deep learning is a promising approach for 6G communications [Letaief *et al.*, 2022; She *et al.*, 2021]. The idea is to train a deep neural network that maps the network status to the optimal decision. After training, the deep neural network can be used in communication systems for real-time decision-making, referred to as inference [He *et al.*, 2019]. Although deep learning has much lower inference complexity compared with iterative optimization algorithms, existing deep learning solutions using fully connected neural networks (FNNs) are not scalable to different number of users in wireless networks [Sun *et al.*, 2023]. This is because the number of training parameters of an FNN depends on the dimensions of the input and output, which change with the number of users. Thus, a well-trained FNN is not applicable in wireless networks with stochastic user requests. In contrast to FNNs, graph neural networks (GNNs) have scalable numbers of training parameters that adapt to the number of users [Gilmer *et al.*, 2017] — making them highly-suitable for developing scalable deep learning-based resource allocation solutions for wireless networks [Liu *et al.*, 2022; Shen *et al.*, 2021]. Furthermore, improving the generalization ability of GNN in wireless networks with diverse QoS requirements remains an open problem.

A key 5G application that requires flexible resource allocation solutions is network slicing, where resources from a shared physical infrastructure is partitioned into distinct network slices supporting diverse QoS requirements, such as data rate [Guo and Yang, 2023; Guo *et al.*, 2019], latency [Tang and Zhang, 2007; Wu *et al.*, 2003], and security [Liu *et al.*, 2019a; Wang *et al.*, 2020; Yang *et al.*, 2021; Yu *et al.*, 2016], in both long and short coding blocklength regimes [Alsenwi *et al.*, 2021; Li *et al.*, 2022; Polyanskiy *et al.*, 2010]. To reserve resources for a single slice, the authors of [Sun *et al.*, 2019d] proposed to compute the weights of different slices based on the corresponding QoS requirements and the number of service requests. With this approach, the amount of reserved resources for each slice is stochastic. Meanwhile, since the wireless channels are non-stationary, the reserved resources and the wireless channels in the training stage could be different from the actual required resources in the testing stage [Do *et al.*, 2017; Lu *et al.*, 2021]. As such, the mismatch between training data samples and testing data samples remains a crucial bottleneck for implementing efficient learning-based policies in practical wireless networks.

Recent works have proposed to reduce the online training time by transfer learning, which involves offline pre-training and online fine-tuning [Dong *et al.*, 2021]. This method effectively reuses previously well-trained neural network features and significantly improves the sample efficiency. To further improve the online training efficiency for unseen tasks, meta-learning has been proposed [Andrychowicz *et al.*, 2016; Finn *et al.*, 2017; Nichol *et al.*, 2016; Raghu *et al.*, 2020]. One of the meta-learning algorithms, model-agnostic meta-learning (MAML), has been applied to solve policy mismatch issues caused by varying user requests and non-stationary wireless channels [Huang *et al.*, 2021a; Wang *et al.*, 2022b; Yuan *et al.*, 2021; Zhang *et al.*, 2022]. While these aforementioned works have highlighted the generalization ability of meta-learning for non-stationary wireless resource allocation, no works have addressed the impact of diverse QoS requirements in different communication scenarios.

In this paper, we put forth a low-complexity bandwidth allocation framework by designing a GNN that is scalable with the number of users and applying meta-learning to generalize the GNN to different communication scenarios. In our simulations, the gap between the sum reward achieved by the GNN-based policy and that of the optimal bandwidth allocation policy obtained from the iterative optimization algorithm is less than 6%. HML also improves the initial performance by up to 8.79% and sample efficiency by up to 73% compared with the MAML benchmark. We also show that the performance gains of HML is even higher when compared to the other two benchmarks.

5.2 Related Works

5.2.1 Intelligent Resource Allocation

Applying deep learning for resource allocation in wireless networks has been widely studied in the existing literature [He *et al.*, 2019; Sun *et al.*, 2023]. In [He *et al.*, 2019], the authors showed that learning-based algorithms could obtain near-optimal solutions, and the computational complexity in inference is low. In [Sun *et al.*, 2023], the authors proposed a FNN-based unsupervised learning algorithm to optimize the bandwidth allocation policy. More recently, due to the fact that FNN is not scalable to the number of users, GNNs have been applied in wireless networks optimizations [Liu *et al.*, 2022; Shen *et al.*, 2021]. In [Liu *et al.*, 2022], the authors designed a GNN, which is scalable to the number of users in a wireless network, to minimize the summation of queuing delay violation probability and packet loss probability. In [Shen *et al.*, 2021], the authors developed GNN-based scalable learning-based methods to solve radio resource management problems.

5.2.2 Generalization of Intelligent Resource Allocation

In wireless networks, the user requests, wireless channels, and available resources for each type of service can be non-stationary. Table 5.1 summarizes some QoS requirements considered in the related works. For example, data rate, latency, and security have been investigated in [Guo and Yang, 2023; Guo *et al.*, 2019; Liu *et al.*, 2019a; Tang and Zhang, 2007; Wu *et al.*, 2003; Yang *et al.*, 2021; Yu *et al.*, 2016]. These papers mainly focus on scenarios with long channel coding blocklengths, where the achievable rate of a wireless link can be approximated by the Shannon capacity. In 5G, the coding blocklength can be short, and Shannon capacity is not

Table 5.1: Considered QoS Requirements in Related Works

QoS / Refs	Data rate		Latency		Security	
	Long	Short	Long	Short	Long	Short
[Guo and Yang, 2023; Guo *et al.*, 2019]	✓					
[Tang and Zhang, 2007; Wu *et al.*, 2003]			✓			
[Liu *et al.*, 2019a; Yang *et al.*, 2021; Yu *et al.*, 2016]					✓	
[Wang *et al.*, 2020]						✓
[Li *et al.*, 2022]				✓		✓
[Alsenwi *et al.*, 2021]	✓	✓				
[Dong *et al.*, 2021]	✓	✓	✓			

applicable. As such, the authors of [Li *et al.*, 2022; Wang *et al.*, 2020] established how to optimize wireless communication systems using the achievable rate in the short blocklength regime [Polyanskiy *et al.*, 2010]. Meanwhile, different services may co-exist in one network, and the authors of [Alsenwi *et al.*, 2021; Dong *et al.*, 2021] considered different QoS requirements in both long and short blocklength regimes. To support diverse QoS requirements in network slicing, the authors of [Sun *et al.*, 2019d] proposed to reserve bandwidth for different slices based on the number of users and the required QoS.

Further considering that the number of requests, the reserved resources, and the wireless channels are dynamic, improving the generalization ability of deep learning policies has attracted significant research interests in recent years. One approach to address this challenge is to carefully initialize the neural network and fine-tune it online. The authors of [Dong *et al.*, 2021] applied transfer learning to fine-tune the parameters of deep neural networks that are trained offline in dynamic wireless networks. To further improve the sample efficiency in an unseen communication scenario, meta-learning has been adopted in [Huang *et al.*, 2021a; Wang *et al.*, 2022b; Yuan *et al.*, 2021; Zhang *et al.*, 2022], where the hyper-parameters of a deep neural network, such as the initial parameters, are updated according to a set of communication scenarios in meta-training. In [Huang *et al.*, 2021a], meta-learning was applied to optimize computing resource allocation policies in mobile edge computing networks to fit both time-varying wireless channels and different requests of computing tasks. In [Wang *et al.*, 2022b], meta-learning was applied in virtual reality to quickly adapt to the user movement patterns changing over time. To improve the training efficiency in non-stationary vehicle networks, the authors in [Yuan *et al.*, 2021] proposed optimizing the beamforming using meta-learning. In [Zhang *et al.*, 2022], the authors combined meta-learning and support vector regression to extract the features for beamforming optimization, further improving training efficiency over non-stationary channels.

5.3 System Model and Problem Formulation

We consider an uplink orthogonal-frequency-division-multiple-access communication system with network slicing where U users are requesting different types of services from one base station (BS). The BS first reserves bandwidth for each type of service according to the QoS requirement and the number of users. Then, it allocates bandwidth to different users within each slice. The resource reservation for different slices has been extensively studied in the existing literature, so we will focus on developing bandwidth allocation policies for individual slices with different numbers of users, non-stationary wireless channels, and dynamic available bandwidth.

5.3.1 Different QoS Requirements for Long and Short Packets

To investigate the generalization ability of our proposed bandwidth allocation policy, we consider both long and short blocklength regimes with three types of QoS requirements, i.e., data rate, queuing delay, and security. Thus, there are six scenarios in total. We denote the reward of the u-th user by

$$r_u^{\Phi,\xi}, \quad \Phi \in \{D, E, S\} \text{ and } \xi \in \{\mathcal{I}, \mathcal{F}\}, \tag{5.3.1}$$

where superscripts D, E, S represent data rate, effective capacity with queuing delay constraint, and secrecy rate, respectively, whilst the superscripts $\mathcal{I}, \mathcal{F}$ represent the scenarios in the infinite long and finite short blocklength regimes, respectively.

Data Rate Requirement

When the blocklength is long, the data rate reward of the u-th user can be expressed as

$$r_u^{D,\mathcal{I}} = \frac{w_u}{\ln 2} \ln\left(1 + \frac{P_u h_u}{w_u N_0}\right), \tag{5.3.2}$$

where w_u is the bandwidth allocated to the u-th user, P_u is the transmit power of the u-th user, N_0 is the single-sided noise spectral density, and $h_u = \alpha_u g_u$ is the channel gain, where α_u and g_u represent the large-scale and small-scale channel gains between the u-th user and the BS, respectively.

When the blocklength is short, decoding errors cannot be neglected. As such, the data rate reward of the u-th user can be approximated by [Polyanskiy *et al.*, 2010]

$$r_u^{D,\mathcal{F}} \approx r_u^{D,\mathcal{I}} - \sqrt{\frac{V_u}{L_u}} \frac{f_Q^{-1}(\epsilon_u)}{\ln 2 / w_u} \tag{5.3.3}$$

where $V_u = 1 - \left(1 + \frac{P_u h_u}{w_u N_0}\right)^{-2}$ is the channel dispersion that measures the stochastic variability of the channel related to a deterministic channel with the same capacity,

$L_u = T_\mathrm{s} w_u$ is the blocklength, and T_s is the transmission duration of each coding block. The function $f_Q^{-1}(x)$ is the inverse of the Gaussian Q-function, and ϵ_u is the decoding error probability.

Latency Requirement

When considering latency constraints due to queueing delays, the effective capacity is applied to characterize the statistical QoS requirement in wireless communications, and is expressed as [Li *et al.*, 2022]

$$r_u^{E,\xi} = -\frac{1}{\vartheta_u T_\mathrm{c}} \ln \left(\mathbb{E}_{g_u} \left[\exp \left(-\vartheta_u T_\mathrm{c} r_u^{D,\xi} \right) \right] \right), \xi \in \{\mathcal{I}, \mathcal{F}\}, \tag{5.3.4}$$

where T_c is the channel coherence time, ϑ_u is the QoS exponent for queuing delay, $\mathbb{E}[\cdot]$ denotes the expectation, and $r_u^{D,\xi}$ is the data rate in (5.3.2) or (5.3.3). We note that $\vartheta_u = \frac{\ln(1/\varepsilon_u)}{a_u \tau_\mathrm{max}}$ is determined by the maximum tolerable delay bound violation probability, ε_u, the packet arrival rate, a_u, and the threshold of queuing delay, τ_max.

Security Requirement

To formulate the wireless security requirement, we consider that there is an eavesdropper that attempts to eavesdrop the information transmitted by each user.

In the long blocklength regime, the secrecy rate of the u-th user can be expressed as [Yu *et al.*, 2016]

$$r_u^{S,\mathcal{I}} = \left[r_u^{D,\mathcal{I}} - r_u^{e,\mathcal{I}} \right]^+, \tag{5.3.5}$$

where $[x]^+ = \max\{0, x\}$, and $r_u^{e,\mathcal{I}} = \frac{w_u}{\ln 2} \ln \left(1 + \frac{P_u h_u^e}{w_u N_0} \right)$ is the data rate of the eavesdropped channel from the u-th user to the eavesdropper. The channel gain of the eavesdropped channel is denoted by $h_u^e = \alpha_u^e g_u^e$, where α_u^e and g_u^e represent the large-scale and small-scale channel gains between the u-th user and the eavesdropper, respectively.

In the short blocklength regime, the achievable secrecy rate of the u-th user can be approximated as [Wang *et al.*, 2020],

$$r_u^{S,\mathcal{F}} = \begin{cases} r_u^{S,\mathcal{I}} - \sqrt{\frac{V_u}{L_u}} \frac{f_Q^{-1}(\epsilon_u)}{\ln 2/w_u} - \sqrt{\frac{V_u^e}{L_u}} \frac{f_Q^{-1}(\delta_u)}{\ln 2/w_u}, & h_u > h_u^e \\ 0, & h_u \leq h_u^e, \end{cases} \tag{5.3.6}$$

where $V_u^e = 1 - \left(1 + \frac{P_u h_u^e}{w_u N0}\right)^{-2}$, and δ_u represents the information leakage, which describes the statistical independence between the transmitted confidential message and the eavesdropper's observation, and is measured by the total variation distance [Wang *et al.*, 2020].

5.3.2 Bandwidth Reservation for Different Slices

We assume that there can be multiple bandwidth reservation policies for different slices in network slicing. Given the total bandwidth of the BS, $W_{\max}$, the bandwidth reserved for the τ-th slice is given by

$$W_\tau^{\Phi,\xi} = f_\tau^{\mathrm{NS}} \left(U_\tau^{\Phi,\xi}, I_{u_\tau}^{\Phi,\xi}\right) \cdot W_{\max}, \tag{5.3.7}$$

where $U_\tau^{\Phi,\xi}$ is the number of users in the τ-th slice, $I_{u_\tau}^{\Phi,\xi}$ is the QoS class identifier (QCI) of the u_τ-th user in the τ-th slice, and $f_\tau^{\mathrm{NS}}(\cdot,\cdot)$ is the network function for bandwidth reservation in network slicing. Since the sum of the bandwidth reserved for all the slices equals the total bandwidth of the BS, thus

$$\sum_{\tau=1}^{T_{\mathrm{NS}}} f_\tau^{\mathrm{NS}} \left(U_\tau^{\Phi,\xi}, I_{u_\tau}^{\Phi,\xi}\right) = 1. \tag{5.3.8}$$

where $T_{\rm NS}$ is the number of slices. Inspired by [Sun *et al.*, 2019d], the bandwidth reserved for each slice depends on the number of users in this slice and the QCI of these users, e.g.,

$$f_\tau^{\rm NS}(\cdot) = \frac{\sum_{u\in\mathcal{U}_\tau^{\Phi,\xi}} I_{u_\tau}^{\Phi,\xi}}{\sum_{\tau=1}^{T_{\rm NS}}\sum_{u\in\mathcal{U}_\tau^{\Phi,\xi}} I_{u_\tau}^{\Phi,\xi}}. \tag{5.3.9}$$

5.3.3 Problem Formulation

To maximize the sum reward subject to the QoS requirements in each slice, we formulate the bandwidth allocation problem as follows,

$$\max_{\boldsymbol{w}_\tau^{\Phi,\xi}} \quad \sum_{u\in\mathcal{U}_\tau^{\Phi,\xi}} r_u^{\Phi,\xi}, \tag{5.3.10}$$

$$\text{s.t.} \quad \sum_{u\in\mathcal{U}_\tau^{\Phi,\xi}} w_u^{\Phi,\xi} \le W_\tau^{\Phi,\xi}, \tag{5.3.10a}$$

$$w_u^{\Phi,\xi} \ge 0, \tag{5.3.10b}$$

$$r_u^{\Phi,\xi} \ge r_\tau^{\Phi,\xi}, \tag{5.3.10c}$$

where $\boldsymbol{w}_\tau^{\Phi,\xi} = [w_1^{\Phi,\xi}, w_2^{\Phi,\xi}, \cdots, w_{U_\tau}^{\Phi,\xi}]^{\rm T}$ is the bandwidth allocated to the users, and $r_\tau^{\Phi,\xi}$ is the minimum threshold of the QoS required by the users. Thus, constraint (5.3.10c) guarantees the QoS of all the users.

Our bandwidth allocation model is designed to optimize resource utilization in various types of wireless networks, such as eMBB and URLLC. By specifically addressing the requirements and challenges associated with these network types, our model aims to enhance the efficiency and performance of wireless communication systems in real-world scenarios.

