Investigation of Deep Learning and Blockchain Applicability for Software-Defined Internet of Things

Bassem Sellami

ED - FST
MISTM
École Doctorale de Mathématiques, Informatique, Science et Technologie de la Matière

كلية العلوم بتونس
FACULTÉ DES SCIENCES DE TUNIS

UTM
جامعة تونس المنار
UNIVERSITÉ DE TUNIS EL MANAR

THESIS

To obtain the degree of
DOCTOR OF PHILOSOPHY
in Computer Science

Prepared in the Laboratory
LIPAH - LR11ES14

Defended by
Bassem SELLAMI

Investigation of Deep Learning and Blockchain Applicability for Software-Defined Internet of Things

Publicly defended on October 24th, 2022

Committee

Dr. Walid BARHOUMI	Professor at the University of Carthage, Tunisia	President
Dr. Aniruddha GOKHALE	Professor at Vanderbilt University, USA	Reviewer
Dr. Abderrazak JEMAI	Professor at University of Carthage, Tunisia	Reviewer
Dr. Hella KAFFEL BEN AYED	Associate Professor at the University of Tunis EL Manar, Tunisia	Examiner
Dr. Pascal BERTHOU	Associate Professor at the University of Paul Sabatier - Toulouse III, France	Visitor
Dr. Akram HAKIRI	Associate Professor at University of Carthage, Tunisia	Supervisor
Dr. Sadok BEN YAHIA	Professor at University of Tunis El Manar, Tunisia	Supervisor

Academic Year: 2021/2022

Dedication

I Dedicate my dissertation work to my dear father *Habib* and dear mother *Zoubeida* who have always believed in me.

All the words in the world cannot express the immense love I have for you, nor the deep gratitude that I show you for all the efforts and sacrifices you have never ceased to make for my education and well-being.

A big THANK YOU for your support.

To my dear brother *Akrem* and his wife, my dear sister *Asma* and her husband, and dear younger brother *Oussama (Ala)* and his fiancee. My dear all, THANK YOU so much for your confidence and encouragement.

To my dear grandfather *Mustapha* for her encouragement. To the souls of dear grandfather *Abbes* and dear all grandmothers *Habiba* and *Mariem*.

To all my extended families, *Sellami* and *Inoubli*.

To all my dear friends.

To all who help me from far or near.

2022, Bassem.

Acknowledgements

First, I would like to thank Allah, my God, the All-Merciful the Most-Merciful, for guiding me to complete this work.

The work presented in this thesis has been carried out at the Laboratory in Computer Science in Algorithmic and Heuristic Programming (LIPAH), directed by Mr. Sadok Ben Yahia and Mme Hella Kaffel Ben Ayed, successively, to whom I wish to express my thanks for their hospitality.

I would also like to think here are the people who, directly or indirectly, have contributed to the realization of this doctoral thesis work. These acknowledgments are written in a moment of gentle intellectual relaxation, without any absolute rigor or taxonomic concern. I left to chance of my memory, more impressed by recent events, repeated, or emotionally charged, the task of finding his people. Thanks would undoubtedly have been quite different, and in another state of mind, I might have forgotten one of the following names. But I chose this precise moment to write them.

My thanks go to the people who offered me the thesis subject and who supervised me throughout these years of study. *Professor Sadok Ben Yahia* and *Dr. Akram Hakiri*. Through our discussions, they gave me a deeper understanding of various aspects of the subject. I also salute the flexibility and open-mindedness of my thesis supervisors, who were able to leave me a large margin of freedom to carry out this research work. I would like to thank them for their confidence in me, follow-up work, and enriching remarks while underlining their kindness and benevolence.

I am also very grateful for the time granted by all the members of the jury of this Ph.D. thesis and for their constructive remarks, as well as for accepting to review this thesis manuscript and criticisms, which they lavished on me and helped improve the present manuscript. My thanks go as well to the rest of my thesis committee for accepting to evaluate my work:

- Dr. **Aniruddha GOKHALE**, Professor at Vanderbilt University, USA.
- Dr. **Abderrazak JEMAI**, Professor at the University of Carthage, Tunisia.
- Dr. **Pascal Berthou**, Associate Professor-HDR at the University of Paul Sabatier-Toulouse III, France.

- Dr. **Hella Kaffel Ben Ayed**, Associate Professor-HDR at the University of Tunis EL Manar, Tunisia.
- Dr. **Walid Barhoumi**, Professor at the University of Carthage, Tunisia.
- Dr. **Sadok Ben Yahia**, Professor at Tallinn University of Technology, Estonia
- Dr. **Akram Hakiri**, Associate Professor at the University of Carthage, Tunisia.

I would like to thank my brother *Akrem Sellami*, Lecturer at Telecom and Research Scientist in the Department D4: Natural Language Processing and Knowledge Discovery at LORIA Labs, both in Nancy, France, for his advice and encouragement during my thesis.

My thanks also go to all the members of the LIPAH Labs, permanent staff, doctoral students and interns for their welcome and good humor. In particular, I would like to thank my colleagues with whom I shared wonderful moments: Bchira Ben Mabrouk, Maher, Radhi, Mohamed, Jamel, Aymen, Wissem, and all my other colleagues at LIPAH and at ISSAT Mateur.

My thanks also go to the secretariat of the LIPAH Team, for their availability, their kindness and their precious help in the preparation of administrative tasks.

I would also like to thank the doctoral students and other members of the various technical and administrative departments of the Faculty of Sciences of Tunis, who allowed me to work in excellent conditions.

Additionally, my acknowledgments go to the Tunisian Ministry of Higher Education, the University of Tunis El Manar, Faculty of Sciences of Tunis and the LIPAH laboratory for the financial support.

To top it off, I close these acknowledgments by dedicating this doctoral thesis to my parents and my friends whom I was lucky enough to have by my side, who supported me and believed in me throughout these years of work.

Abstract

5G mobile network has seen phenomenal growth in providing IoT services and applications. IoT devices are often battery-powered to perform their operations autonomously and serve a variety of situations, such as smart cities, autonomous cars, smart manufacturing, etc., thereby needing efficient energy consumption to extend their lifespan. IoT networks should provide i) an on-demand resource allocation to support adaptive horizontal and vertical scaling of the network resources; ii) flexible infrastructure virtualization that exploits in-network programmability capabilities to operate inside an SDN-enabled virtualization platform; iii) a device-driven and human-driven intelligence to address the issues of energy efficiency and ultra-low latency requirements for future reliable and real-time IoT applications. Despite the promise, IoT networks face several challenging issues stemming from resource constraints and low-computation performance. Additionally, IoT systems encounter several security and privacy concerns to prevent unauthorized access to smart devices and secure trust-less interactions between devices themselves and service providers on the Internet.

To address this plethora of challenges, this thesis presents an energy-efficiency IoT system, less computation-intensive, easy to implement, and amenable to online adaptation to the variations of the network condition. In the first contribution, we introduce a novel IoT network virtualization approaches based on SDN/NFV to offer a high degree of automation in service chaining delivery for IoT devices. The second contribution introduces a Deep Reinforcement Learning energy-efficient task assignment and scheduling in SDN-based fog IoT Network. Furthermore, we present a computing model for reducing network latency and traffic overhead by centralizing the network control and orchestration in a single SDN controller layer. The last contribution introduces a deep learning approach that combines SDN and blockchain to achieve task scheduling and offloading, improve the response rate of IoT services to offer high levels of performance, and strive to perform dynamic resource management.

Résumé

Le réseau mobile 5G a connu une croissance phénoménale dans la fourniture de services et d'applications IoT. Les appareils IoT sont souvent alimentés par batterie pour effectuer leurs opérations de manière autonome et servir à diverses situations, telles que les villes intelligentes, les voitures autonomes, la fabrication intelligente, etc. ont donc besoin d'une consommation d'énergie efficace pour prolonger leur durée de vie. Les réseaux IoT devraient fournir: i) une allocation de ressources à la demande pour prendre en charge une mise à l'échelle horizontale et verticale adaptative des ressources du réseau; ii) une virtualisation d'infrastructure flexible qui exploite les capacités de programmabilité en réseau pour fonctionner à l'intérieur d'une plate-forme de virtualisation compatible SDN; iii) une intelligence pilotée par les appareils et pilotée par l'homme pour répondre aux problèmes d'efficacité énergétique et aux exigences de latence ultra-faible pour les futures applications IoT fiables et en temps réel. Malgré la promesse, le réseau IoT est confronté à plusieurs problèmes complexes liés à ses contraintes de ressources et à ses faibles performances de calcul. De plus, les systèmes IoT rencontrent plusieurs problèmes de sécurité et de confidentialité pour empêcher l'accès non autorisé aux appareils intelligents et pour sécuriser les interactions sans confiance entre les appareils eux-mêmes et avec les fournisseurs de services sur Internet.

Pour relever cette pléthore de défis, cette thèse présente un système IoT à haut rendement énergétique, moins gourmand en calculs, facile à mettre en œuvre et pouvant être adapté en ligne aux variations de l'état du réseau. Dans la première contribution, nous introduisons une nouvelle approche de virtualisation de réseau IoT basée sur SDN/NFV pour offrir un degré élevé d'automatisation dans la prestation de chaînage de services pour les appareils IoT. Dans la deuxième contribution, nous introduisons une attribution et une planification des tâches économes en énergie par Apprentissage par Renforcement dans un réseau IoT de brouillard basé sur SDN. Nous présentons un modèle informatique pour réduire la latence du réseau et la surcharge de trafic en centralisant le contrôle et l'orchestration du réseau dans une seule couche de contrôleur SDN. La dernière contribution introduit une approche d'apprentissage en profondeur qui combine SDN et blockchain pour réaliser la planification et le déchargement des tâches, améliorer le taux de réponse des services IoT pour offrir des niveaux de performance élevés et s'efforcer d'effectuer une gestion dynamique des ressources.

Contents

List of Figures

List of Tables

List of Acronyms

5G	Fifth Generation
IoT	Internet of Things
SDN	Software Defined Networking
NFV	Network Functions Virtualization
QoS	Quality of Service
DoS	Denial of Service
DDoS	Distributed Denial-of-Service
VANET	Vehicular Ad-Hoc Networks
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
DL	Deep Learning
ML	Machine Learning
AI	Artificiel Intelligence
TDSR	Time Series Data Repository
RFID	Radio Frequency Identification
MEC	Mobile Edge Computing
ESTI	European Telecommunications Standards Institute
IoV	Internet of Vehicles
VR	Virtual Reality
AR	Augmented Reality
MR	Mixed Reality
M2M	Machine to Machine
Wi-Fi	Wireless Fidelity
LTE	Long Term Evolution
V2X	Vehicle to Everything
V2V	Vehicle to Vehicle
P2D	Patient-to- Doctor
P2M	Patient-to-Machine
D2M	Doctor-to-Machine
WIoT	Wearable IoT
BLE	Bluetooth Low Energy

E2E	Long End-to-End
ICT	Information and Communication Technologies
IIoT	Industrial Internet of Things
ITS	Intelligent Transportation Networks
VM	Virtual Machine
LPWAN	Low Power Wide Area Network
CPU	Central Processing Unit
D2D	Device-to-Device
IP	Internet Protocol
TCP	Transmission Control Protocol
VLAN	Virtual Local Area Network
TLS	Transport Layer Security
IDS	Intrusion Detection System
API	Application Programming Interface
NFVI	Network Functions Virtualization Interface
MANO	Management and Orchestration
PoP	Point of Presence
RAN	Radio Access Network
NSC	Network Service Chaining
NFVO	Network Functions Virtualization Orchestrator
NLP	Natural Language Processing
RBM	Restricted Boltzmann Machine
DBN	Deep Belief Network
AE	AutoEncoder
DAE	Deep AutoEncoder
DNN	Deep Neural Network
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
SGD	Stochastic Gradient Decent
DQL	Deep Q-Learning
DDPG	Deep Deterministic Policy Gradient
A3C	Asynchronous Advantage Actor-Critic
P2P	Peer-to-Peer
IPFS	Interplanetary File System
DHT	Distributed Hash Table
PoW	Proof of Work
PoS	Proof of Stack
PoA	Proof of Authority
PoET	Proof of Elapsed Time