Algorithm 3: User Scheduling Algorithm

```
1  Initialize the set of scheduled users as all the users: $\mathcal{K}_\tau^{\Phi,\xi} = \mathcal{U}_\tau^{\Phi,\xi}$.
2  *Drop u-th user if the target in* (5.3.10c) *cannot be satisfied with* $W_\tau^{\Phi,\xi}$:
3  for $u \in \mathcal{U}_\tau^{\Phi,\xi}$ do
4      Calculate the reward of each user by taking $w_u = W_\tau^{\Phi,\xi}$ into the target
         achievable rate shown in eqs. (5.3.2)-(5.3.6) as: $r_{u,\max}^{\phi,\xi} = r_u^{\phi,\xi}(W_\tau^{\Phi,\xi})$.
5      if $r_{u,\max}^{\phi,\xi} < r_\tau^{\Phi,\xi}$ then
6          Drop the $u$-th user: $\mathcal{K}_\tau^{\Phi,\xi} = \mathcal{K}_\tau^{\Phi,\xi} - \{u\}$, $K = K - 1$, and $w_u = 0$.
7      else
8          Get $w_{k,\min}$ using binary search function: $w_{k,\min} = f_{\mathrm{BiS}}(h_k, r_\tau^{\Phi,\xi})$.
9      end
10 end
11 *Drop k-user if* (5.3.10a) *cannot be satisfied with* $w_{k,\min}$:
12 while $\sum_{k=1}^{K} w_{k,\min} > W_\tau^{\Phi,\xi}$ do
13     Identify user holding highest $w_{k,\min}$ in $\mathcal{K}_\tau^{\Phi,\xi}$: $k_{\mathrm{drop}} = \arg\max_k w_{k,\min}$.
14     Drop the $k_{\mathrm{drop}}$-th user: $\mathcal{K}_\tau^{\Phi,\xi} = \mathcal{K}_\tau^{\Phi,\xi} - \{k_{\mathrm{drop}}\}$, $K = K - 1$, and $w_{k_{\mathrm{drop}}} = 0$.
15 end
```

5.3.4 Analysis of Problem Feasibility

Given the available bandwidth constraint in (5.3.10a) and the QoS constraint in (5.3.10c), problem (5.3.10) will be infeasible when some of the users in this slice have weak channels. To solve this issue, we propose to schedule the users to satisfy constraints (5.3.10a) and (5.3.10a) by Algorithm 3.

We denote the minimum bandwidth required to meet constraint (5.3.10c) by $\boldsymbol{w}_{\min}^{\Phi,\xi} = \left[w_{1,\min}^{\Phi,\xi}, \cdots, w_{U_\tau,\min}^{\Phi,\xi}\right]^{\mathrm{T}}$. If some users experience deep fading, leading to $\sum_{u \in \mathcal{U}_\tau^{\Phi,\xi}} w_{u,\min}^{\Phi,\xi} > W_\tau^{\Phi,\xi}$, then problem (5.3.10) is infeasible. In this case, the BS will only schedule the users with sufficiently strong channels. Alternatively, to maximize the number of scheduled users in problem (5.3.10), we consider that the BS schedules the K users with the smallest bandwidth requirement. Denote the set of scheduled users by $\mathcal{K}_\tau^{\Phi,\xi}$. Then, for any $k \in \mathcal{K}_\tau^{\Phi,\xi}$ and $u \notin \mathcal{K}_\tau^{\Phi,\xi}$, we have $w_{k,\min}^{\Phi,\xi} \leq w_{u,\min}^{\Phi,\xi}$. After user scheduling, problem (5.3.10) can be reformulated as follows,

$$\max_{\boldsymbol{w}_\tau^{\Phi,\xi}} \quad \sum_{k\in\mathcal{K}_\tau^{\Phi,\xi}} r_k^{\Phi,\xi}, \tag{5.3.11}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{K}_\tau^{\Phi,\xi}} w_k^{\Phi,\xi} \le W_\tau^{\Phi,\xi}, \tag{5.3.11a}$$

$$w_k^{\Phi,\xi} \ge w_{k,\min}^{\Phi,\xi}. \tag{5.3.11b}$$

In the following, we investigate how to find the optimal solution to problem (5.3.11).

5.4 HML for GNN-based Scalable Bandwidth Allocation

In this section, we first illustrate how to obtain the optimal bandwidth allocation by using an iterative optimization algorithm. Next, we utilize feature engineering techniques to reformulate the problem, and represent the bandwidth allocation policy by a GNN. To generalize the GNN, the feature of required minimum bandwidth that can be used to represent different QoS requirements is used as the GNN's input. Then, we develop a meta-learning approach to train the GNN. The goal is to obtain a policy that is scalable to the number of users and can generalize well in diverse communication scenarios with different channel distributions, QoS requirements, and available bandwidth.

5.4.1 Optimal Policy by Iterative Algorithm

Inspired by the optimization algorithm for resource allocation in [Dong *et al.*, 2021], we propose an iterative optimization algorithm for solving our problems. We denote the bandwidth of each resource block by Δw. At the beginning of the iteration, the bandwidth allocated to each user is $w_{k,\min}^{\Phi,\xi}$. In each iteration, we calculate the

Algorithm 4: Iterative Bandwidth Allocation Algorithm

```
1  Initialize: Bandwidth of a resource block: $\Delta w$.
2  Use user scheduling algorithm to get the minimum required bandwidth for
     each scheduled user: $w_k^{\Phi,\xi} = w_{k,\min}^{\Phi,\xi}, \forall k \in \mathcal{K}_\tau^{\Phi,\xi}$.
3  while $W_\tau^{\Phi,\xi} - \sum_{k\in\mathcal{K}_\tau^{\Phi,\xi}} w_k^{\Phi,\xi} \geq \Delta w$ do
4      for $k \in \mathcal{K}_\tau^{\Phi,\xi}$ do
5          $\Delta r_k^{\Phi,\xi}(w_k^{\Phi,\xi}) = r_k^{\Phi,\xi}(w_k^{\Phi,\xi} + \Delta w) - r_k^{\Phi,\xi}(w_k^{\Phi,\xi})$.
6      end
7      Identify user has highest $\Delta r_k^{\Phi,\xi}(w_k^{\Phi,\xi})$ in $\mathcal{K}_\tau^{\Phi,\xi}$: $k_{\text{allo}} = \arg\max_k \Delta r_k^{\Phi,\xi}(w_k^{\Phi,\xi})$.
8      Allocate extra $\Delta w$ bandwidth to the $k_{\text{allo}}$-th user: $w_{k_{\text{allo}}}^{\Phi,\xi} = w_{k_{\text{allo}}}^{\Phi,\xi} + \Delta w$.
9  end
10 Output: Optimal bandwidth allocation policy: $\boldsymbol{w}^{\Phi,\xi,\text{opt}} = \boldsymbol{w}^{\Phi,\xi}$.
```

incremental reward of each user when an extra resource block is allocated to it, denoted by $\Delta r_k^{\Phi,\xi}(w_k) = r_k^{\Phi,\xi}(w_k^{\Phi,\xi} + \Delta w) - r_k^{\Phi,\xi}(w_k^{\Phi,\xi})$. Finally, the resource block is allocated to the user with the highest $\Delta r_k^{\Phi,\xi}$. The details of the algorithm can be found in Algorithm 4. The optimality of the algorithm depends on the properties of the problems. For problem (5.3.11), if it is a convex problem, then Algorithm 4 can obtain the optimal solution [Dong *et al.*, 2021]. To validate whether problem (5.3.11) is convex or not, we only need to validate whether $r_u^{\Phi,\xi}$ is concave or not. In the long blocklength regime, we can prove that the secrecy rate is concave in bandwidth. See proof in Appendix B.1. Since Shannon's capacity is a special case of the secrecy rate when the eavesdropped channel is in deep fading, thus Shannon's capacity is also concave in bandwidth. In addition, the authors of [Xiong *et al.*, 2013] proved that the effective capacity is concave in bandwidth. Therefore, Algorithm 4 can obtain the optimal solution in the long blocklength regime. In the short-blocklength regime, $r_u^{\Phi,\xi}$ is not concave when $w_k^{\Phi,\xi} \in (0, \infty)$. Nevertheless, based on the results in [Sun *et al.*, 2019c], the optimal bandwidth can be obtained in a region $[0, w_{\text{th}}] \subset (0, \infty)$, where

$r_u^{\Phi,\xi}$ is concave in bandwidth. By searching for the optimal bandwidth in $[0, w_{\text{th}}]$, Algorithm 4 can obtain the optimal solution in the short blocklength regime.

5.4.2 Feature Engineering and Problem Reformulation

To obtain a policy that can generalize well in different scenarios, we propose to use feature engineering technology to represent the channels and QoS requirements with more general features. Specifically, we first normalize the bandwidth allocation policy by the bandwidth reserved for this slice. The normalized bandwidth allocated to the k-th user, $k \in \mathcal{K}_\tau^{\Phi,\xi}$, is given by $\tilde{w}_k^{\Phi,\xi} \triangleq w_k^{\Phi,\xi}/W_\tau^{\Phi,\xi}$. Then, the normalized minimum bandwidth required by the scheduled users is denoted by $\tilde{\boldsymbol{w}}_{\tau,\min}^{\Phi,\xi} = [\tilde{w}_{1,\min}^{\Phi,\xi}, \tilde{w}_{2,\min}^{\Phi,\xi}, ..., \tilde{w}_{K_\tau,\min}^{\Phi,\xi}]^{\text{T}}$. We define the surplus bandwidth as $w_{\text{S}}^{\Phi,\xi} = W_\tau^{\Phi,\xi} - \sum_{k\in\mathcal{K}_\tau^{\Phi,\xi}} w_k$, and further denote the normalized surplus bandwidth by $\tilde{w_{\text{S}}}^{\Phi,\xi} \triangleq w_{\text{S}}^{\Phi,\xi}/W_\tau^{\Phi,\xi}$.

We note that bandwidth allocation policy maps channels and constraints to the bandwidth allocated to each user. After scheduling and normalization, the features of the channel state information and constraints (5.3.11a) and (5.3.11b) can be represented by $\tilde{\boldsymbol{w}}_{\tau,\min}^{\Phi,\xi}$. Therefore, the bandwidth allocation policy can be reformulated as the mapping from $\tilde{\boldsymbol{w}}_{\tau,\min}^{\Phi,\xi}$ and $\tilde{w}_{\text{S}^{\Phi,\xi}}$ to $\tilde{\boldsymbol{w}}^{\Phi,\xi}$. We denote this function by

$$\tilde{\boldsymbol{w}}_\tau^{\Phi,\xi} = \boldsymbol{f}^{\text{W}}\left(\tilde{\boldsymbol{w}}_{\tau,\min}^{\Phi,\xi}, \tilde{w}_{\text{S}}^{\Phi,\xi}\right) \tag{5.4.1}$$

where $\boldsymbol{f}^{\text{W}}(\tilde{\boldsymbol{w}}_{\tau,\min}^{\Phi,\xi}, \tilde{w}_{\text{S}}^{\Phi,\xi}) = \left[f_1^{\text{W}}(\tilde{\boldsymbol{w}}_{\tau,\min}^{\Phi,\xi}, \tilde{w}_{\text{S}}^{\Phi,\xi}), f_2^{\text{W}}(\tilde{\boldsymbol{w}}_{\tau,\min}^{\Phi,\xi}, \tilde{w}_{\text{S}}^{\Phi,\xi}), \cdots, f_K^{\text{W}}(\tilde{\boldsymbol{w}}_{\tau,\min}^{\Phi,\xi}, \tilde{w}_{\text{S}}^{\Phi,\xi})\right]^{\text{T}}$ and $\tilde{w}_k^{\Phi,\xi} = f_k^{\text{W}}(\tilde{\boldsymbol{w}}_{\tau,\min}^{\Phi,\xi}, \tilde{w}_{\text{S}}^{\Phi,\xi})$. Given the bandwidth reserved for this slice, the achievable rates of the scheduled users can be expressed as

$$\begin{aligned} \boldsymbol{r}_\tau^{\Phi,\xi} &= \boldsymbol{f}^{\Phi,\xi}\left(\tilde{\boldsymbol{w}}^{\Phi,\xi} \cdot W_\tau^{\Phi,\xi}\right) \\ &= \boldsymbol{f}^{\Phi,\xi}\left(\boldsymbol{f}^{\text{W}}\left(\tilde{\boldsymbol{w}}_{\tau,\min}^{\Phi,\xi}, \tilde{w}_{\text{S}}^{\Phi,\xi}\right) \cdot W_\tau^{\Phi,\xi}\right) \end{aligned} \tag{5.4.2}$$

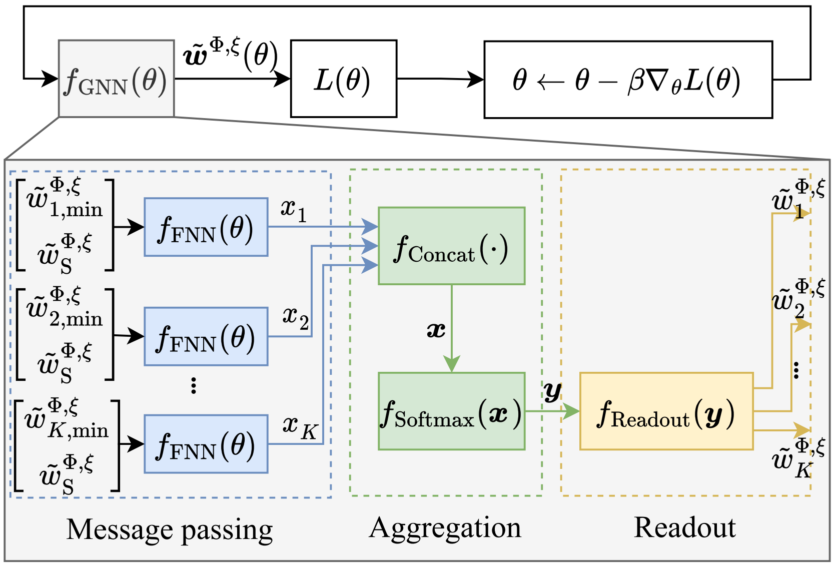

$f_{\mathrm{GNN}}(\theta)$
$\tilde{\boldsymbol{w}}^{\Phi,\xi}(\theta)$
$L(\theta)$
$\theta \leftarrow \theta - \beta\nabla_\theta L(\theta)$
$\tilde{w}^{\Phi,\xi}_{1,\min}$, $\tilde{w}^{\Phi,\xi}_{\mathrm{S}}$
$\tilde{w}^{\Phi,\xi}_{2,\min}$, $\tilde{w}^{\Phi,\xi}_{\mathrm{S}}$
$\tilde{w}^{\Phi,\xi}_{K,\min}$, $\tilde{w}^{\Phi,\xi}_{\mathrm{S}}$
$f_{\mathrm{FNN}}(\theta)$
x_1
x_2
x_K
$f_{\mathrm{Concat}}(\cdot)$
$\boldsymbol{x}$
$f_{\mathrm{Softmax}}(\boldsymbol{x})$
$\boldsymbol{y}$
$f_{\mathrm{Readout}}(\boldsymbol{y})$
$\tilde{w}^{\Phi,\xi}_1$
$\tilde{w}^{\Phi,\xi}_2$
$\tilde{w}^{\Phi,\xi}_K$
Message passing
Aggregation
Readout

Figure 5.1: GNN-based scalable bandwidth allocation.

where the function $\boldsymbol{f}^{\Phi,\xi}(\tilde{\boldsymbol{w}}^{\Phi,\xi}_\tau \cdot W^{\Phi,\xi}_\tau) = \left[f^{\Phi,\xi}_1(\tilde{w}^{\Phi,\xi}_1 \cdot W^{\Phi,\xi}_\tau), \cdots, f^{\Phi,\xi}_K(\tilde{w}^{\Phi,\xi}_{K_\tau} \cdot W^{\Phi,\xi}_\tau)\right]^{\mathrm{T}}$, $\boldsymbol{r}^{\Phi,\xi}_\tau = \left[r^{\Phi,\xi}_1, \cdots, r^{\Phi,\xi}_{K_\tau}\right]^{\mathrm{T}}$, and $r^{\Phi,\xi}_k = f^{\Phi,\xi}_k\left(f^{\mathrm{W}}_k(\tilde{\boldsymbol{w}}^{\Phi,\xi}_{\tau,\min}, \tilde{w}_{\mathrm{S}}) \cdot W^{\Phi,\xi}_\tau\right)$. Then, we can reformulate problem (5.3.11) as a functional optimization problem,

$$\max_{\boldsymbol{f}^{\mathrm{W}}(\cdot)} \sum_{k\in\mathcal{K}^{\Phi,\xi}_\tau} f^{\Phi,\xi}_k\left(f^{\mathrm{W}}_k\left(\tilde{\boldsymbol{w}}^{\Phi,\xi}_{\tau,\min}, \tilde{w}^{\Phi,\xi}_{\mathrm{S}}\right) \cdot W^{\Phi,\xi}_\tau\right), \tag{5.4.3}$$

$$\text{s.t.} \quad 1 - \sum_{k\in\mathcal{K}^{\Phi,\xi}_\tau} f^{\mathrm{W}}_k\left(\tilde{\boldsymbol{w}}^{\Phi,\xi}_{\tau,\min}, \tilde{w}^{\Phi,\xi}_{\mathrm{S}}\right) \geq 0, \tag{5.4.3a}$$

$$f^{\mathrm{W}}_k\left(\tilde{\boldsymbol{w}}^{\Phi,\xi}_{\tau,\min}, \tilde{w}^{\Phi,\xi}_{\mathrm{S}}\right) \geq \tilde{w}^{\Phi,\xi}_{k,\min}. \tag{5.4.3b}$$

In the rest part of this section, we will find the optimal solution to problem (5.4.3).