PBFT	Practical Byzantine Fault Tolerance
PoB	Proof of Burn
PoC	Proof of Capacity
DApps	Decentralized Applications
EVM	Ethereum Virtual Machine
EOA	Externally Owned Accounts
TSDR	Time Series Data Repository
HVAC	Heating, Ventilation and Air-Conditioning
MQTT	Message Queuing Telemetry Transport
PAN	Personal Area Network
CRUD	Create, Read, Update, Delete
CM	Cluster Member
CH	Cluster Head
ADV	Advertisement Message
OVS	OpenVSwitch
ODL	OpenDaylight
SFC	Service Function Chaining
PNF	Physical Network Function
UI	User Interface
AR	Activity Recognition
MSE	Mean Square Error
PCA	Principal Component Analysis
ICA	Independent Component Analysis
DQN	Deep Q-Network
DDQN	Double Deep Q-Network
MDP	Markov Decision Process
BC	Blockchain
BCO	Blockchain Order

Chapter 1

Introduction

1.1 Context

The widespread evolution of the Internet of Things (IoT) is being fueled by the rising number of devices connected to the Internet, which has a substantial influence on the global amount of data produced and ushers in a world of interconnected smart devices [1]. The current estimate of the number of deployed things is on the order of 50 billion connected devices and by 2025 will encompass 1000 times more connected mobile devices [2]. Furthermore, according to recent studies [3], IoT devices are expected to generate 45 percent of the internet traffic by 2026. Smart IoT devices will exchange data and gather data from their surrounding environment using sensors, store, analyze it, and actively interfere in it using actuators. Furthermore, data collected from IoT devices is processed and stored in the cloud. Cloud-hosted servers have been used to meet the vast majority of IoT applications' demands by enabling access to information anywhere, anytime, using an internet connection, thereby supporting ubiquitous access, availability, and scalability of processing and storage capacity.

Despite the promise, the explosive growth of IoT data contributed to the need for increased bandwidth to support different application areas, including smart cities, e-health, industrial, transportation, retail, safety, and environmental services [4]. As a result, cloud network becomes a bottleneck [5], and their architecture must be flexible enough to be reprogrammed following any change in IoT application needs [6]. Furthermore, IoT networks should be able to support fast connections to improve user experience and ultra-low latency to unlock the potential of IoT devices. This is, however, a challenging goal to achieve because today's network is limited in its ability to address the requirements of even current IoT deployments [4].

Fog and Edge computing have been seen as promising approaches to moving cloud servers close to IoT devices. Specifically, fog computing brings cloud computing services, such as communication, computation, network control, and storage, to edge network [7]. Thus, Fog services can be hosted at users' edge wireless devices to improve

network reliability and latency and overcome the issues stemming from geographically distributed locations in cloud computing. Additionally, fog computing relies on discoverable, generic, forward-deployed decentralized servers located in single-hop proximity of mobile devices. These servers should be used to offload expensive computation at the network edge, perform data filtering to remove unnecessary data from streams intended for dismounted users, aggregation, and processing at the network edge, resulting in decreased latency, network bandwidth preservation, and improved quality of service (QoS), and serve as collection points for data heading for enterprise repositories [8].

Indeed, the individual fog nodes should coordinate their processing with surrounding helper IoT nodes by considerably offloading their responsibilities to minimize task execution time. Fog nodes, in particular, should be able to handle burst and unpredictable IoT traffic over a variety of periods. They're also appealing to new time-sensitive IoT services and applications, such as updating maps for self-driving cars or delivery drones, energy usage measurements from an intelligent grid, emergency monitoring, intelligent manufacturing, interactive multiplayer online games, and disaster relief. However, wireless communication and computing resources (CPU, memory, storage) are typically limited and energy-intensive, making it challenging to meet the ever-increasing demand and dynamic needs of IoT applications and the heterogeneous conditions of smart objects communicating over the Internet. As a result, flexible resource management, sophisticated network control, and fast task scheduling algorithms are essential to maintain fair and consistent performance.

Moreover, millions of battery-powered IoT devices, such as smart cameras, smartphones, home entertainment systems, smart TVs, environmental monitoring sensors, and smart meters, are deployed to support a variety of scenarios, including smart cities, autonomous farming, smart buildings, and smart manufacturing [9]. These applications necessitate delivering large amounts of data over ultra-reliable, low-latency wireless communication services. However, the rising traffic created by IoT devices, together with the rise of IoT services and applications, raises worries about the increased energy consumption required to power, distribute, and install IoT solutions [10].

Furthermore, Software Defined Networking (SDN) [11] was introduced to deliver dramatic improvements in network agility and flexibility. SDN offers intelligent and centralized control of the underlying network infrastructure, which can help schedule and provision the network types of equipment. The decoupling of the control plane from the data plane helps to solve the resource management issues in diverse IoT environments. For example, SDN can offer a flexible and collaborative task offloading service orchestration [12]. Along with SDN, Network Function Virtualization (NFV) [13] can be used to chain and orchestrate virtual network functions (VNFs) and help the SDN controller to steer the traffic among virtual and physical appliances [14].

Even though SDN offers service providers the ability to schedule and automate network control to respond to changing QoS quickly, the security has been inadequately documented and deployed for as long as the benefits it provides, rendering it is vulnerable to assaults such as denial of service (DoS), distributed DoS, and volumetric IoT botnets [15] [16] [17].

Blockchain is being envisioned to enforce security and trustworthiness in diverse IoT environments, including transactive energy auctions, connected vehicles, and trusted healthcare systems. Blockchain has been coupled with SDN to create systematic collaboration at scale in order to limit the dangers posed by malicious attacks [18] [19] [20]. Blockchain, in particular, is secure by design since it is a transparent, immutable, verified, and pseudo-anonymous technology that can be used as the backbone for a variety of IoT applications [21]. Blockchain can, for instance, keeps track of global records in vehicle networks [22] to ensure that distributed Vehicular Ad-Hoc Networks are fair (VANET). Furthermore, SDN and blockchain have been utilized in tandem to redesign the trust relationship in IoT networks [23].

Nevertheless, blockchain incurs brutal latency, high compute costs and limited storage to process IoT transactions [24]. It also could be cost-ineffective, as it consumes substantially computing power [25] and higher energy to process and validate IoT transactions. Additionally, the lack of IoT-focused consensus protocols makes it challenging to coordinate distributed IoT systems to detect and destroy large-scale botnets. Thus, IoT edge devices must optimize their energy usage, use energy harvesting technologies to charge their batteries, and continue to run for as long as feasible, all during processing and scheduling their duties and providing suitable services.

This dissertation focuses on developing holistic management solutions for edge IoT environments that ensure energy efficiency while also considering the various demands of IoT applications workloads, their dynamicity, and responsiveness to the QoS requirements. Diverse, challenging issues are linked to the realization of cooperative and intelligent IoT communication and ensuring interoperability and energy efficiency. On the one hand, excessive and unnecessary use of IoT power and energy resources should be limited to drive advances in energy-efficient. On the other hand, IoT devices should be able to support context-awareness to offer ambient intelligence to learn their surroundings.' Environment and adapt behaviors accordingly. On the other hand, IoT devices should see their resources accessible to fog nodes to transfer heavy computational tasks from one resource-limited device to more resourceful edge servers. Finally, IoT networks should benefit from decentralized security infrastructure to deal with DDoS attacks and eliminate malicious risks. Specifically, edge networks should implement decentralized trust management and secure usage control schemes to enforce massive IoT transactions' security.

1.2 Contribution

To address this plethora of challenges, the objectives of this dissertation are to develop solutions that focus on the three essential issues in massively distributed IoT network.

- **Supporting a Context- and Energy-Aware Architecture for IoT.** First, we investigate the feasibility of extending SDN and NFV to support distributed IoT automation and orchestration at the network edge while implementing an IoT data management model retrieving raw sensor data into a hierarchical containment tree. We inquire into the context-aware knowledge learning model for transforming those sensors IoT raw data into meaningful context representation models to achieve a low-energy footprint.

- **Supporting Device-Driven and Human-Driven Intelligence for Cyber Foraging IoT devices.** To address the issues of energy efficiency and ultra-low latency requirements for future reliable and real-time IoT applications, we examine Deep Reinforcement Learning (DRL) techniques to achieve IoT task assignment, scheduling and offloading. We look into task allocation and resource planning approaches in dynamic and distributed IoT environments, while ensuring consumable energy optimization minimizes latency.

- **Decentralized Trust Management for Secure and Transactional IoT Data.** We delve into the integration of Blockchain and Reinforcement Learning (DRL) to achieve energy-aware task cyber foraging in SDN-enabled IoT systems. We dig into different DRL policies to improve the reliability, low latency, and energy efficiency, in tandem with the blockchain consensus algorithm to verify and validate IoT transactions.

The recent research works are mainly based on meticulously designed heuristics that ignore the patterns of incoming tasks. Therefore, we can develop an approach that uses SDN to enhance the control and management of fog-enabled IoT networks in terms of flexibility and intelligence. Also, Edge devices should trust this centralized intermediary, creating a single failure point. Concurrently, a handful of research activities have paid heed to address symbiotic security, energy efficiency, low latency, cyber foraging, and scalability.

To fill the gap by ensuring energy efficiency, reliability, security, trustworthiness, and low latency in a 5G massive IoT network, we present the contributions of this thesis as illustrated in Figure 1.1 and described as follows:

1. **Contribution 1** : Due to the significant increase of the global energy consumption as a result of the expansion of urban populations, it becomes necessary to

use a context-aware IoT approach where sensors can learn from their surrounding environment to regulate the actuators in a coordinated network and offer efficient energy consumption in smart buildings.

This contribution introduces a thorough architectural framework that enables Context-Aware IoT systems in SDN-aware networks to enable efficient energy management in smart buildings. This contribution makes use of lightweight virtualized network functions to allow distributed automation and orchestration of IoT devices, and describe a context-aware method for gathering, filtering, and processing data from sensing data in campus buildings.

2. **Contribution 2** : The IoT infrastructure at the network edge is under increasing stress as a result of the rising demand and the variety of traffic patterns originating from unpredictable, heterogeneous connected IoT objects. Edge nodes, however, frequently struggle to implement energy-awareness and resource allocation strategies in a way that effectively balances reaction time limitations, model quality, inference accuracy, and task schedulability. Edge resources, such as servers, routers, controllers, and IoT gateways show varying execution durations and an increasing energy consumption, which makes it difficult to schedule the traffic optimally. To handle the unpredictability of job management, the contribution 2 of this thesis describes a DRL strategy for IoT traffic scheduling at the network edge that is built on SDN. The contribution introduces an architectural framework to fill the gap with task allocation and scheduling issues, improve the network performance, ensure lower latency and efficient energy saving.

3. **Contribution 3** : Edge IoT services and location-sensitive distributed computing are made possible by the IoT systems. By making data access and processing quick and effective close to IoT devices, it enhances network dependability and reduce latency right away. However, because of the decentralization of localization service, the security and privacy-preserving become challenging issues. Furthermore, the mobility of IoT devices becomes more challenging so that the resource allocation and task scheduling should consider diverse parameters of wireless channel, including interference management, fading, and QoS. While blockchain has been seen as solution to enhance security and data integrity, it however comes at the cost of the energy consumption, which increases drastically. The contribution 3 of this thesis offers a symbiotic blockchain and DRL approach to improve IoT task scheduling and offloading and reduce the energy consumption in distributed connected IoT devices. The blockchain ensure privacy-preserving and coordinated activity against distributed attacks, by validating IoT transactions and blocks, using a novel consensus protocol. The DRL approach offers

distributed and parallel multi-agent Actor-Critic Agent (A3C) strategy to enable effective work offloading and scheduling, while ensuring ultra-lower energy consumption.