5.4.3 Proposed GNN

In this subsection, we propose a GNN-based unsupervised learning algorithm to obtain a scalable bandwidth allocation policy.

Algorithm 5: GNN for Scalable Bandwidth Allocation.

```
1  Initialize batch size, $J$, number of training epochs, $N$, and learning rate $\beta_\theta$.
2  Randomly initialize $\theta^0$.
3  for $n = 0, 1, \cdots, N-1$ do
4      Message passing: $x_k^n = f_{\text{FNN}}\left(\tilde{w}_{k,\min}^{\Phi,\xi}, \tilde{w}_{\text{S}}^{\Phi,\xi} \middle| \theta^n\right), \forall k \in \mathcal{K}_{\tau}^{\Phi,\xi}$.
5      Aggregation: $\boldsymbol{x}^n = f_{\texttt{Concat}}\left(x_1^n, \cdots, x_K^n\right) = [x_1^n, \cdots, x_K^n]^{\text{T}}$ and $\boldsymbol{y}^n = f_{\texttt{Softmax}}(\boldsymbol{x}^n)$.
6      Readout: $\tilde{\boldsymbol{w}}^n = f_{\texttt{Readout}}(\boldsymbol{y}^n) = \boldsymbol{y}^n \cdot \tilde{w}_{\text{S}}^{\Phi,\xi} + \tilde{\boldsymbol{w}}_{\min}^{\Phi,\xi}$.
7      Update the loss function by eq. (5.4.4), denoted by $f^{\text{L}}(\theta^n)$.
8      Update parameters of the GNN by SGA: $\theta^{n+1} = \theta^n + \beta_\theta \nabla_\theta f^{\text{L}}(\theta^n)$.
9  end
10 Return the parameters of the GNN as: $\theta^{\text{opt}}$.
```

Structure of GNN

As shown in Fig. 5.1, the proposed GNN-based bandwidth allocation algorithm comprises three key steps: message passing, aggregation, and readout.

Message passing Each scheduled user is a vertex in the GNN. We use a fully connected neural network (FNN) to obtain the embedding of each vertex, denoted by $x_k, \forall k \in \mathcal{K}_{\tau}^{\Phi,\xi}$. The inputs of each FNN include $\tilde{w}_{k,\min}^{\Phi,\xi}$ and $\tilde{w}_{\text{S}}^{\Phi,\xi} = 1 - \sum_{k \in \mathcal{K}_{\tau}^{\Phi,\xi}} \tilde{w}_{k,\min}^{\Phi,\xi}$. We use θ to denote the training parameters of the FNN. In the n-th epoch, the message passing function is given by $x_k^n = f_{\text{FNN}}\left(\tilde{w}_{k,\min}^{\Phi,\xi}, \tilde{w}_{\text{S}}^{\Phi,\xi} \middle| \theta^n\right)$. Since the vertices are homogeneous, the training parameters of all the FNNs are the same.

Aggregation In the aggregation step, we first aggregate the embeddings of all the scheduled users by using a concatenation function, $f_{\texttt{Concat}}(\cdot)$, followed by a `Readout` function, $f_{\texttt{Readout}}(\cdot)$, which serves as the activation function in the aggregation. The output after aggregation is denoted by $\boldsymbol{y}$.

Readout The GNN's output of each vertex is updated by a readout function given by, $f_{\texttt{Readout}}(\boldsymbol{y}) = \boldsymbol{y} \cdot \tilde{w}_{\mathrm{S}}^{\Phi,\xi} + \tilde{\boldsymbol{w}}_{\tau,\min}^{\Phi,\xi}$. Since $\boldsymbol{y}$ is obtained from the `Readout` function, the summation of its elements is one. From the readout function, all the surplus bandwidth is allocated to the users, and constraints (5.4.3a) and (5.4.3b) can be satisfied.

Unsupervised Learning

The learning algorithm is detailed in Algorithm 5. Specifically, in the n-th epoch, we use our GNN to obtain the bandwidth allocation and estimate the expectation of the objective function by using the batch samples according to

$$f^{\mathrm{L}}(\theta) = \frac{1}{J} \sum_{j=1}^{J} \sum_{k \in \mathcal{K}_{\tau}^{\Phi,\xi}} f_k^{\Phi,\xi} \left(\tilde{w}_{j,k}^{\Phi,\xi} \cdot W_{j,\tau}^{\mathrm{NS}} \right), \tag{5.4.4}$$

where J is the batch size. Then, we use stochastic gradient descent (SGA) to maximize the estimated expectation of the objective function in (5.4.3). As shown in [Sun *et al.*, 2023], maximizing the expectation of the objective function, where the expectation is taken over channels, is equivalent to maximizing the objective function with given channels. Thus, from Algorithm 5, we can find the bandwidth allocation policy that maximizes the objective function in (5.4.3).

Computational Complexity

We compare the computational complexity of our GNN with the iterative algorithm introduced in Section 5.4.1. In cellular systems, both algorithms will be implemented in each transmission time interval with a duration of less than 1 ms. Thus, we are interested in the inference complexity of our GNN, i.e., the number of operations to be executed to obtain the bandwidth allocation in each transmission time interval.

Inference complexity of our GNN To compute the embedding of each vertex, we need to compute the output of the FNN in Fig. 5.1. We denote the number of layers of the FNN by L_{FNN} and the number of neurons in the ℓ-th layer by m^{ℓ}_{FNN}. Then, the number of multiplications required to compute the output of the ℓ-th layer is $m^{\ell}_{\text{FNN}} \cdot m^{\ell+1}_{\text{FNN}}$ and the total number of multiplications for computing the embedding is $M_{\text{FNN}} = \sum_{\ell=1}^{L_{\text{FNN}}} m^{\ell}_{\text{FNN}} \cdot m^{\ell+1}_{\text{FNN}}$ [Dong *et al.*, 2021]. After obtaining the embeddings of K users, the number of multiplications required by $f_{\texttt{Readout}}(\boldsymbol{x})$ and $f_{\texttt{Readout}}(\boldsymbol{y})$ is $2K$. Therefore, the inference complexity of the GNN-based bandwidth allocation policy is

$$O_{\text{GNN}} = O(K \cdot (M_{\text{FNN}} + 2)). \tag{5.4.5}$$

Complexity of the iterative algorithm In each iteration of the optimization algorithm, we assign a small portion of the normalized surplus bandwidth, denoted by $\Delta\tilde{w}$, to a user that can maximize the objective function. The algorithm needs to compute the objective function K times and find the best user. We denote the complexity for computing the objective function by Ω, then the complexity of the iterative algorithm is given by

$$O_{\text{iter}} = O\left(K \cdot \frac{w_{\text{S}}}{\Delta w} \cdot \Omega\right), \tag{5.4.6}$$

where $w_{\text{S}}/\Delta w$ represents the number of iterations used in the iterative algorithm.

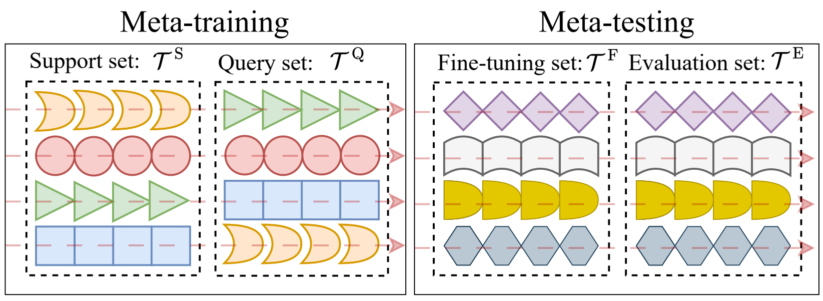
Complexity comparison To obtain bandwidth allocation in each transmission time interval, the transmitter either uses the forward propagation algorithm to compute the outcome of the GNN or executes the iterative algorithm. From eqs. (5.4.5) and (5.4.6), we can see that the computational complexity of our GNN and the iterative algorithm increase linearly with the number of users. Recall that M_{FNN} in eq. (5.4.5) is quite limited. In contrast, the complexity of the iterative algorithm also increases with the amount of surplus bandwidth and the resource block and thus depends on the channels of the users. In addition, the computing complexity for evaluating the objective function, denoted by Ω in eq. (5.4.6), in each iteration of the optimization algorithm could also be extremely high. Thus, the inference complexity of the GNN is much lower than the complexity of the iterative optimization algorithm.

5.4.4 Proposed HML Algorithm

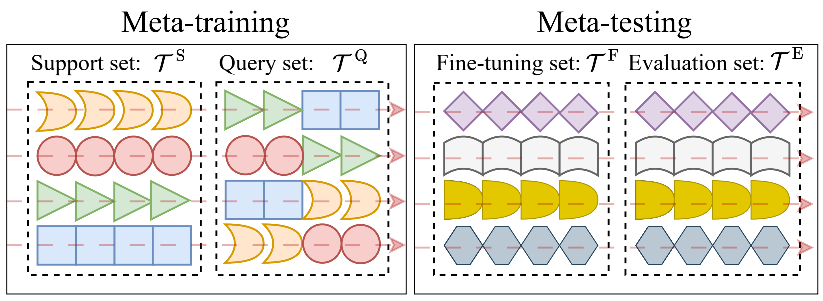
To obtain a GNN with strong generalization ability, we propose an HML algorithm that combines multi-task learning and meta-learning.

Task, Sample, and Taskset

To apply the meta-learning framework, we first define tasks, samples, and tasksets in the context of bandwidth allocation problems. A task is a specific bandwidth allocation problem with a unique combination of system parameters, including the number of users, U, the channel model (i.e., path loss model, shadowing, and small-scale channel fading), the QoS requirement, $r_{\tau}^{\Phi,\xi}$, and the reserved bandwidth, $W_{\tau}^{\Phi,\xi}$. If any of the above system parameters change, it would result in a different task. For each task, the samples correspond to the wireless channels that have been transformed into the minimum bandwidth requirement by feature engineering, as specified in constraint (5.4.3b). There are four tasksets in meta-learning, and a taskset consists of multiple tasks. We will provide their definitions in the sequel.


Meta-training
Meta-testing
Support set: $\mathcal{T}^{\mathrm{S}}$
Query set: $\mathcal{T}^{\mathrm{Q}}$
Fine-tuning set: $\mathcal{T}^{\mathrm{F}}$
Evaluation set: $\mathcal{T}^{\mathrm{E}}$

(a) Model-agnostic meta-learning (MAML).


Meta-training
Meta-testing
Support set: $\mathcal{T}^{\mathrm{S}}$
Query set: $\mathcal{T}^{\mathrm{Q}}$
Fine-tuning set: $\mathcal{T}^{\mathrm{F}}$
Evaluation set: $\mathcal{T}^{\mathrm{E}}$

(b) Hybrid-task meta-learning (HML).

Figure 5.2: Tasksets of meta-learning algorithms, where different shapes represent different tasks.

Support Set and Query Set in Meta-Training

As shown in Fig. 5.2(a), most meta-learning learning algorithms, such as MAML, consist of a meta-training stage and a meta-testing stage. In meta-training, there are two tasksets, support set $\mathcal{T}^{\mathrm{S}}$ and query set $\mathcal{T}^{\mathrm{Q}}$. The tasks in the two tasksets are the same, but the samples of each task in the two tasksets are different. Specifically, we first set the initialize parameters of the GNN to ϕ, which is randomly initialized at the beginning of meta-training, and updated in every iteration of the meta-training. Then, we train the parameters of the GNN by using the tasks and the corresponding samples in the support set, where θ is initialized with parameters ϕ. Then, we update the initial parameters ϕ by using the tasks and the corresponding samples in the query set. We denote the initial parameters trained in meta-training of MAML by ϕ^*. The details of the MAML algorithm can be found in [Finn *et al.*, 2017].

Fine-Tuning Set and Evaluation Set in Meta-Testing

To evaluate the generalization ability of the GNN, a different set of tasks that are unseen in the meta-training stage are used in meta-testing. As shown in Fig. 5.2(a), the tasks in meta-testing are divided into a fine-tuning set and an evaluation set, denoted by $\mathcal{T}^{\mathrm{F}}$ and $\mathcal{T}^{\mathrm{E}}$, respectively. The tasks in $\mathcal{T}^{\mathrm{F}}$ and $\mathcal{T}^{\mathrm{E}}$ are the same, but the samples of each task in these two tasksets are different. For each new task in meta-testing, the samples from the fine-tuning set are used to fine-tune θ, which is initialized by ϕ^* obtained in meta-training. After fine-tuning, the updated GNN is tested with the samples from the evaluation set. If no sample is used to fine-tune the GNN in the meta-testing stage, we refer to this approach as zero-shot meta-learning. Otherwise, it is known as few-shot meta-learning. The meta-testing algorithm is detailed in Algorithm 6.

Meta-Training of Proposed HML Algorithm

Fig. 5.2(b) illustrates the tasks and tasksets used in the meta-training and meta-testing of the proposed HML algorithm. The difference between MAML and HML lies in the selection of tasks from the query set. In MAML, the tasks selected from the query set are identical to those selected from the support set in each meta-training epoch. To improve the generalization ability, in HML, we select different tasks from the query set to train the initial parameters of the GNN. Specifically, I' tasks are randomly selected from the query set to estimate the average loss of the GNN parameterized by ϕ^m in the m-th epoch of meta-training. The step-by-step algorithm

Algorithm 6: Meta-Testing

```
1  Initialize the number of training epochs, $N$, and the learning rate of the
     target testing task, $\beta_\theta$.
2  Select the $i$-th task from the fine-tuning set and the evaluation set: $\mathcal{T}_i^{\mathrm{F}} \in \mathcal{T}^{\mathrm{F}}$
     and $\mathcal{T}_i^{\mathrm{E}} \in \mathcal{T}^{\mathrm{E}}$.
3  Set the initialization parameters of the GNN as: $\theta^0 = \phi^*$.
4  for $n = 0, 1, \cdots, N-1$ do
5      Randomly select $J$ samples from task $\mathcal{T}_i^{\mathrm{F}}$.
6      Calculate loss of in the fine-tuning set according to (5.4.4), denoted by
         $f^{\mathrm{L,F}}(\theta^n)$.
7      Update the parameters of GNN by: $\theta^{n+1} = \theta^n + \beta_\theta \nabla_\theta f^{\mathrm{L,F}}(\theta^n)$.
8  end
9  Randomly select $J'$ samples from task $\mathcal{T}_i^{\mathrm{E}}$.
10 Evaluate the fine-tuned policies of the $i$-th task using $\theta^N$:
     $\boldsymbol{w}_i^{\Phi,\xi} = f_{\mathrm{GNN}}\left(\tilde{\boldsymbol{w}}_{\min,i,j'}^{\Phi,\xi}, \tilde{w}_{\mathrm{S},i,j'}^{\Phi,\xi} \middle| \theta^N\right)$.
```

for meta-training of the proposed HML algorithm is described in Algorithm 7, and the meta-testing algorithm of HML is the same as that of MAML in Algorithm 6.

5.5 Performance Evaluation

In this section, we evaluate the performance of our GNN-based HML algorithm. The GNN is first initialized by the parameters obtained from meta-training, where all the tasks aim to maximize the sum of the secrecy rate with different numbers of users and channel models. Then, we evaluate the performance of our GNN in unseen tasks with different numbers of users, channel models, objective functions, QoS constraints, and reserved bandwidth.

5.5.1 Simulation Setup

We consider a BS, located at $(0, 0)$ m, serving multiple users randomly distributed in a rectangular area, where the coordinates of the users are denoted by $(c_{x,u}, c_{y,u})$,

Algorithm 7: Meta-Training of Hybrid-Task Meta-Learning

```
1  Randomly initialize the training parameters for all the tasks, $\phi$, the number
     of meta-training epochs, $M$, the learning rate of meta-training, $\beta_\phi$, and the
     learning rate of each task, $\beta_\theta$.
2  for $m = 0, 1, \cdots, M-1$ do
3      Select a batch of $I$ tasks from the support set:
        $\mathcal{T}_i^{\mathrm{S}} \in \mathcal{T}^{\mathrm{S}}, \quad i \in \{1, 2, \cdots, I\}$.
4      for $i = 1, 2, \cdots, I$ do
5          Set the initial parameters of the GNN to $\theta_i^m = \phi^m$.
6          for $n = 0, 1, \cdots, N-1$ do
7              Randomly select $J$ samples from task $\mathcal{T}_i^{\mathrm{S}}$.
8              Calculate the loss function in the support set according to (5.4.4),
                 denoted by $f^{\mathrm{L,S}}(\theta_i^{m,n})$.
9              Update the parameters by: $\theta_i^{m,n+1} = \theta_i^{m,n} + \beta_\theta \nabla_\theta f^{\mathrm{L,S}}(\theta_i^{m,n})$.
10         end
11         Select a batch of $I'$ tasks from the query set:
            $\mathcal{T}_{i'}^{\mathrm{Q}} \in \mathcal{T}^{\mathrm{Q}}, \quad i' \in \{1, 2, \cdots, I'\}$.
12         for $i' = 1, \cdots, I'$ do
13             Randomly select $J'$ samples from task $\mathcal{T}_{i'}^{\mathrm{Q}}, \quad j' \in \{1, 2, ..., J'\}$.
14             Calculate the loss function in the query task:
                $f_i^{\mathrm{L,Q}}\left(\theta_i^{m,N}\right) = \frac{1}{I'}\frac{1}{J'}\sum_{i'=1}^{I'}\sum_{j'=1}^{J'}\sum_{k\in\mathcal{K}_\tau^{\Phi,\xi}} f_k^{\Phi,\xi}\left(\tilde{w}_{i',j',k}^{\Phi,\xi,m} \cdot W_{\tau,j}^{\mathrm{NS}}\right)$,
15         end
16     end
17     Calculate the loss function in meta-training:
        $f^{\mathrm{L,Meta},m}\left(\phi^m\right) = \frac{1}{I}\sum_{i=1}^{I} f_i^{\mathrm{L,Q}}\left(\theta_i^{m,N}\right)$.
18     Update the initial parameters: $\phi^{m+1} = \phi^m + \beta_\phi \nabla_\phi f^{\mathrm{L,Meta},m}\left(\phi^m\right)$.
19 end
20 Return the optimal initial parameters of the GNN: $\phi^{\mathrm{opt}} = \phi^M$.
```

where $c_{x,u}$ and $c_{y,u} \in [-100, 100]$ m. These users are distributed randomly within these coordinates. When the QoS requirement is secrecy rate, an eavesdropper is also randomly located in the above rectangular area. The transmitted signal of each user is a complex Gaussian process with zero-mean and equal variance, $\sigma^2 = 1$. Channel

Table 5.2: Chapter 5 Key Simulation Parameters

Simulation parameters	**Values**
Transmit power of each user, P_u	23 dBm
Single-sided noise spectral density, N_0	-174 dBm/Hz
Channel coherence time, T_c	1ms [Li *et al.*, 2022]
Duration of one time slot, T_s	0.125ms
Decoding error probability, ϵ_u	10^{-5} [Li *et al.*, 2022]
Information leakage, δ_u	10^{-2} [Li *et al.*, 2022]
QoS exponent of queuing delay, ϑ_u	10^{-3} [Li *et al.*, 2022]
Minimum size of bandwidth resource block, Δw	10 kHz
Learning rates, β_θ/β_ϕ	10^{-4}
Batch sizes of GNN, J/J'	32
Batch sizes of meta optimizer, I, I'	4, 2

models include large-scale channels and small-scale channels. Specifically, the large-scale channels depend on path loss and shadowing fading, whilst small-scale channels follow Rice, Nakagami, and Rayleigh distributions with various parameters in Table 5.3, in this table

$$p_u^\mathrm{S}(\psi) = \frac{10/\ln 10}{\sqrt{2\pi}\sigma_{\psi_\mathrm{dB}}\psi} \exp\left(-\frac{(10\log_{10}\psi - \mu_{\psi_\mathrm{dB}})^2}{2\sigma_{\psi_\mathrm{dB}}^2}\right), \tag{5.5.1}$$

$$p_u^\mathrm{I}(z|s,\sigma) = \frac{z}{\sigma^2}\exp\left(-\frac{z^2+s^2}{2\sigma^2}\right)\cdot I_0\left(\frac{zs}{\sigma^2}\right), \tag{5.5.2}$$

$$p_u^\mathrm{N}(z|m,\sigma) = \frac{2m^m z^{2m-1}}{\Gamma(m)(2\sigma^2)^m}\exp\left(-\frac{mz^2}{2\sigma^2}\right), \tag{5.5.3}$$

$$p_u^\mathrm{R}(z|\sigma) = \frac{z}{\sigma^2}\exp\left(-\frac{z^2}{2\sigma^2}\right). \tag{5.5.4}$$

. The number of neurons in each layer of the GNN is 2/32/64/64/32/1. Unless otherwise mentioned, the simulation parameters are summarized in Table 5.2, and the parameters of tasksets are defined in Table 5.3.

Table 5.3: Chapter 5 System Parameters of Different Tasks

	Parameters	$\mathcal{T}^{\mathrm{S}}$ & $\mathcal{T}^{\mathrm{Q}}$	$\mathcal{T}^{\mathrm{F}}$ & $\mathcal{T}^{\mathrm{E}}$
Network scale	Number of users	$U_{\tau}^{\Phi,\xi} \in \{10, 11, \cdots, 30\}$	$U_{\tau}^{\Phi,\xi} = 50$
Channels	Path loss: $\alpha_u = (d_u)^{-\gamma_u}$	$\gamma_u \in \{2, 3\}$	$\gamma_u = 4$
	Shadowing: $p_u^{\mathrm{S}}(\psi)$	$\psi_{\mathrm{dB}} \in \{3, 4, 5\}$	$\psi_{\mathrm{dB}} = 8$
	Small-scale: $p_u^{\mathrm{I}}(z\|s, \sigma)$, $p_u^{\mathrm{N}}(z\|m, \sigma)$, $p_u^{\mathrm{R}}(z\|\sigma)$	$p_u^{\mathrm{I}}(z\|s, \sigma), s \in \{1 \cdots 5\}$, $p_u^{\mathrm{N}}(z\|m, \sigma), m \in \{2, \cdots, 6\}$	$p_u^{\mathrm{R}}(z\|\sigma)$
QoS	Rewards	$\max\limits_{\boldsymbol{w}} \sum\limits_{u \in \mathcal{U}_{\tau}^{S,\mathcal{I}}} r_u^{S,\mathcal{I}}$	$\max\limits_{\boldsymbol{w}} \sum\limits_{u \in \mathcal{U}_{\tau}^{\Phi,\xi}} r_u^{\Phi,\xi}$, $\Phi \in \{D, E, S\}$, $\xi \in \{\mathcal{I}, \mathcal{F}\}$
	Constraints (Mbps)	$r_{\tau}^{S,\mathcal{I}} \in \{1, \cdots, 10\}$	$r_{\tau}^{\Phi,\xi} = 10$
Reserved Bandwidth	Constraints (MHz)	$W_{\tau}^{S,\mathcal{I}} \in \{10, \cdots, 100\}$	$W_{\tau}^{\Phi,\xi} = 100$

5.5.2 Performance of GNN

Fig. 5.3 shows the training losses when the number of users increases from 10 to 50. The results show that the unsupervised learning algorithm can converge after a few hundred training epochs for different numbers of users, and the convergence time increases slightly with the number of users.

After the training stage of the unsupervised learning algorithm, we select 1000 samples from the evaluation set of the same task to evaluate the constraint and reward achieved by the GNN in Fig. 5.4. The results in Fig. 5.4(a) show that the secrecy rates of all the scheduled users are equal to or higher than the requirement, $r_{\tau}^{S,\mathcal{I}} = 10$ Mbps. The results in Fig. 5.4(b) show that the sum secrecy rate achieved by the GNN is close to that achieved by the iterative optimization algorithm in Section 5.4.1 (with legend "Optimal"). In other words, the unsupervised learning algorithm can obtain

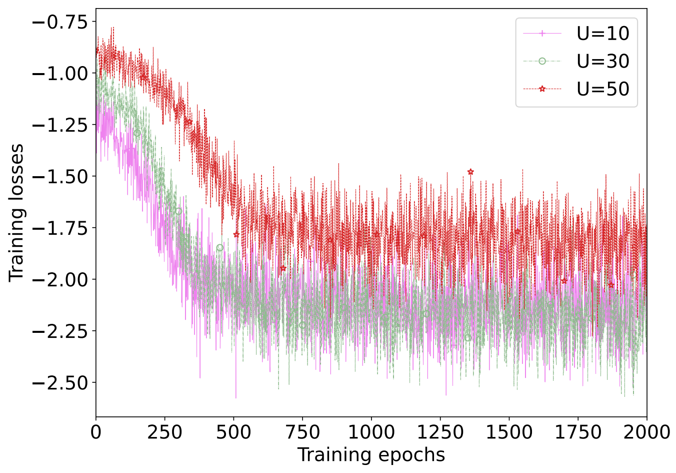


Figure 5.3: Training losses with different numbers of users, where the secrecy rate in the long blocklength regime is considered, $r_{\tau}^{S,\mathcal{I}} = 10$ Mbps, and $W_{\tau}^{S,\mathcal{I}} = 100$ MHz.

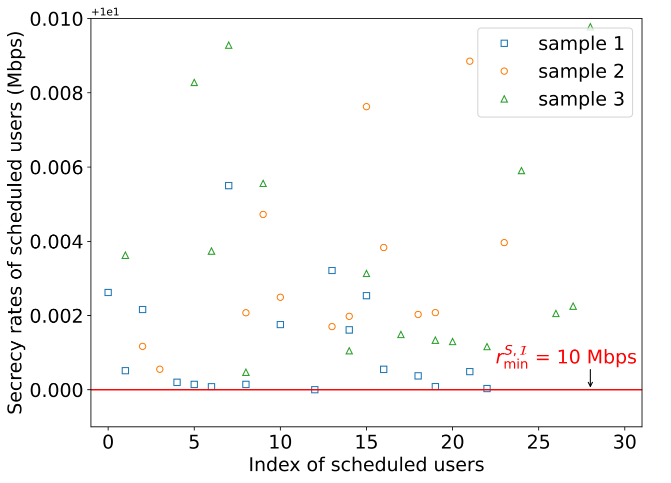
a near-optimal solution.

5.5.3 Meta-Testing Performance of HML

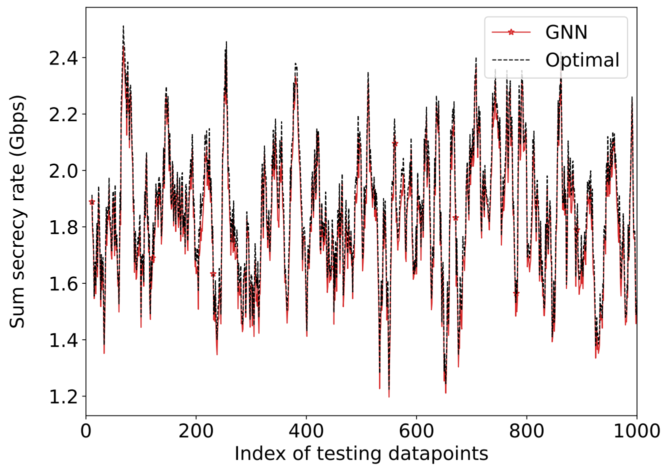
In this subsection, we evaluate the generalization ability of the proposed HML algorithm. The differences between tasks in meta-training and meta-testing are shown in Table. 5.3. In meta-testing, we first select an unseen task that is not included in meta-training. In each training epoch of the meta-testing, 32 samples are randomly selected from $\mathcal{T}^{\mathrm{F}}$ to fine-tune the GNN, whilst all the 1000 testing samples from the same task in $\mathcal{T}^{\mathrm{E}}$ are used to evaluate the performance.

Different Wireless Channels and QoS Requirements

In this part, we set $W_{\tau}^{S,\mathcal{I}} = 100$ MHz and $r_{\tau}^{S,\mathcal{I}} = 10$ Mbps for all types of services. The other parameters follow the rules in $\mathcal{T}^{\mathrm{S}}$ and $\mathcal{T}^{\mathrm{Q}}$ as shown in Table. 5.3. We compare the initial performance and sample efficiency of HML with four benchmarks: 1) Optimal, 2) Model-agnostic meta-learning (MAML), 3) Multi-task learning-based



(a) Secrecy rates of scheduled users.



(b) Sum secrecy rate.

Figure 5.4: Testing samples are selected from taskset $\mathcal{T}^{\mathrm{F}}$ and $\mathcal{T}^{\mathrm{E}}$ in Table. 5.3.

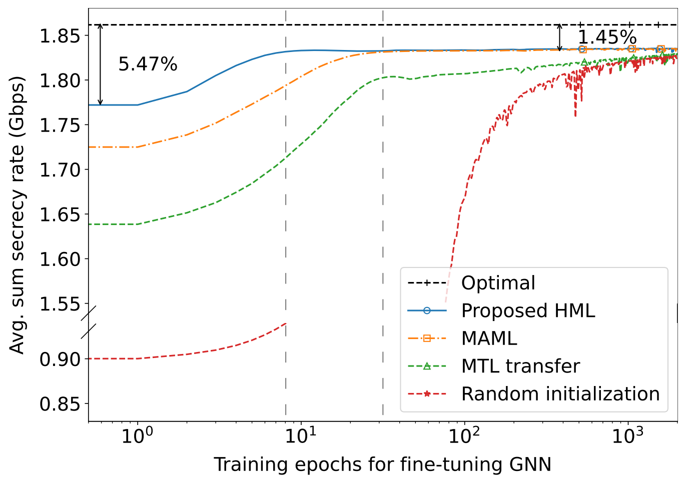
transfer learning (MTL Transfer), and 4) Random initialization.

- *Optimal*: The optimal solution is obtained by the iterative algorithm detailed in Section 5.4.1, and its optimality has been proved in [Dong *et al.*, 2021].
- *MAML*: MAML is one of the most widely used meta-learning algorithms, and its key ideas have been discussed in Section 5.4.4.

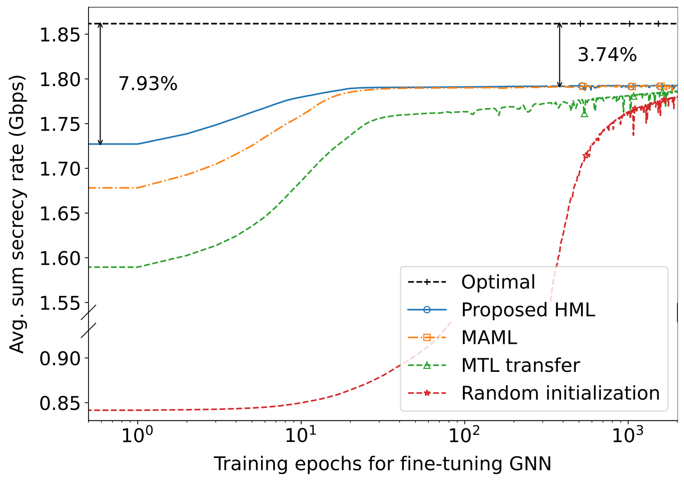
- *MTL Transfer*: Transfer learning improves the sample efficiency by fine-tuning the parameters of the pre-trained GNN in a task with fewer training samples. With multi-task learning (MTL), the initial performance is much better than random initialization as the GNN is pre-trained in multiple tasks [FINN *et al.*, 2017; YE *et al.*, 2020]. To execute MTL transfer learning, we only need to replace the initialization in line 2 of Algorithm 5 by the pre-trained parameters.
- *Random Initialization*: Random initialization is the conventional method that trains the GNN from scratch with a new task.

In figures 5.5–5.7, we compare our HML with the benchmark algorithms by analyzing the training epochs needed for convergence, the horizontal axis represents the training epochs used to fine-tune the GNN, and 32 samples from $\mathcal{T}^{\mathrm{F}}$ are used to train the GNN. The vertical axis represents the sum of the rewards of all the users, and the average is taken over samples, i.e., 1000 testing samples from $\mathcal{T}^{\mathrm{E}}$ are used. We refer to it as the average sum reward.

In Fig. 5.5, we consider the average sum of secrecy rates and illustrate the impacts of the number of users, channel models, and coding blocklength on the initial performance and sample efficiency of different methods. The results in Fig. 5.5 show that HML achieves the best initial average sum secrecy rate and the highest sample efficiency compared with all the benchmarks. In Fig. 5.5(a), HML can converge in 8 training epochs. Both MAML and MTL transfer learning takes more than 30 epochs to converge. Thus, HML can reduce the convergence time by up to 73%. After the fine-tuning, the gap between learning methods and the optimal solution is around 1.45%. In Fig. 5.5(b), the coding blocklength in meta-testing is also different from that in meta-training. As a result, the gap between the initial performance of



(a) Secrecy rate in long blocklength regime.