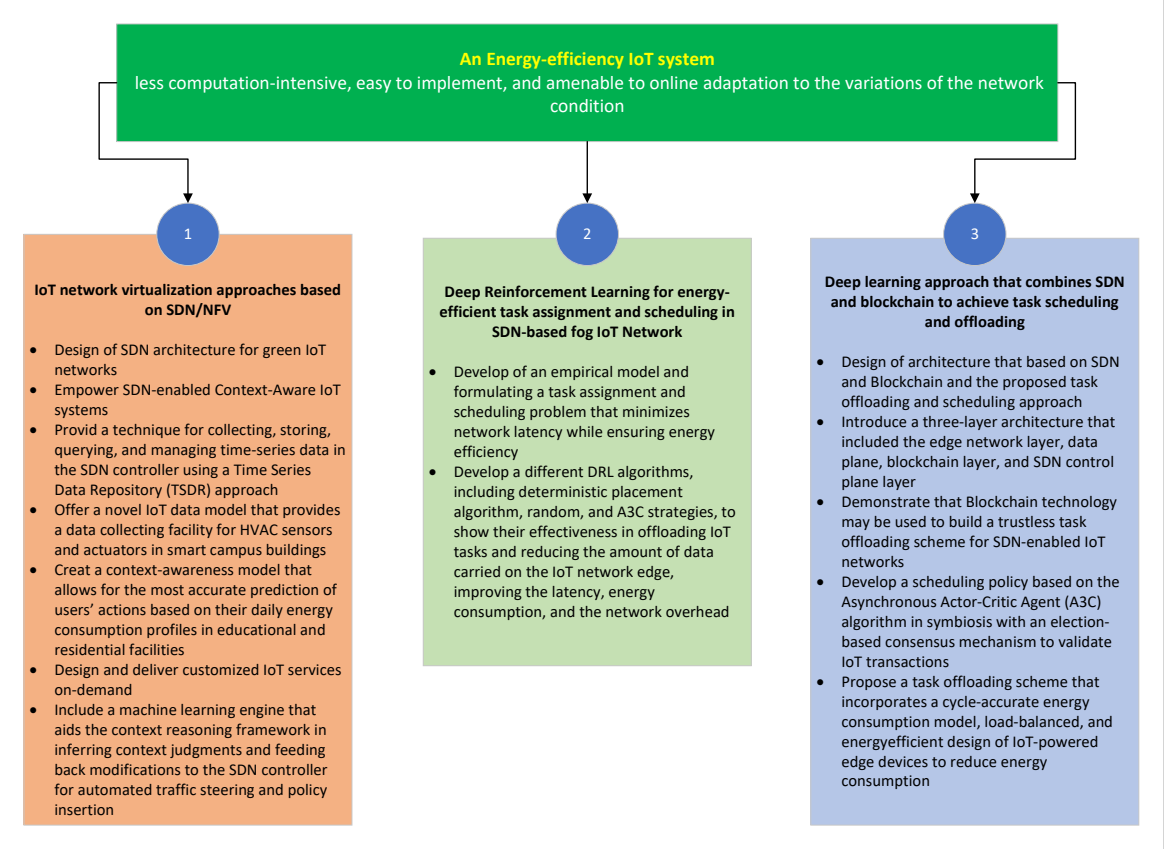

Figure 1.1: Thesis Contributions

1.3 Dissertation Organization

To achieve the objectives above, the organization of this thesis, in addition to this introduction, includes the following five chapters:

- **Chapter 2: State-of-the-Art.** This chapter describes the state-of-the-art on edge/fog IoT communication and the challenges related to massively distributed IoT systems. First, it elucidates concepts and approaches that have been discussed in the literature in SDN and NFV support for 5G mobile backbone, network softwarization, and slicing for IoT. Then, we discuss AI, and Machine Learning (ML) approaches for traffic management in different IoT systems and discuss their influences on revolutionizing technologies in several real-world areas. Finally, we emphasize task offloading and resource allocation problems using Reinforcement Learning and empowering cognitive and autonomic control and management of IoT services. Next, we draw on the research directions on the

blockchain, smart contracts, and distributed ledgers to create tamper-resistant records of shared transactions and improve trustworthiness and model validation. Finally, we highlight the convergence of blockchain and the Internet of Things (IoT) and empowering Blockchain-based IoT networks with SDN/NFV.

- **Chapter 3: A Context-Awareness Energy-Efficient Framework for SDN-enabled IoT Network.** In this chapter, we first present a novel IoT network virtualization approach based on SDN/NFV to offer a high degree of automation in service chaining delivery for IoT devices. Next, we introduce an IoT data model that provides a data collection facility to accommodate smart buildings' sensors and actuators. Then, we dig into our novel *Context-Awareness Model* that represents the functional intelligence of IoT environment. after that, we provide a novel approach to realizing the context information with Time Series Data Repository (TSDR) and model hierarchical structures and relationships across distributed edge IoT nodes.

- **Chapter4: Reinforcement Learning-based Framework for Task Offloading in IoT Network Edge.** This chapter first depicts the details of our deep learning architecture that performs dynamic IoT task scheduling and resource management. Then, it scrutinizes our DRL algorithm to address the task offloading and scheduling issue and describes the implementation details of our solution. Finally,

- **Chapter5: Enhancing Decentralization of IoT Network with Blockchain and Deep Reinforcement Learning.** The first part of this chapter devolves into the description of our proposed architecture, which integrates our secure, reliable, and flexible blockchain into the IoT network. Additionally, we present our multi-objective optimization model to achieve IoT task offloading and distribute tasks to IoT fog/edge nodes efficiently.

- **Chapter 6: Conclusion and Perspectives.** This chapter presents concluding remarks of this dissertation, alluding to lessons learned and future work.

Chapter 2

State of the Art

2.1 Introduction

In this chapter, we discuss the Internet of Things and Edge Computing background. First, in Section 2.2, we elucidate the state-of-the-art on edge/fog IoT communication. Furthermore , in Section 2.2.7 we highlight the key challenging issues in managing trusted IoT Networks. Then, in Section 2.3 we present the key concepts behind SDN and NFV to support massively distributed 5G IoT networks, network softwarization, virtualization, and slicing for IoT. Next, we highlight in Section 2.4 Machine Learning (ML) techniques for traffic management in different IoT systems. We emphasize task offloading and resource allocation problems using Reinforcement Learning and empowering cognitive and autonomic control and management of IoT services. Finally, in Section 2.5, we draw on the research directions on the blockchain, smart contracts, and distributed ledgers to create tamper-resistant records. We draw on the research directions on the convergence of blockchain and the Internet of Things (IoT) and empowering Blockchain-based IoT networks with SDN/NFV.