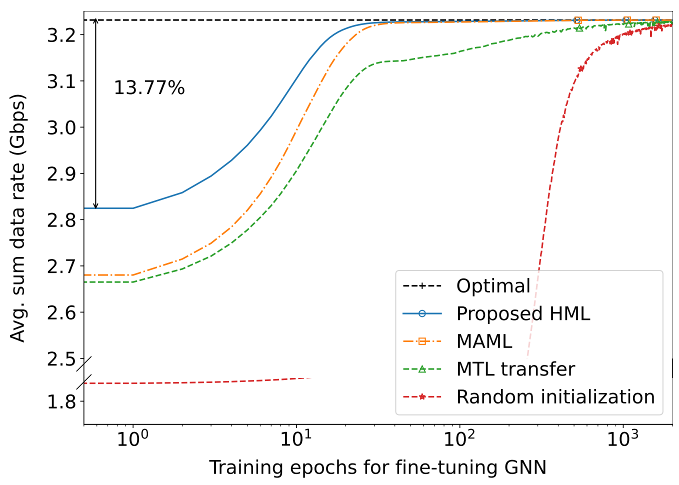


(b) Secrecy rate in short blocklength regime.

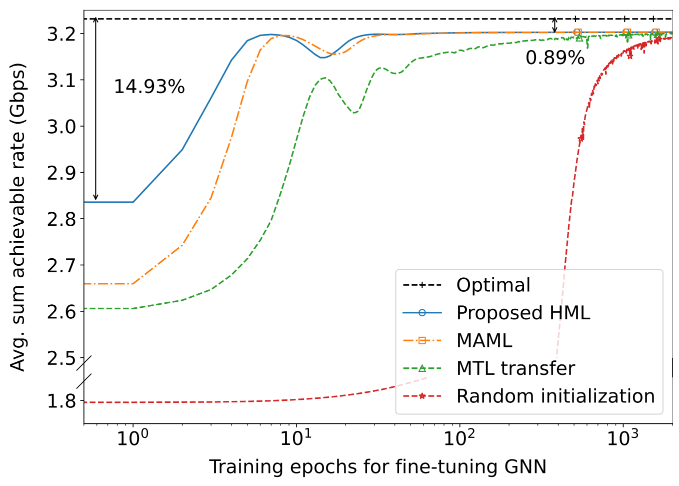
Figure 5.5: Meta-testing with unseen channel models.

HML and the optimal solution is 7.93%. After fine-tuning, the gap reduced to 3.74%, which is larger than the gap in Fig. 5.5(a), where the blocklength is the same in meta-training and meta-testing.

Fig. 5.6 shows the average sum of data rates achieved by different methods. The results indicate that when the reward function and the QoS constraint in meta-testing



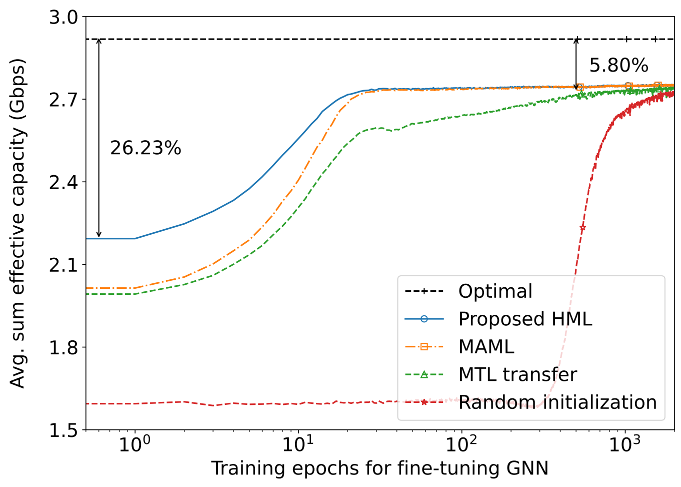
(a) Shannon capacity in long blocklength regime.



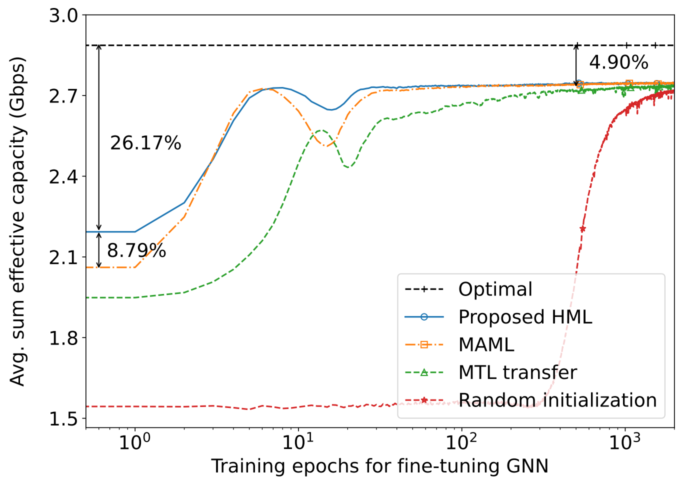
(b) Achievable rate in short blocklength regime.

Figure 5.6: Meta-testing with unseen QoS requirements of rate rates and unseen channels.

are different from that in meta-training, the gaps between the initial performance of HML and the optimal solution increase to 13.77% and 14.93% in long and short blocklength regimes, respectively. After fine-tuning, the gaps between the learning methods and the optimal solution are smaller than that in Fig. 5.5. This is because Shannon's capacity/achievable rate are two special cases of the secrecy rate in the



(a) Effective capacity in long blocklength regime.



(b) Effective capacity in short blocklength regime.

Figure 5.7: Meta-testing with unseen QoS requirements and unseen channels.

long/short blocklength regimes when the eavesdropped channels are in deep fading. It is easier to learn a good policy when the problem becomes less complicated.

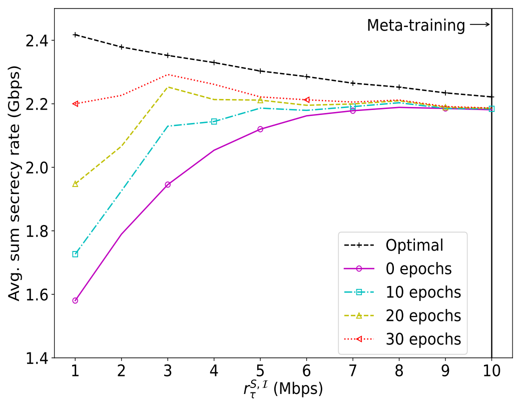
Fig. 5.7 shows the average sum of effective capacities achieved in the meta-testing stage, where the initial parameters of the GNN are obtained from meta-training, and the GNN is trained with tasks maximizing the sum secrecy rate in the long blocklength regime. In other words, the QoS requirement in meta-testing is queuing delay requirement, which is quite different from the security requirement in meta-training. By comparing the results in Figs. 5.7 and 5.5, we can observe that the gaps between the HML and the optimal solution in Fig. 5.7 are larger than the gaps in Fig. 5.5. Nevertheless, HML can still converge in around 10 to 30 epochs and outperforms the other benchmarks in Fig. 5.7.

Meta-Testing with Different System Parameters

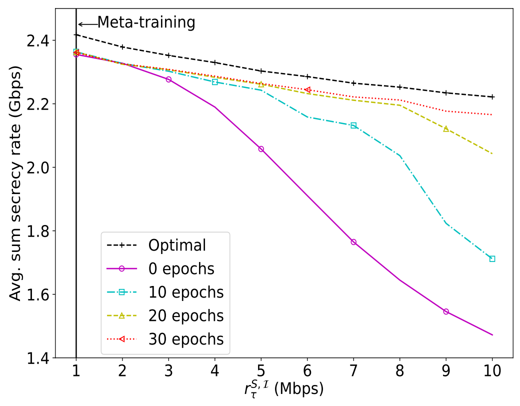
In this part, we focus on secrecy rates in the long blocklength regime in both meta-training and meta-testing, and change the values of $r_{\tau}^{S,\mathcal{I}}$, $W_{\tau}^{S,\mathcal{I}}$, and $U_{\tau}^{S,\mathcal{I}}$ to investigate their impacts on the initial performance and sample efficiency of HML in meta-testing.

In Fig. 5.8, we evaluate the initial performance and sample efficiency with different $r_{\tau}^{S,\mathcal{I}}$ in support sets and query sets in meta-training. Specifically, we set $r_{\tau}^{S,\mathcal{I}}$ to 10 Mbps and 1 Mbps in meta-training in Figs. 5.8(a) and 5.8(b), respectively. In Fig. 5.8(c), $r_{\tau}^{S,\mathcal{I}}$ is randomly selected from the set $\{1, \cdots, 10\}$ Mbps in meta-training. In meta-testing, we increase $r_{\tau}^{S,\mathcal{I}}$ from 1 Mbps to 10 Mbps. The results in Figs. 5.8(a) and 5.8(b) indicate that the gaps between zero-shot learning (with 0 training epochs in meta-testing) and the optimal solution increase with the difference between $r_{\tau}^{S,\mathcal{I}}$ in meta-training and $r_{\tau}^{S,\mathcal{I}}$ in meta-testing. To increase the generalization ability, we can increase the diversity of tasks in meta-training as shown in Fig. 5.8(c). In this way, our GNN is near-optimal with zero-shot learning.

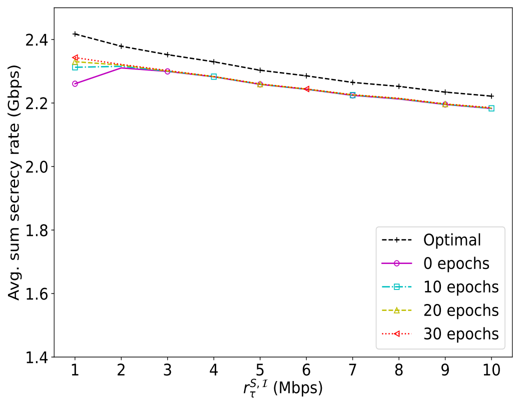
In Fig. 5.9, we validated the generalization ability of our GNN with dynamic bandwidth $W_{\tau}^{S,\mathcal{I}}$. In meta-training, $W_{\tau}^{S,\mathcal{I}}$ is randomly selecting from the set $\{10, \cdots, 100\}$ MHz. In meta-testing, we increase $W_{\tau}^{S,\mathcal{I}}$ from 10 to 100 MHz. The results in Fig. 5.9 show



(a) $r_\tau^{S,\mathcal{I}} = 10$ Mbps in meta-training.



(b) $r_\tau^{S,\mathcal{I}} = 1$ Mbps in meta-training.



(c) $r_\tau^{S,\mathcal{I}} \in \{1, \cdots, 10\}$ Mbps in meta-training.

Figure 5.8: Meta-testing with dynamic secrecy rate requirements, $r_\tau^{S,\mathcal{I}} \in \{1, \cdots, 10\}$ Mbps, where$W_\tau^{S,\mathcal{I}} = 100$ MHz and $U_\tau^{S,\mathcal{I}} = 10$.

that our GNN is near-optimal with different values of $W_\tau^{S,\mathcal{I}}$. In Fig. 5.10, we further validate the generalization ability of our GNN with different numbers of users. In meta-training, the number of total users is randomly selected, $U_\tau^{S,\mathcal{I}} \in \{10, 11, ..., 30\}$. In meta-testing, we increase the number of total users from 5 to 50. The results in Fig. 5.10 show that the proposed HML can obtain a GNN that has strong generalization ability with different numbers of users. The gap between the GNN and the optimal policy increases slightly with $U_\tau^{S,\mathcal{I}}$. This is because the scale of the problem

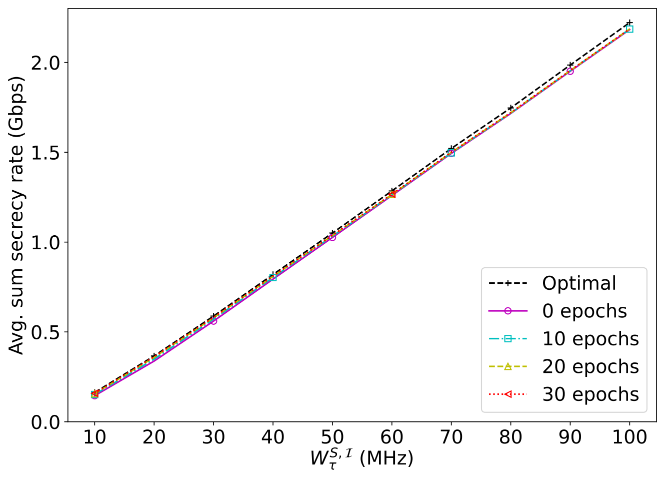


Figure 5.9: Meta-testing with dynamic bandwidth $W_\tau^{S,\mathcal{I}} \in \{10, \cdots, 100\}$ MHz in meta-training, where $r_\tau^{S,\mathcal{I}} = 10$ Mbps and $U_\tau^{S,\mathcal{I}} = 10$.

increases with $U_\tau^{S,\mathcal{I}}$, and it is more difficult to learn the bandwidth allocation policy of a large-scale problem compared with that of a small-scale problem.

5.6 Chapter Summary

In this paper, we developed an HML approach to train a GNN-based scalable bandwidth allocation policy that can generalize well in various communication scenarios, including different number of users, wireless channels, QoS requirements, and bandwidth. The main idea is to train the initial parameters of the GNN with various tasks in meta-training, and then fine-tune the parameters with a few samples in meta-testing. Simulation results showed that the performance gap between the GNN and the optimal policy obtained by an iterative algorithm is less than 5% in most of the cases. For unseen communication scenarios, the GNN can converge in 10 to 30 training epochs, which are much faster than the existing benchmarks. Our approach can be extended beyond bandwidth allocation, such as power allocation, precoding,

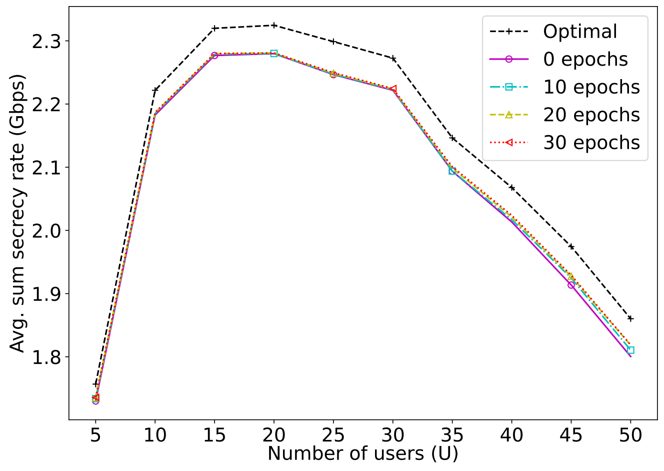


Figure 5.10: Meta-testing with different numbers of users $U_{\tau}^{S,\mathcal{I}} \in \{5, 10, \cdots, 50\}$, where $r_{\tau}^{S,\mathcal{I}} = 10$ Mbps and $W_{\tau}^{S,\mathcal{I}} = 100$ MHz.

and repetitions. Nevertheless, the featuring engineering and the structure of GNN in other scenarios deserve further investigation.

Chapter 6

Conclusions and Future Work

In this thesis, we focused on developing secure and intelligent resource allocation optimization frameworks for low-latency wireless communications. The contributions, insights, and possible extensions in future work are provided in this chapter.

6.1 Summary of Contributions and Insights

We proposed double blockchain (DBC), blockchain-secured deep reinforcement learning (BC-DRL), and hybrid-task meta-learning (HML) to highlight new insights for developing novel resource allocation schemes in low-latency wireless communication networks.

First, we aimed to reduce the latency by optimizing the resources of the nodes in a cloud-fog IoT network, in which we proposed a scalable DBC architecture composing a cloud-layer information blockchain (IBC) and an fog-layer reputation blockchain (RBC). The IBC was managed by the cloud server located in the cloud layer, whilst the RBC was managed by the base stations (BSs) randomly located in the fog layer. The scalable DBC addressed the high resource requirement issue of conventional single blockchain by utilizing the proposed mapping protocol and node classification algorithms. In addition, we provided stochastic analysis for the proposed DBC architecture, which demonstrated the effectiveness of applying DBC in large-scale wireless IoT networks with low-latency and tampering-resistant requirements.

Next, we focused on the edge layer, where multiple randomly located BSs provided MEC services for multiple users. The target was to minimize the transmission and computing latency subject to a maximum threshold on the denial-of-service (DoS) probability in a secure manner. To achieve this goal, we proposed to embed blockchain into the decentralized MEC network. Specifically, we designed a constrained deep reinforcement learning (DRL) algorithm to optimize the computing resources for blockchain management and MEC service-provisioning. Simulation results validated that the proposed BC-DRL framework can effectively resist data tampering attacks, satisfy the time-varying user requests, and handle the dynamic requirement of DoS probability.

Finally, we extended the wireless network's quality-of-service (QoS) requirement beyond low-latency, incorporating high data rates and enhanced security through network slicing (NS) technology. The wireless network comprised different slices supporting diverse QoS requirements. Within each slice, we further considered the issue of dynamic numbers of requesting users, a challenging problem in wireless networks. To achieve the scalable resource allocation policy, a scalable graph neural network (GNN) was designed. The GNN adapted its topology according to the dynamics of user numbers. Subsequently, the HML algorithm was proposed to address the low training efficiency issue caused by the non-stationary wireless channels. The combination of GNN and HML generalized the bandwidth allocation to perform in a scalable and transferable manner.

6.2 Future Work

With the rapid development of wireless and machine learning technologies, we have identified several important emerging requirements in the area of low-latency wireless communications. To satisfy these requirements, we outline several potential extensions to the work presented in this thesis.

6.2.1 Network Slicing

In this thesis, we focused on resource allocation for computation and bandwidth in Chapters 4 and 5. In real-world wireless networks, further performance improvements could be achieved by optimizing critical wireless resources, such as transmit power and base station (BS) selection. We can add the transmit power in the policy spaces of BC-DRL and the scalable GNN frameworks, followed by joint optimization of the bandwidth and transmit power. In addition, in the decentralized wireless network considered in BC-DRL, we can include a machine learning-based BS selection algorithm to further reduce the complexity and improve the robustness in dynamic environments.

An end-to-end learning model can be utilized to reduce the latency of signal processing [Kotary *et al.*, 2021]. Therefore, it can be interesting to develop an end-to-end learning algorithm that achieves the same goal of scalable and transferable bandwidth allocation as that in HML, but with lower latency. A potential solution is to use a cascaded neural network [Dong *et al.*, 2021], composing both the user scheduling and bandwidth allocation algorithms.

6.2.2 Wireless Security

In Chapter 5, we investigated the scenario where a single base station (BS) serves multiple users in the presence of one eavesdropper. It would be interesting to extend this single eavesdropper model to multiple eavesdroppers trying to eavesdrop the confidential information transmitted in the wireless network. A potential solution is to represent the wireless network with multiple users and multiple eavesdroppers as a bipartite graph, in which the vertices are divided into the user vertices and the eavesdropper vertices. Based on the represented structure of the bipartite graph, we can optimize the resource allocation according to the features of these two kinds of vertices.

In [Hao *et al.*, 2023], we have demonstrated that the CSI uncertainty can greatly affect the value of the secrecy rate. It would be interesting to include the channel state information (CSI) uncertainty when evaluating the data rate of the eavesdropper in Chapter 5. Autoencoder has demonstrated its potency as a formidable tool capable of reconstructing original information through training on datasets with and without CSI uncertainty. Therefore, it would be possible to consider enhancing the secrecy rate by designing an autoencoder when the accurate CSI of the eavesdropper is hard to achieve.

6.2.3 Integrated Terrestrial and Non-Terrestrial Network

In future 6G communication systems, non-terrestrial networks, including UAVs and satellite networks, will serve as a complement to the current terrestrial network. As such, it would be interesting to expand our current system model from a terrestrial network to an integrated terrestrial and non-terrestrial (T-NT) network, and investigate the secure and intelligent resource allocation accordingly.

A new integrated blockchain will need to be developed for integrated T-NT networks. In Chapter 3, we investigated the DBC blockchain architecture of the cloud-edge wireless network, which comprises cloud servers, edge servers, and IoT devices. However, the resources of nodes in the non-terrestrial network cover a much wider range compared with terrestrial nodes. Thus, how to design the integrated blockchain for secure node management for the T-NT network with maximum resource utilization efficiency will be a challenging problem with possible solutions arising from multi-type blockchain data scheduling, which was analyzed in [Hu *et al.*, 2023].

In addition to considering more categories of resources, it can be interesting to develop novel machine learning-based resource allocation for the T-NT network composed of nodes with resource amounts across a wider range than the terrestrial networks of IoT, MEC, and NS networks.

Lastly, to enhance the flexibility and applicability, we could explore how our model adapts to various point processes [Hourani *et al.*, 2016]. Understanding how different point processes affect resource allocation and network performance is crucial for optimizing system design across diverse deployment scenarios. This research avenue promises to advance wireless communication systems by addressing evolving network needs.

Appendix A

Appendix for Chapter 3

A.1 Derivation of Blockchain Processing Latency

In the pre-prepare step, the processing latency is expressed as

$$\tau^{\mathrm{g}}_{\mathrm{bc},i}(t) = \frac{f^{\mathrm{g}}_{\mathrm{bc},i}(t)}{a_i(t)} + \max_{i \in \mathcal{N}_{\mathrm{M}}(t), j \in \mathcal{N}_{\mathrm{V}}(t)} \left\{ \frac{\ell_{\mathrm{b}}(t)}{W_{i,j}} \right\}, \quad \text{(slots)} \tag{A.1.1}$$

where the first term is the block miner signature generation time and the second term is the maximum block multicasting time from the miner BS $i \in \mathcal{N}_{\mathrm{M}}(t)$ to the validator BSs $j \in \mathcal{N}_{\mathrm{V}}(t)$, where $W_{i,j}$ is the data transmission rate from the i-th BS to the j-th BS.

In the prepare step, each validator BS adds their signature to the received block from the miner BS. The CPU cycles required for the j-th validator in the t-th time slot can be calculated by $f^{\mathrm{v}}_{\mathrm{bc},j}(t) = \mathbb{1}\{j \in \mathcal{N}_{\mathrm{V}}(t)\} \cdot \kappa_{\mathrm{bc}} \ell_{\mathrm{b}}(t)$ (CPU cycles). Each validator BS multicasts the block to all the other committee BSs after validating the signature in the block header. Thus, the processing latency is derived as

$$\tau^{\mathrm{v}}_{\mathrm{bc},i}(t) = \max_{j \in \mathcal{N}_{\mathrm{V}}(t)} \left\{ \frac{f^{\mathrm{v}}_{\mathrm{bc},j}(t)}{a_j(t)} \right\} + \max_{j \in \mathcal{N}_{\mathrm{V}}(t), k \in \mathcal{N}_{\mathrm{C}}(t), k \neq j} \left\{ \frac{\ell_{\mathrm{b}}(t)}{W_{j,k}} \right\}, \quad \text{(slots)} \tag{A.1.2}$$

where the first term is the maximum signature validating time amongst all validator BSs $j \in \mathcal{N}_{\mathrm{V}}(t)$, and the second term is the maximum block multicasting time to all committee BSs $k \in \mathcal{N}_{\rm C}(t)$.

Finally, in the commit step, the block is multicasted to all other BSs for secure storage in the blockchain. The processing latency is given by

$$\tau^{\rm c}_{{\rm bc},i}(t) = \max_{k\in\mathcal{N}_{\rm C}(t)} \left\{ \frac{f^{\rm c}_{{\rm bc},k}(t)}{a_k(t)} \right\} + \max_{k\in\mathcal{N}_{\rm C}(t), l\in\mathcal{N}_{\rm B}, l\neq k} \left\{ \frac{\ell_{\rm b}(t)}{W_{k,l}} \right\}, \quad \text{(slots)} \tag{A.1.3}$$

where the first term is the maximum signature validating time amongst all committee BSs $k \in \mathcal{N}_{\rm C}(t)$ and the second term is the maximum multicasting time to all BSs $l \in \mathcal{N}_{\rm B}$.

A.2 Derivation of Service Rate Allocated in One Time Slot

Based on (4.5.4), the processing latency of the service rates allocated in the t'-th time slot is given by

$$\tau_i(t') = \lceil \tau_{{\rm bc},i}(t') + \tau_{{\rm sp},i}(t') \rceil, \quad \text{(slots)} \tag{A.2.1}$$

where $\lceil\cdot\rceil$ is the ceiling function. In the t-th time slot, the remaining processing latency of the service rates allocated in the t'-th time slot can be expressed as

$$\hat{\tau}_i(t,t') = (\tau_i(t') - (t-t'))^+, \quad \text{(slots)}. \tag{A.2.2}$$

In the t-th time slot, the service rates been hold equal to the service rates allocated in the t'-th time slot, and can be described as

$$\hat{a}_i(t,t') = a_i(t') \cdot \mathbb{1}\{\hat{\tau}_i(t,t') > 0\}, \quad \text{(CPU cycles/slot)} \tag{A.2.3}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function, which equals one when $\hat{\tau}_i(t,t') > 0$, and equals zero otherwise.

The state of the service rates allocated in the t'-th time slot is defined as its remaining processing latency and the service rate hold by it, i.e.,

$$\hat{\boldsymbol{s}}_i\left(t, t'\right) = \left[\hat{\tau}_i(t, t'), \hat{a}_i(t, t')\right]. \tag{A.2.4}$$

If the BS did not allocate any service rates in the t'-th time slot, both $\hat{\tau}_i(t, t')$ and $a_i(t, t')$ are zeros. If the initialized processing latency $\tau_i(t')$ is smaller than $T_{\max}$, then $\hat{\tau}_i(t, t')$ and $\hat{a}_i(t, t')$ are also updated to zeros before $t' + T_{\max}$.

A.3 Proof of Equivalence

We prove the equivalence of problem (4.5.1) and problem (4.5.12) by proving the equivalence of the objective function and the constraint, respectively.

For the equivalence of the objective function, since our considered stochastic processes are stationary and ergodic, we have $\mathbb{E}[\tau_i(t)] = \mathbb{E}[\tau_i(\hat{t})]$, where $\hat{t} \neq t$. Thus, the objective function can be derived by

$$\begin{aligned}
\max_{\mu(\cdot)} R_{i,\mu}(t) &= \max_{\mu(\cdot)} \mathbb{E}_\mu \left[\sum_{\hat{t}=t}^{\infty} \gamma_r^{\hat{t}-t} r_i(t)\right] \\
&= \max_{\mu(\cdot)} \mathbb{E}_\mu \left[\frac{r_i(t)}{1-\gamma_r}\right] \\
&= \max_{a_i(t)} \mathbb{E}_\mu[-\tau_i^{\mathrm{a}}(t)] \\
&= \min_{a_i(t)} \mathbb{E}[\tau_i(t)].
\end{aligned} \tag{A.3.1}$$

For the equivalence of the constraint, we take eq. (4.5.10) and $\mathcal{E}_{\max} = \epsilon_{\max}/(1-\gamma_c)$ into eq. (4.5.11), we have

$$C_{i,\mu}(t) = \frac{\mathbb{E}_\mu[c_i(t)]}{1-\gamma_c} \leq \frac{\epsilon_{\max}}{1-\gamma_c}. \tag{A.3.2}$$

Thus, we have $\mathbb{E}_\mu[c_i(t)] \leq \epsilon_{\max}$, which is mathematically equivalent to the constraint in problem (4.5.1).

In conclusion, the objective function and the constraint of problem (4.5.1) and problem (4.5.12) are all equivalent, thus these two problems are equivalent. This completes the proof. □

A.4 Proof of the Markov Property

In the t-th time slot, the action of the i-th BS is assigned to provide its physical service rate denoted by $a_i(t)$. As shown in (4.5.7), the state of the i-th BS in the $(t+1)$-th time slot is

$$\hat{\boldsymbol{s}}_i(t+1) = \begin{bmatrix} \boldsymbol{s}_i(t+1, t+1-T_{\max}) \\ \boldsymbol{s}_i(t+1, t+1-T_{\max}+1) \\ \vdots \\ \boldsymbol{s}_i(t+1, t+1) \end{bmatrix}^{\mathrm{T}}, \tag{A.4.1}$$

where the dimension of the state in (A.4.1) is $T_{\max}$. Since $s_i(t, t')$ beyond $T_{\max}$ are uncorrelated. Thus, the next state, $\hat{s}_i(t+1)$, only depend on the state and action in the t-th time slot. Therefore, the Markov property holds. This completes the proof. □

Appendix B

Appendix for Chapter 4

B.1 Proof of Concavity for Secrecy Rate in Long Blocklength Regimes

To prove the concavity of the secrecy rate in long blocklength regimes, we only need to prove that the second derivative of the secrecy rate is positive. We first calculate the partial derivative of the secrecy rate of the k-th scheduled user as follows,

$$\begin{aligned}
\frac{\partial r_k^{S,\mathcal{I}}(w_{\tau,k}^{D,\mathcal{I}})}{\partial w_{\tau,k}^{D,\mathcal{I}}} =& \frac{\partial \left(r_k^{D,\mathcal{I}}(w_{\tau,k}^{D,\mathcal{I}}) - r_k^{e,\mathcal{I}}(w_{\tau,k}^{D,\mathcal{I}}) \right)}{\partial w_k} \\
=& \frac{-\zeta_k + \ln\left(1 + \frac{\zeta_k}{w_{\tau,k}^{D,\mathcal{I}}}\right)(w_{\tau,k}^{D,\mathcal{I}} + \zeta_k)}{\ln(2)(w_{\tau,k}^{D,\mathcal{I}} + \zeta_k)} - \frac{-\zeta_k^e + \ln\left(1 + \frac{\zeta_k^e}{w_{\tau,k}^{D,\mathcal{I}}}\right)(w_{\tau,k}^{D,\mathcal{I}} + \zeta_k^e)}{\ln(2)(w_{\tau,k}^{D,\mathcal{I}} + \zeta_k^e)} \\
=& \frac{\ln\left(\frac{w_{\tau,k}^{D,\mathcal{I}} + \zeta_k}{w_{\tau,k}^{D,\mathcal{I}} + \zeta_k^e}\right)}{\ln(2)} + \frac{(\zeta_k^e - \zeta_k) w_{\tau,k}^{D,\mathcal{I}}}{\ln(2)(w_{\tau,k}^{D,\mathcal{I}} + \zeta_k)(w_{\tau,k}^{D,\mathcal{I}} + \zeta_k^e)},
\end{aligned} \tag{B.1.1}$$

where $\zeta_k = P_k h_k / N0$ and $\zeta_k^e = P_k h_k^e / N0$. Since the secrecy rate of the user increases with the increasing of the allocated bandwidth, we have $\partial r_k^{S,\mathcal{I}}(w_{\tau,k}^{D,\mathcal{I}}) / \partial w_{\tau,k}^{D,\mathcal{I}} < 0$. The second derivative of $r_k^{S,\mathcal{I}}(w_{\tau,k}^{D,\mathcal{I}})$ can be derived as follows,

$$\begin{aligned}\frac{\partial^2 r_k^{S,\mathcal{I}}(w_{\tau,k}^{D,\mathcal{I}})}{\partial {w_{\tau,k}^{D,\mathcal{I}}}^2} &= \frac{\partial}{\partial w_{\tau,k}^{D,\mathcal{I}}}\left(\frac{\partial r_k^{S,\mathcal{I}}(w_{\tau,k}^{D,\mathcal{I}})}{\partial w_{\tau,k}^{D,\mathcal{I}}}\right) \\ &= \frac{(\zeta_k^e - \zeta_k)\left((\zeta_k^e + \zeta_k)w_{\tau,k}^{D,\mathcal{I}} + 2\zeta_k^e\zeta_k\right)}{\ln(2)(w_{\tau,k}^{D,\mathcal{I}} + \zeta_k)^2(w_{\tau,k}^{D,\mathcal{I}} + \zeta_k^e)^2}.\end{aligned} \tag{B.1.2}$$

For any scheduled user, we have $\zeta_k > \zeta_k^e$. Thus, $\partial^2 r_k^{S,\mathcal{I}}(w_{\tau,k}^{D,\mathcal{I}})/\partial {w_{\tau,k}^{D,\mathcal{I}}}^2 < 0$. Therefore, $r_k^{S,\mathcal{I}}(w_{\tau,k}^{D,\mathcal{I}})$ is concave. This completes the proof. □

Bibliography

Afolabi, I., T. Taleb, K. Samdanis, A. Ksentini and H. Flinck [2018]. "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 3, pp. 2429–2453.

Alrubei, S. M., E. A. Ball, J. M. Rigelsford, and C. A. Willis [2020]. "Latency and performance analyses of real-world wireless IoT-blockchain application," *IEEE Sens. J*, vol. 20, no. 13, pp. 7372–7383.

Alrubei, S., E. Ball, and J. Rigelsford [2021]. "The use of blockchain to support distributed AI implementation in IoT systems," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14790–14802.

Alsenwi, M., N. H. Tran, M. Bennis, S. R. Pandey, A. K. Bairagi, and C. S. Hong [2021]. "Intelligent resource slicing for eMBB and URLLC coexistence in 5G and beyond: A deep reinforcement learning based approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4585–4600.

Andrews, J. G., F. Baccelli, and R. K. Ganti [2011]. "A tractable approach to coverage and rate in cellular networks," *IEEE Trans. Commun.*, vol. 59, no. 11, pp. 3122–3134.

Andrychowicz, M., M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, N. Freitas [2016]. "Learning to learn by gradient descent by gradient descent," in *Proc. 30th Conf. Neural Inf. Process. Syst. (NIPS 2016)*, Barcelona, Spain.

Asheralieva, A. and D. Niyato [2020]. "Reputation-based coalition formation for secure self-organized and scalable sharding in IoT blockchains with mobile-edge computing," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11830–11850.

Boyd, S. and L. Vandenberghe [2004]. *Convex Optimization.* New York, NY, USA: Cambridge Univ. Press.