2.2 Internet of Things and Edge Computing

2.2.1 Internet of Things

The term "Internet of Things" (IoT) was first used in 1999 by Kevin Ashton [26] to describe the Radio Frequency Identification (RFID) network. However, the term IoT has subsequently evolved and allowed the development of diverse lightweight and inexpensive computing types of equipment. These objects are referred to by the terms *"IoT object"*, *"connected object"* or *"intelligent objects"* can intercept various changes in the surrounding environment and act accordingly. IoT extends the current digital infrastructure with everyday connected devices by making these objects intelligent, addressable, and equipped with computing, communication, and storage capabilities.

Furthermore, the IoT paradigm is considered the cornerstone for developing diverse applications and services for smart cities, industry 4.0, intelligent transportation, supply chains, smart grids, healthcare systems, and many more [27].

2.2.2 Edge Computing

Edge computing is a distributed computing framework that is distinguished by its dispersed processing capacity [28] that brings the cloud capabilities to local servers at single-hop proximity of end-users. Edge computing allows processing data at the network edge, directly by edge devices located as close to the originating source as possible. Since data are localized and processed at the periphery of the network, it is no longer required to send the data to a remote data center for processing and decision-making. Instead, edge computing can be assimilated into distributed microdata centers capable of processing or storing data locally. Additionally, edge networks are made up of edge devices, such as smartphones, connected cars, smart meters, etc., and edge servers, such as border routers, base stations, and wireless access points, all of which can be outfitted with the essential capabilities for edge computation [29]. Edge computing guarantees that localized calculations are realized at the edge, and that computational service requests are answered more quickly [30].

Edge computing has several advantages to IoT networks. Bringing computation facilities closer to the data source offers better performance. It delivers faster response times because putting processing functions closer to the end-users reduces transmission latency compared to cloud-hosted services. Furthermore, by keeping data at the network edge, the amount of traffic sent to centralized cloud servers is reduced. Thereby, bandwidth performance is improved [31]. Similarly, keeping data at the local edge servers helps to improve data management, as these servers contain valuable insights that are useful to perform real-time processing and data analysis.

2.2.3 Mobile Edge Computing

Mobile Edge Computing (MEC), also called Multi-access Edge Computing, did the European Telecommunications introduce a network architecture Standards Institute (ETSI) [32] to enable cloud computing capabilities at the edge of cellular networks, i.e., cellular base stations. MEC technology allows for software programs to access local materials and real-time information about the network conditions in the local area. Mobile core networks are relieved of additional congestion and may efficiently serve local needs by deploying various services and caching material at the network edge. MEC technologies have emerged as a critical enabler for achieving the IoT and 5G goals. MEC research is located at the crossroads of mobile computing and wireless communications, where many research opportunities have resulted in a burgeoning

field.

In recent years, academic and industrial researchers have looked into various MEC-related topics, including system and network modeling, optimum control, multiuser resource allocation, implementation, and standardization. Additionally, many real-world MEC implementations include some type of hybrid cloud access that relies on various networking topologies, such as fixed and wireless, Wi-Fi, and mobile networks. MEC is expected to enable new vertical IoT business use cases such as the Internet of Vehicles (IoV), immersive user experiences based on Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR), data caching, location services, content distribution, Machine to Machine (M2M), enhanced mobile broadband channels, and video streaming and analytics.