Busoniu, L., R. Babuska, B. De Schutter, and D. Ernst [2010]. *Reinforcement learning and dynamic programming using function approximators.* Boca Raton, FL, USA: CRC Press.

Cao, B., Z. Wang, L. Zhang, D. Feng, M. Peng, L. Zhang, and Z. Han [2023]. “Blockchain systems, technologies, and applications: A methodology perspective,” *IEEE Commun. Surv. Tutorials*, vol. 25, no. 1, pp. 353–385.

Castro, M. and B. Liskov [1999]. “Practical Byzantine fault tolerance,” in *Proc. USENIX Symp. Oper. Syst. Design Implement.*, vol. 99, pp. 173–186.

Chekired, D. A., M. A. Togou, L. Khoukhi, and A. Ksentini [2019]. “5G-slicing-enabled scalable SDN core network: Toward an ultra-low latency of autonomous driving service,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1769–1782.

Chen, Z., B. Yin, H. Zhu, Y. Li, M. Tao, and W. Zhang [2022]. “Mobile communications, computing, and caching resources allocation for diverse services via multi-objective proximal policy optimization,” *IEEE Trans. Commun.*, vol. 70, no. 7, pp. 4498–4512.

Chowdhury, M. Z., M. Shahjalal, S. Ahmed, and Y. M. Jang [2020]. “6G wireless communication systems: Applications, requirements, technologies, challenges, and research directions,” *IEEE Open J. Commun. Soc.*, vol. 1, pp. 957–975.

Cui, Y., D. Jiang, and Y. Wu [2016]. “Analysis and optimization of caching and multicasting in large-scale cache-enabled wireless networks,” *IEEE Trans. Wireless Commun.*, vol. 15, no. 7, pp. 5101–5112.

Dakic K., C. C. Chan, B. A. Homssi, K. Sithamparanathan, and A. Al-Hourani [2023]. “On delay performance in mega satellite networks with inter-satellite links,” in *Proc. Global Commun. Conf. (GLOBECOM)*, Kuala Lumpur, Malaysia, 2023, pp. 4896-4901.

Dang, T., C. Liu, and M. Peng [2023]. “Low-latency mobile virtual reality content delivery for unmanned aerial vehicle-enabled wireless networks with energy constraints,” *IEEE Trans. Veh. Technol.*, vol. 72, no. 2, pp. 2189–2201.

Debe, M., K. Salah, M. H. U. Rehman, and D. Svetinovic [2019]. “IoT public fog nodes reputation system: A decentralized solution using Ethereum blockchain,” *IEEE Access*, vol. 7, pp. 178082–178093.

Deng, T., X. Liu, H. Zhou, and V. C. M. Leung [2022]. “Global resource allocation for high throughput and low delay in high-density VANETs,” *IEEE Trans. Wireless Commun.*, vol. 21, no. 11, pp. 9509–9518.

Ding, Q., S. Gao, J. Zhu, and C. Yuan [2020]. “Permissioned blockchain-based double-layer framework for product traceability system,” *IEEE Access*, vol. 8, pp. 6209–6225.

Ding, G., J. Yuan, G. Yu, and Y. Jiang [2022]. “Two-timescale resource management for ultrareliable and low-latency vehicular communications,” *IEEE Trans. Commun.*, vol. 70, no. 5, pp. 3282–3294.

Do, T. T., T. J. Oechtering, S. M. Kim, M. Skoglund, and G. Peters [2017]. “Uplink waveform channel with imperfect channel state information and finite constellation Input,” *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1107–1119.

Dong, R., C. She, W. Hardjawana, Y. Li, and B. Vucetic [2019]. “Deep learning for hybrid 5G services in mobile edge computing systems: Learn from a digital twin,” *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4692–4707.

Dong, R., C. She, W. Hardjawana, Y. Li, and B. Vucetic [2021]. “Deep learning for radio resource allocation with diverse quality-of-service requirements in 5G,” *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2309–2324.

Douceur, J. R. [2002]. “The sybil attack,” in *Proc. Int. Workshop Peer-to-Peer Syst.*, pp. 251–260.

Eisen, M., C. Zhang, L. F. O. Chamon *et al.* [2019]. Learning optimal resource allocations in wireless systems,” *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2775–2790.

Feriani, A. and E. Hossain [2021]. “Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial,” *IEEE Commun. Surv. Tutorials*, vol. 23, no. 2, pp. 1226–1252.

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

Feng, J., F. Richard Yu, Q. Pei, X. Chu, J. Du, and L. Zhu [2020a]. “Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach,” *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6214–6228.

Feng, J., F. R. Yu, Q. Pei, J. Du, and L. Zhu [2020b]. “Joint optimization of radio and computational resources allocation in blockchain-enabled mobile edge computing systems,” *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 4321–4334, Jun. 2020.

Filho, P. F. A., G. Kaddoum, D. R. Campelo, A. G. Santos, D. Macêdo, and C. Zanchettin [2021]. “Intrusion detection for cyber–physical systems using generative adversarial networks in fog environment,” *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6247–6256.

Finn C., P. Abbeel, and S. Levine [2017]. “Model-agnostic meta-learning for fast adaptation of deep networks,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, Sydney, Australia, pp. 1126–1135.

Garadi, M. A. A., A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani [2020]. “A survey of machine and deep learning methods for Internet of Things (IoT) security,” *IEEE Commun. Surv. Tutorials*, vol. 22, no. 3, pp. 1646–1685.

Gao, W., C. Han, and Z. Chen [2020] "Receiver artificial noise aided terahertz secure communications with eavesdropper in close proximity," in *Proc. Global Commun. Conf. (GLOBECOM)*, Taipei, Taiwan.

Gilmer, J., S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl [2017]. "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Sydney, Australia, pp. 1263–1272.

Goldsmith A. [2005]. *Wireless communications.* Cambridge, U.K.: Cambridge Univ. Press.

Gopala, P. K., L. Lai, and H. El Gamal [2008] "On the secrecy capacity of fading channels" *IEEE Trans. Inf. Theory*, vol. 54, no. 10, pp. 4687–4698.

Gu, Y., C. She, Z. Quan, C. Qiu, and X. Xu [2023]. "Graph neural networks for distributed power allocation in wireless networks: Aggregation over-the-air," *IEEE Trans. Wireless Commun.*, Early access.

Guo, J. and C. Yang [2022]. "Learning power allocation for multi-cell-multi-user systems with heterogeneous graph neural networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 2, pp. 884–897, Feb. 2022.

Guo, J. and C. Yang [2023]. "Deep neural networks with data rate model: Learning power allocation efficiently," *IEEE Trans. Commun.*, vol. 71, no. 3, pp. 1447–1461, Mar. 2023.

Guo, A. and M. Haenggi [2015]. "Asymptotic deployment gain: A simple approach to characterize the SINR distribution in general cellular networks," *IEEE Trans. Commun.*, vol. 63, no. 3, pp. 962–976.

Guo, F., F. R. Yu, H. Zhang, H. Ji, M. Liu, and V. C. M. Leung [2020]. "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1689–1703.

Guo, C., L. Liang, and G. Y. Li [2019]. "Resource allocation for vehicular communications with low latency and high reliability," *IEEE Trans. Wireless Commun.*, vol. 18, no. 8, pp. 3887–3902.

Gupta, A. K., X. Zhang, and J. G. Andrews [2015]. "SINR and throughput scaling in ultra-dense urban cellular networks," *IEEE Wireless Commun. Lett.*, vol. 4, no. 6, pp. 605–608.

Haenggi, M [2005]. "On distances in uniformly random networks," *IEEE Trans. Inf. Theory*, vol. 51, no. 10, pp. 3584–3586.

Haenggi, M [2013]. *Stochastic geometry for wireless networks.* Cambridge, U.K.: Cambridge Univ. Press.

HAN, B., V. SCIANCALEPORE, X. COSTA-PÉREZ, D. FENG, AND H. D. SCHOTTEN [2020]. "Multiservice-based network slicing orchestration with impatient tenants," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 5010–5024.

HAO, X., P. L. YEOH, T. WU, Y. YU, Y. LI, AND B. VUCETIC [2021]. "Scalable double blockchain architecture for IoT information and reputation management," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, New Orleans, LA, USA, pp. 171–176.

HAO, X., P. L. YEOH, Z. JI, Y. YU, B. VUCETIC, AND Y. LI [2022]. "Stochastic analysis of double blockchain architecture in IoT communication networks," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9700–9711.

HAO, X., P. L. YEOH, Y. LIU, C. SHE, B. VUCETIC, AND Y. LI [2023] "Graph neural network-based bandwidth allocation for secure wireless communications," in *Proc. 2023 IEEE Int. Conf. on Commun. Workshops (ICC workshops)*, Rome, Italy, pp. 332–337, 2023.

HE, D., C. LIU, H. WANG, AND T. Q. S. QUEK [2019]. "Learning-based wireless powered secure transmission," *IEEE Wireless Commun. Lett.*, vol. 8, no. 2, pp. 600–603.

HEILMAN, E., A. KENDLER, A. ZOHAR, AND S. GOLDBERG [2015]. "Eclipse attacks on bitcoin's peer-to-peer network," in *Proc. USENIX Security Symp.*, Washington, D. C., USA, Aug. 2015, pp. 129–144.

HOMSSI B. A. ET AL. [2023]. "Artificial intelligence techniques for next-generation massive satellite networks," *IEEE Commun. Mag.*, Early access.

HOURANI A. A. AND M. HAENGGI, "Performance of next-generation cellular networks guarded with frequency reuse distance," *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 7277–7287.

HOURANI A. A. R. J. EVANS, AND S. KANDEEPAN, "Nearest neighbor distance distribution in hard-core point processes," *IEEE Communications Letters*, vol. 20, no. 9, pp. 1872–1875.

HU W., Y. YU, X. HAO, P. L. YEOH, L. GUO, AND Y. LI, "A cost-effective multi-type data scheduling for blockchain in massive Internet of UAVs," *IEEE Internet Things J.*. Accepted.

HU, Y., M. CHEN, W. SAAD, H. V. POOR AND S. CUI [2020]. "Meta-reinforcement learning for trajectory design in Wireless UAV networks," GLOBECOM 2020 - 2020 IEEE Global Communications Conference, Taiwan, China, pp. 1–6.

HUANG, C., Z. WANG, H. CHEN, Q. HU, Q. ZHANG, W. WANG, AND X. GUAN [2021a]. "RepChain: A reputation based secure, fast and high incentive blockchain system via sharding," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4291–4304.

Huang, L., L. Zhang, S. Yang, L. P. Qian, and Y. Wu [2021a]. “Meta-learning based dynamic computation task offloading for mobile edge computing networks,” *IEEE Commun. Lett.*, vol. 25, no. 5, pp. 1568–1572.

Ji, Z., P. L. Yeoh, G. Chen, C. Pan, Y. Zhang, Z. He, H. Yin, and Y. Li [2021]. “Random shifting intelligent reflecting surface for OTP encrypted data transmission,” *IEEE Wireless Commun. Lett.*, vol. 10, no. 6, pp. 1192–1196.

Jiang, F., K. Wang, L. Dong, C. Pan, and K. Yang [2020]. “Stacked autoencoder-based deep reinforcement learning for online resource scheduling in large-scale MEC networks,” *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9278–9290.

Kang, H., J. Joung, J. Ahn, and J. Kang [2019a]. “Secrecy-aware altitude optimization for quasi-static UAV base station without eavesdropper location information,” *IEEE Commun. Lett.*, vol. 23, no. 5, pp. 851–854.

Kang, J., Z. Xiong, D. Niyato, D. Ye, D. I. Kim, and J. Zhao [2019b]. “Toward secure blockchain-enabled internet of vehicles: Optimizing consensus management using reputation and contract theory,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2906–2920, Mar. 2019.

Karas, D. S., A. A. Boulogeorgos, and G. K. Karagiannidis [2016]. “Physical layer security with uncertainty on the location of the eavesdropper,” *IEEE Wireless Commun. Lett.*, vol. 5, no. 5, pp. 540–543.

Kotary, J., F. Fioretto, P. V. Hentenryck, and B. Wilder [2021]. “End-to-end constrained optimization learning: A survey,” *arXiv:2103.16378.*

Lee, H., J. Park, S. H. Lee, and I. Lee [2023]. “Message-passing based user association and bandwidth allocation in HetNets with wireless backhaul,” *IEEE Trans. Wireless Commun.*, vol. 22, no. 1, pp. 704–717.

Lei, K., M. Du, J. Huang, and T. Jin [2020]. “Groupchain: Towards a scalable public blockchain in fog computing of IoT services computing,” *IEEE Trans. Serv. Comput.*, vol. 13, no. 2, pp. 252–262.

Letaief, K. B., Y. Shi, J. Lu, and J. Lu [2022]. “Edge artificial intelligence for 6G: Vision, enabling technologies, and applications,” *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 5–36.

Li, T., C. Xue, Y. Li, and O. A. Dobre [2020]. “Insecure region around receiver for downlink transmissions with randomly located active eavesdropper,” *IEEE Wireless Commun. Lett.*, vol. 9, no. 9, pp. 1552–1556.

Li, W., A. Sforzin, S. Fedorov, and G. O. Karame [2017]. “Towards scalable and private industrial blockchains,” in *Proc. Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts (ACM Workshop)*, United Arab Emirates, pp. 9–14.

Li, M., H. Tang, and X. Wang [2019a]. "Mitigating routing misbehavior using blockchain-based distributed reputation management system for IoT networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Shanghai, China, pp. 1–6.

Li, Y., Y. Zhang, Y. Liu, Q. Meng, and F. Tian [2019b]. "Fog node selection for low latency communication and anomaly detection in fog networks," in *Proc. Int. Conf. Commun. Inf. Syst. Comput. Eng. (CISCE)*, Haikou, China, pp. 276–279.

Li, M., F. R. Yu, P. Si, W. Wu, and Y. Zhang [2020]. "Resource optimization for delay-tolerant data in blockchain-enabled IoT with edge computing: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9399–9412.

Li, S., C. She, Y. Li and B. Vucetic [2021a]. "Constrained deep reinforcement learning for low-latency wireless VR video streaming," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Madrid, Spain, pp. 01–06.

Li, T., Z. Hong, L. Liu, Z. Wen, and L. Yu [2021b]. "Meta-WF: Meta-learning-based few-shot wireless impersonation detection for Wi-Fi networks," *IEEE Commun. Lett.*, vol. 25, no. 11, pp. 3585–3589.

Li, C., C. She, N. Yang, and T. Q. S. Quek [2022]. "Secure transmission rate of short packets with queueing delay requirement," *IEEE Trans. Wireless Commun.*, vol. 21, no. 1, pp. 203–218.

Liang, Q., F. Que, and E. Modiano [2018]. "Accelerated primal-dual policy optimization for safe reinforcement learning," *arXiv:1802.06480.*

Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra [2016]. "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICLR)*, New York, USA.

Lin, Y., H. Du, D. Niyato, J. Nie, J. Zhang, Y. Cheng, and Z. Yang [2023]. "Blockchain-aided secure semantic communication for AI-generated content in Metaverse," *IEEE Open J. Comput. Soc.*, vol. 4, pp. 72–83.

Ling, X., Y. Le, J. Wang, Z. Ding, and X. Gao [2021]. "Practical modeling and analysis of blockchain radio access network," *IEEE Trans. Commun.*, vol. 69, no. 2, pp. 1021–1037.

Liu, C., J. Lee, and T. Q.S. Quek [2019a]. "Safeguarding UAV communications against full-duplex active eavesdropper," *IEEE Trans. Wireless Commun.*, vol. 18, no. 6, pp. 2919–2931.

Liu, Y., K. Wang, and Y. Lin [2019b]. "LightChain: A lightweight blockchain system for industrial internet of things," *IEEE Trans. Ind. Inf.*, vol. 15, no. 6, pp. 3571–3581.

Liu, Y., F. R. Yu, X. Li, H. Ji, and V. C. M. Leung [2020]. “Blockchain and machine learning for communications and networking systems,” *IEEE Commun. Surv. Tutorials*, vol. 22, no. 2, pp. 1392–1431.

Liu, H., P. Zhang, G. Pu, T. Yang, S. Maharjan, and Y. Zhang [2020]. “Blockchain empowered cooperative authentication with data traceability in vehicular edge computing,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4221–4232.

Liu, Y., C. She, Y. Zhong, W. Hardjawana, F.-C. Zheng, and B. Vucetic [2022]. “Interference-limited ultra-reliable and low-latency communications: Graph neural networks or stochastic geometry?,” *arXiv:2207.06918.* [Online]. Available: https://arxiv.org/abs/2207.06918

Liu, Y., J. Wang, Z. Yan, Z. Wan, and R. Jäntti [2023]. “A survey on blockchain-based trust management for Internet of Things,” *IEEE Internet Things J.*, vol. 10, no. 7, pp. 5898–5922.