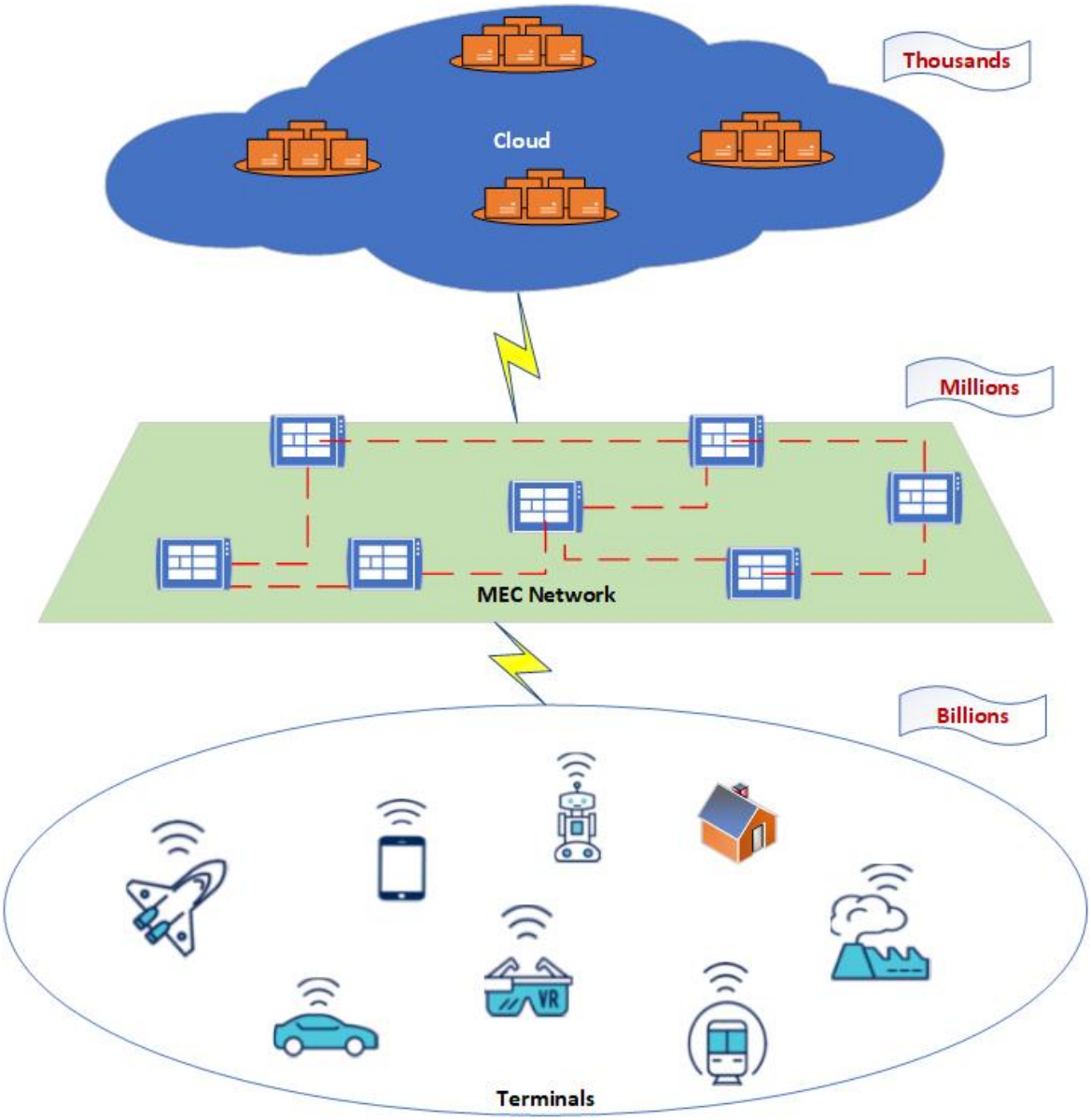

Figure 2.1: Mobile edge computing-Internet-of-Things (MEC-IoT) architecture

2.2.4 IoT and MEC Applications and Scenarios

As shown in Figure 2.2, MEC technology may be used in a variety of IoT applications, including smart cities, smart transportation, healthcare, etc. In this section, we describe different real-world use cases that make use of MEC technologies and show how each application can benefit from MEC-IoT integration.

Figure 2.2: Mobile edge computing-Internet-of-Things Applications

2.2.4.1 Smart Home and Smart City

Home automation and consumer electronics have been pioneering applications of the IoT technology [33]. Several smart home applications use IoT sensors ranging from simple thermostat sensors to more advanced automation systems, such as smart metering, smart heating and lighting, cleaning services, and home entertainment systems. The amount of data generated on a typical smart home network is predicted to grow, making it difficult to rely on cloud-hosted centralized servers to process and analyze these data.

Leveraging MEC technologies offer unprecedented opportunities for connecting future smart homes [34] by enriching home infrastructure with intelligent devices that makes life more convenient and comfortable for residents by offering reduced communication latency, easy instantiation, and fast relocation while ensuring privacy-preserving of sensitive data [35]. Furthermore, the MEC server's capabilities can be added to IoT gateways and enable extending gateway functionality to the context of smart building.

The collaboration between these gateways and MEC servers provides additional benefits such as quick instantiation, relocation, privacy preservation, and updating when needed [36].

Moreover, IoT technology has progressed from the house to its neighborhood and city-scale applications. We see a lot of potential in public safety, healthcare, utilities, tourism, and transportation. Massive amounts of IoT data generated in smart cities may be efficiently handled at the network's edge, resulting in low latency and position awareness [37]. For instance, video streaming cameras connected to a Long Term Evolution (LTE) network can send video feeds to the MEC server for real-time processing and anomaly detection [36]. In addition, applications that need the processing of geographically distant data will benefit from collaborative edge models that link numerous MEC servers.

2.2.4.2 Healthcare

5G has been envisioned as a key enabler for mobile health and telemedicine. In public healthcare facilities, wearable low-power IoT medical sensors are used to monitor health-related data and store health records. Similarly, IoT smart devices for healthcare have enabled Patient-to- Doctor (P2D), Patient-to-Machine (P2M), and Doctor-to-Machine (D2M) connectivity. Many healthcare providers adopt IoT solutions and wellness applications to help patients follow their prescribed care programs and facilitate analytics. Patients can gradually become digital stewards of their health data by using healthcare applications to monitor their conditions at home and communicate the state of their health with their families and doctors.

Despite these capabilities, sharing such confidential patient data has placed an increasing strain on the security of the MEC network. Smart healthcare systems can encounter severe security and privacy concerns, e.g., if unauthorized access is granted to wearable gadgets or trust-less transactions are performed between smart healthcare devices and their healthcare service providers over the network. Furthermore, Humanoid robots sitting next to an older adult, for example, require tactile feedback with a *1-millisecond* delay for their caretaking services. Remote surgery, for example, needs ultra-low latency, uninterrupted communication connectivity, and collaboration among doctors in various places [38].

2.2.4.3 Autonomous Vehicles/IoT Automotive

5G is a fundamental enabler of the V2X (Vehicle to Everything) idea, which includes Vehicle to Vehicle (V2V), vehicle to infrastructure, vehicle to device, vehicle to pedestrian, vehicle to home, and vehicle to grid [39]. V2X demands important communication infrastructure in IoT automotive, where dependability and ultra-low latency are critical elements. Autonomous and semi-autonomous driving, vehicle maintenance,

and in-car infotainment are examples of use cases in these areas [40]. Several elements must be addressed to operate an effective and dependable vehicular network, including real-time traffic monitoring, continuous sensing in cars, support for Infotainment apps, and increased security.

However, existing mobile networks are unable to provide these functionalities. In this line, future 5G mobile systems are likely to offer greater flexibility by exploiting network softwarization technology. In this context, V2X paired with MEC offers a practical and cost-effective alternative for accelerating V2X and IoT automotive system development [41].

2.2.4.4 Wearable IoT (WIoT)

Wearable technology has advanced dramatically in recent years, from Walkman to step trackers, smartwatches to smart eyewear. Wearable devices are being fueled by the development of low-power wireless technologies such as BLE (Bluetooth Low Energy). Wearable computing devices range from low-cost gadgets like health and fitness trackers to high-end products like virtual reality helmets and smartwatches. Wearable devices will require more advanced, reliable, and high-performance communication infrastructure as new application areas and enabling services to emerge. Virtual reality and augmented reality wearables, for example, require gigabit/s network bandwidth to execute their applications. Wearable devices' implementation in smart cities, on the other hand, will increase network traffic on communication networks. As a result, next-generation communication networks should be able to deliver gigabit speeds to the expected extremely dense wearable devices [42].

Even though cloud computing has enabled a wide range of new networking services, it will not be enough to meet the future requirements for the wearable ecosystem. Long End-to-End (E2E) latency is why centralized cloud data centers fail. Wearable applications, e.g., virtual reality perceptual stability, are delay-sensitive and require incredibly low latency. By combining cloud and MEC infrastructures, MEC offers the ability to alleviate the constraints in present cloud-based solutions. Providers will be able to deploy storage, computation, and caching capabilities close to wearable devices as a result of this [42].