Lu, Y., P. Cheng, Z. Chen, W. H. Mow, Y. Li, and B. Vucetic [2021]. “Deep multi-task learning for cooperative NOMA: System design and principles,” *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 61–78.

Malik, S., V. Dedeoglu, S. S. Kanhere, and R. Jurdak [2019]. “TrustChain: Trust management in blockchain and IoT supported supply chains,” in *Proc. International Conference on Blockchain (Blockchain)*, Atlanta, GA, USA, 2019, pp. 184–193.

Nassar, E. M. A., A. M. Iliyasu, P. M. El-Kafrawy, O. Song, A. K. Bashir, and A. A. A. El-Latif [2020]. “DITrust chain: Towards blockchain-based trust models for sustainable healthcare IoT systems,” *IEEE Access*, vol. 8, pp. 111223–111238.

Malik, S., V. Dedeoglu, S. S. Kanhere, and R. Jurdak [2019]. “TrustChain: Trust management in blockchain and IoT supported supply chains,” in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Atlanta, GA, USA, 2019, pp. 184–193.

Z. Meng, C. She, G. Zhao and D. De Martini [2023]. “Sampling, communication, and prediction co-design for synchronizing the real-world device and digital model in Metaverse,” *IEEE J. Sel. Areas Commun.*, vol. 41, no. 1, pp. 288–300, Jan. 2023.

Mohammady, R. D., M. Y. Naderi, and K. R. Chowdhury [2014]. “Spectrum allocation and QoS provisioning framework for cognitive radio with heterogeneous service classes,” *IEEE Trans. Wireless Commun.*, vol. 13, no. 7, pp. 3938–3950.

Nakamoto, S. [2008]. *Bitcoin: A peer-to-peer electronic cash system.* [Online]. Available: http://bitcoin.org/bitcoin.pdf.

Nasrallah, A., A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury [2019]. “Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research,” *IEEE Commun. Surv. Tutorials*, vol. 21, no. 1, pp. 88–145.

Nichol, A., J. Achiam, and J. Schulman [2018]. “On first-order meta-learning algorithms,” *arXiv:1803.02999*.

Pan, J. and J. McElhannon [2018]. “Future edge cloud and edge computing for Internet of Things applications,” *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449.

Parvez, I., A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai [2018]. “A survey on low latency towards 5G: RAN, core network and caching solutions,” *IEEE Commun. Surv. Tutorials*, vol. 20, no. 4, pp. 3098–3130.

Pokhrel, S. R. and J. Choi [2020]. “Federated learning with blockchain for autonomous vehicles: Analysis and design challenges,” *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4734–4746.

Polyanskiy, Y., H. V. Poor, and S. Verdu [2010]. “Channel coding rate in the finite blocklength regime,” *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2307–2359.

Rafique, W., L. Qi, I. Yaqoob, M. Imran, R. U. Rasool, and W. Dou [2020]. “Complementing IoT services through software defined networking and edge computing: A comprehensive survey,” *IEEE Commun. Surv. Tutorials*, vol. 22, no. 3, pp. 1761–1804.

Raghu, A., M. Raghu, S. Bengio, and O. Vinyals [2020]. “Rapid learning or feature reuse? Towards understanding the effectiveness of MAML,” in *Proc. Int. Conf. Learn. Representations (ICLR)*.

Raya, M., P. Papadimitratos, V. D. Gligor, and J. -. Hubaux [2008]. “On data-centric trust establishment in ephemeral Ad Hoc networks,” in *Proc. IEEE 27th Conf. Comput. Commun. (INFOCOM’08)*, Phoenix, AZ, USA, pp. 1238–1246.

Ranaweera, P., A. D. Jurcut, and M. Liyanage [2021]. “Survey on multi-access edge computing security and privacy,” *IEEE Commun. Surv. Tutorials*, vol. 23, no. 2, pp. 1078–1124.

Rodrigues, T. K., K. Suto, H. Nishiyama, J. Liu, and N. Kato [2020]. “Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective,” *IEEE Commun. Surv. Tutorials*, vol. 22, no. 1, pp. 38–67.

Sakr, A. H. and E. Hossain [2015]. “Cognitive and energy harvesting-based D2D communication in cellular networks: Stochastic geometry modeling and analysis,” *IEEE Trans. Commun.*, vol. 63, no. 5, pp. 1867–1880.

Shafique, K., B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim [2020]. "Internet of Things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios," *IEEE Access*, vol. 8, pp. 23022–23040.

Shakarami, A., M. Ghobaei-Arani, and A. Shahidinejad [2020]. "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," *Comput. Netw.*, vol. 182.

Sharma, H., I. Budhiraja, N. Kumar, and R. K. Tekchandani [2022]. "Secrecy rate maximization for THz-enabled femto edge users using deep reinforcement learning in 6G," in *IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, New York, USA.

She, C., C. Sun, Z. Gu, Y. Li, C. Yang, H. V. Poor, and B. Vuceti [2021]. "A tutorial on ultrareliable and low-latency communications in 6G: Integrating domain knowledge into deep learning," *Proc. IEEE*, vol. 109, no. 3, pp. 204–246.

She, C., Y. Duan, G. Zhao, T. Q. S. Quek, Y. Li, and B. Vucetic [2019]. "Cross-layer design for mission-critical IoT in mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9360–9374.

Shen, Y., Y. Shi, J. Zhang, and K. B. Letaief [2021]. "Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 101–115.

Shi, J., H. Yang, C. Pan, X. Chen, Q. Sun, Z. Yang, and W Xu [2023]. "Low-latency design for satellite assisted wireless VR networks," *IEEE Commun. Lett.*, vol. 27, no. 6, pp. 1555–1559.

Shojaeifard, A., K. A. Hamdi, E. Alsusa, D. K. C. So, and J. Tang [2015]. "Exact SINR statistics in the presence of heterogeneous interferers," *IEEE Trans. Inf. Theory*, vol. 61, no. 12, pp. 6759–6773.

Shuai, K., Y. Miao, K. Hwang, and Z. Li [2023]. "Transfer reinforcement learning for adaptive task offloading over distributed edge clouds," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 2175–2187.

Spinelli, F. and V. Mancuso [2021]. "Toward enabled industrial verticals in 5G: A survey on MEC-based approaches to provisioning and flexibility," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 1, pp. 596–630.

Sun, C., C, She, and C. Yang [2023]. "Unsupervised deep learning for optimizing wireless systems with instantaneous and statistic constraints" in *Ultra-reliable and low-latency communications (URLLC) theory and practice: Advances in 5G and beyond*, 1st ed. Hoboken, NJ, USA: John Wiley&Sons, Ltd., ch. 4, pp. 85–118.

Sun, Y., L. Zhang, G. Feng, B. Yang, B. Cao, and M. A. Imran [2019a]. "Blockchain-enabled wireless internet of things: Performance analysis and optimal communication node deployment," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5791–5802.

Sun, C. and C. Yang [2019b]. "Unsupervised deep learning for ultra-reliable and low-latency communications," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Waikoloa, HI, USA, pp. 1–6.

Sun, C., C. She, C. Yang, T. Q. S. Quek, Y. Li, and B. Vucetic [2019c]. "Optimizing resource allocation in the short blocklength regime for ultra-reliable and low-latency communications," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 402–415.

Sun, G., Z. T. Gebrekidan, G. O. Boateng, D. A.-Mensah, and W. Jiang [2019d]. "Dynamic reservation and deep reinforcement learning based autonomous resource slicing for virtualized radio access networks," *IEEE Access*, vol. 7, pp. 45758–45772.

Sun, C., J. Wang, X. Gao, Z. Ding, and X. Zheng [2021]. "Fiber-enabled optical wireless communications with full beam coverage," *IEEE Trans. Commun.*, vol. 69, no. 5, pp. 3207–3221.

Sutton, R. S. and A. G. Barto [2016]. *Reinforcement learning: An introduction.* Cambridge, MA, USA: MIT Press.

Tang J. and X. Zhang [2007]. "Quality-of-service driven power and rate adaptation over wireless links," *IEEE Trans. Wireless Commun.*, vol. 6, no. 8, pp. 3058–3068.

Tang, F., B. Mao, N. Kato, and G. Gui [2021]. "Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 3, pp. 2027–2057, Q3 2021.

Taylor, C. R. [1994]. "Dynamic programming and the curses of dimensionality" in *Applications of dynamic programming to agricultural decision problems*, 1st ed. Boca Raton, USA: CRC Press.

Tschorsch, F. and B. Scheuermann [2016]. "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 3, pp. 2084–2123.

Vaezi, M., A. Azari, S. R. Khosravirad, M. Shirvanimoghaddam, M. M. Azari, D. Chasaki, and P. Popovski [2022]. "Cellular, wide-area, and non-terrestrial IoT: A survey on 5G advances and the road toward 6G," *IEEE Commun. Surv. Tutorials*, vol. 24, no. 2, pp. 1117–1174.

Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio [2018]. "Graph attention networks," *arXiv:1710.10903.*

Wang, H. -M., Q. Yang, Z. Ding, and H. V. Poor [2019]. "Secure short-packet communications for mission-critical IoT applications," *IEEE Trans. Wireless Commun.*, vol. 18, no. 5, pp. 2565–2578.

Wang, C., D. Deng, L. Xu, and W. Wang [2022a]. “Resource scheduling based on deep reinforcement learning in UAV assisted emergency communication networks,” *IEEE Trans. Commun.*, vol. 70, no. 6, pp. 3834–3848.

Wang, Y., M. Chen, Z. Yang, W. Saad, T. Luo, S. Cui, H. V. Poor [2022b]. “Meta-reinforcement learning for reliable communication in THz/VLC wireless VR networks,” *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 7778–7793.

Wang, J., Y. Wang, P. Cheng, K. Yu, and W. Xiang [2023]. “DDPG-based joint resource management for latency minimization in NOMA-MEC networks,” *IEEE Commun. Lett.*, vol. 27, no. 7, pp. 1814–1818.

Wang, S., D. Li, Y. Zhang, and J. Chen [2019]. “Smart contract-based product traceability system in the supply chain scenario,” *IEEE Access*, vol. 7, pp. 115122–115133.

Wijethilaka, S. and M. Liyanage [2021]. “Survey on network slicing for Internet of Things realization in 5G networks,” *IEEE Commun. Surv. Tutorials*, vol. 23, no. 2, pp. 957–994.

Wu, B. and Q. Wang [1997]. “Maximization of the channel utilization in wireless heterogeneous multiaccess networks,” *IEEE Trans. Veh. Technol.*, vol. 46, no. 2, pp. 437–444.

Wu, D. and R. Negi [2003]. “Effective capacity: a wireless link model for support of quality of service,” *IEEE Trans. Wireless Commun.*, vol. 2, no. 4, pp. 630-–643.

Wu, H., K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu [2021a]. “EEDTO: An energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing,” *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2163–2176.

Wu, Z., S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu [2021b]. “A comprehensive survey on graph neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24.

Wu, J., W. Chen, and A. Ephremides [2023]. “Achieving extremely low latency: Incremental coding for real-time applications,” *IEEE Trans. Commun.*, vol. 71, no. 8, pp. 4453–4467.

Xiao, L., Y. Ding, D. Jiang, J. Huang, D. Wang, J. Li, and H. V. Poor [2020]. “A reinforcement learning and blockchain-based trust mechanism for edge networks,” *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5460–5470.

Xiao, Y., Y. Song, and J. Liu [2023]. “Multi-agent deep reinforcement learning based resource allocation for ultra-reliable low-latency Internet of controllable things,” *IEEE Trans. Wireless Commun.*, vol. 22, no. 8, pp. 5414–5430.

Xiong, C., G. Y. Li, Y. Liu, Y. Chen, and S. Xu [2013]. "Energy-efficient design for downlink OFDMA with delay-sensitive traffic," *IEEE Trans. Wireless Commun.*, vol. 12, no. 6, pp. 3085–3095.

Xiong, X., K. Zheng, L. Lei, and L. Hou [2020]. "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1133–1146.

Xiong, Z., J. Kang, D. Niyato, P. Wang, and H. V. Poor [2020]. "Cloud/Edge computing service management in blockchain networks: Multi-leader multi-follower game-based ADMM for pricing," *IEEE Trans. Serv. Comput.*, vol. 13, no. 2, pp. 356–367.

Xu, L. D., Y. Lu, and L. Li [2021a]. "Embedding blockchain technology into IoT for security: A survey," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10452–10473.

Xu, Y., Z. Zhao, P. Cheng, Z. Chen, M. Ding, B. Vucetic, and Y. Li [2021b]. "Constrained reinforcement learning for resource allocation in network slicing," *IEEE Commun. Lett.*, vol. 25, no. 5, pp. 1554–1558.

Xu, W., Z. Yang, D. W. K. Ng, M. Levorato, Y. C. Eldar, and M. Debbah [2023a]. "Edge learning for B5G networks with distributed signal processing: Semantic communication, edge computing, and wireless sensing," *IEEE J. Sel. Top. Signal Process.*, vol. 17, no. 1, pp. 9–39.

Xu, Q., Z. Su, D. Fang, and Y. Wu [2023b]. "Hierarchical bandwidth allocation for social community-oriented multicast in space-air-ground integrated networks," *IEEE Trans. Wireless Commun.*, vol. 22, no. 3, pp. 1915–1930.

Yang H., Z. Xiong, J. Zhao, D. Niyato, L. Xiao, and Q. Wu [2021]. "Deep reinforcement learning based intelligent reflecting surface for secure wireless communications," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 375–388.

Yang, Z., K. Yang, L. Lei, K. Zheng, and V. C. M. Leung [2019]. "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1495–1505.

Yao, Y., X. Chang, J. Mišić, V. B. Mišić, and L. Li [2019]. "BLA: Blockchain-assisted lightweight anonymous authentication for distributed vehicular fog services," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3775–3784.

Ye, N., X. Li, H. Yu, L. Zhao, W. Liu, and X. Hou [2020]. "DeepNOMA: A unified framework for NOMA using deep multi-task learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2208–2225.

Yu, W., A. Chorti, L. Musavian, H. Vincent Poor, and Q. Ni [2019]. "Effective secrecy rate for a downlink NOMA network," *IEEE Trans. Wireless Commun.*, vol. 18, no. 12, pp. 5673–5690.

Yu, Y., S. Liu, L. Guo, P. L. Yeoh, B. Vucetic, and Y. Li [2020]. "CrowdR-FBC: A distributed fog-blockchains for mobile crowdsourcing reputation management," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8722–8735.

Yu, Y., S. Liu, P. Yeoh, B. Vucetic, and Y. Li [2021]. "LayerChain: A hierarchical edge-cloud blockchain for large-scale low-delay IIoT applications," *IEEE Trans. Ind. Inf.*, vol. 17, no. 7, pp. 5077–5086.

Yuan, Y., G. Zheng, K. -K. Wong, B. Ottersten, and Z. -Q. Luo [2021]. "Transfer learning and meta learning-based fast downlink beamforming adaptation," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1742–1755.

Zanzi, L., A. Albanese, V. Sciancalepore, and X. Costa-Pérez [2020]. "NSBchain: A secure blockchain framework for network slicing brokerage," in *Proc. 2020 IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, pp. 1–7.

Zanzi, L., V. Sciancalepore, A. Garcia-Saavedra, H. D. Schotten, and X. Costa-Pérez [2021]. "LACO: A latency-driven network slicing orchestration in beyond-5G networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 667–682.

Zappone, A., M. Di Renzom, and M. Debbah [2019]. "Wireless networks design in the era of deep learning: Model-based, AI-based, or both?," *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 7331–7376.

Zhang, H., R. Wang, W. Sun, and H. Zhao [2021a]. "Mobility management for blockchain-based ultra-dense edge computing: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7346–7359.

Zhang, M., K. Cumanan, J. Thiyagalingam, Y. Tang, W. Wang, Z. Ding, and O. A. Dobre [2021b]. "Exploiting deep learning for secure transmission in an underlay cognitive radio network," *IEEE Trans. Veh. Technol.*, vol. 70, no. 1, pp. 726–741.

Zhang, J., Y. Yuan, G. Zheng, I. Krikidis, and K. -K. Wong [2022]. "Embedding model-based fast meta learning for downlink beamforming adaptation," *IEEE Trans. Wireless Commun.*, vol. 21, no. 1, pp. 149–162.

Zhou, Y., F. R. Yu, J. Chen, and B. He [2022]. "Joint resource allocation for ultra-reliable and low-latency radio access networks with edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 1, pp. 444–460.

Zhou, Y., P. L. Yeoh, H. Chen, Y. Li, R. Schober, L. Zhuo, and B. Vucetic [2018]. "Improving physical layer security via a UAV friendly jammer for unknown eavesdropper location," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11280–11284.

Zuo, Y., S. Jin, S. Zhang, Y. Han, and K. -K. Wong [2021]. “Delay-limited computation offloading for MEC-assisted mobile blockchain networks,” *IEEE Trans. Commun.*, vol. 69, no. 12, pp. 8569–8584.