2.2.4.5 Smart Energy

The smart grid system is an ICT-enabled energy generating, transmission, and distribution network. It can real-time perceive, analyze, and monitor energy flow and energy-transportation infrastructure. Adding digital controls and allowing network monitoring and telecommunication capabilities enable such functions. As a result, an intelligent grid not only allows for two-way electrical power flows and real-time, automated, bidirectional information flow. Incorporating such intelligence into the

old energy infrastructure will result in a more efficient energy system. The IoT is regarded as the cornerstone for achieving intelligence capabilities in smart grid systems. Transformers, breakers, switches, meters, relays, intelligent electronic devices, capacitor banks, voltage regulators, cameras, and other grid components are connected to the Internet through IoT. The data collected by these IoT devices are then used to allow automation. Reduced capital investment, optimal renewable capacity, decreased maintenance costs, and improved customer interaction are advantages of IoT-enabled smart grids. On the one hand, transforming an electrical grid into a smart system necessitates the inclusion of built-in, secure, networked intelligence in practically every item and piece of equipment [43]. On the other hand, an effective system is necessary to handle the generated data, i.e., to transport, store, and analyze the massive volumes of data acquired by these smart devices. As a result, cloud computing is an attractive option for these IoT-based smart grids.

Smart grids, on the whole, cover enormous geographical regions. Due to inadequate network connectivity and the large number of devices generating data, they frequently face bandwidth bottlenecks and communication delays. As a result, because it relies heavily on centralized processing, the traditional centralized cloud architecture is unsuitable for the smart grid domain. Furthermore, many delay-sensitive smart grid applications, such as fault detection, isolation, service restoration, or Volt/VAR optimization, cannot withstand round-trip delays to contact centralized cloud systems. MEC has been suggested as a feasible cloud computing alternative to solve these restrictions. The calculation may be done closer to the data source using MEC. Furthermore, as the number of omnipresent sensors grows, the number of possible attack spots for the grid grows. Every smart IoT device is susceptible to cyber-attacks. MEC enables security mechanisms to be enforced closer to the end devices. As a result, even if an attacker gains access to an endpoint device, the assault will only acquire access to the local network segment since MEC can detect the intrusion and disable access [44].

2.2.4.6 IoT in Smart Agriculture

To satisfy future food production demands, the agricultural industry will need to undergo a substantial transformation, with IoT being incorporated into numerous production, management, and analytical operations. When compared to other sectors such as smart cities and healthcare, the agriculture industry has been hesitant to adopt developing Machine-to-Machine (M2M) and IoT technology. Using autonomous vehicles (tractors), remote monitoring, and real-time analytics, precision farming, and smart agriculture can be done. According to reports, farmers are increasingly using agricultural drones and satellites to monitor their properties and gather crop data. IoT sensors may give vital information regarding agricultural yields, rainfall, insect

infestation, and soil nutrition, which can help farmers improve their farming practices over time. Although low latency is not an essential need in the smart farming context, managing massive data quantities will be a major consideration. On-site MEC servers can help high-tech farming by gathering and analyzing huge data on agriculture to improve efficiency [45]. Similarly, MEC systems can save the farmer money on data access, synchronization, storage, and other overhead costs without shifting ordinary agricultural applications to a remote cloud.

In poultry farms, the adoption of IoT-based automated data collecting and monitoring systems may improve job productivity and service quality and get a better knowledge of chicken care. Carbon dioxide and luminosity sensing are critical characteristics in large-scale chicken houses, and sensing technologies may be utilized to measure them. Gas sensors can provide all the information needed to avoid chicken infertility caused by issues like as low carbon dioxide levels. Luminance sensors can assist in maintaining the right amount of luminosity for maximum productivity. Low latency is not as crucial for smart poultry houses as it is for smart farms. On the other hand, large data collections need the usage of on-site MEC servers. Furthermore, transferring data between poultry houses and keeping old data in centralized servers are critical for detecting unusual farm events. Poultry houses can employ MEC to function with sporadic connectivity to the centralized clouds. MEC servers can temporarily store the data in this instance until the farms are linked to the centralized clouds[46].

2.2.4.7 Industrial Internet

The Industrial Internet, often known as Industry 4.0 [47], is another IoT use case in the Industrial Internet of Things (IIoT) [48]. To boost intelligence and connection, the IIoT includes a variety of sophisticated communication and automation technologies such as M2M communication, machine learning, and big data analytics. IIoT networks can connect all the manufacturing floor employees' data and procedures and send them to the executive offices. Using the IIoT network, decision-makers or employees may get a complete and accurate picture of their manufacturing process, allowing them to make better decisions. IIoT also aids in the exploitation and application of new intelligent technologies to speed up manufacturing workforce innovation and transformation.

IIoT is primarily viewed as a means of increasing operational efficiency. However, the IIoT offers many other advantages, including increased connection, efficiency, scalability, time savings, and cost reductions for industrial processes making smart devices the most. Furthermore, these intelligent machines function with more precision, efficiency, and consistency than people. As a result, IIoT has a lot of promise in enhancing quality control, sustainability, and overall supply chain efficiency [49].

By solving the inadequacies of M2M communication in the IIoT domain (e.g., la-

tency, resilience, cost, peer-to-peer, connection, and security), MEC will play a critical role in enabling future IIoT applications. Furthermore, edge computing will be used in various IIoT deployment scenarios, according to current industry trends. Real-time edge analytics and improved edge security, for example, are two important factors in the development of new IIoT implementations. As a result, including MEC in IIoT networks will accelerate IIoT progress and open up new commercial opportunities [50].

2.2.4.8 Smart Transportation

Control, efficiency, and safety are the three main purposes of smart transportation. Intelligent Transportation Networks (ITS) employs various technologies to track, assess, and manage transportation systems to increase efficiency and safety. As a result, traveling around a city with smart transportation is more pleasant, cost-effective (for both cities and individuals), and safer. The proliferation of IoT devices and 5G communication technologies has aided these new opportunities. The former allows for integrating low-cost sensors and controllers into nearly any physical equipment that can be managed and maintained remotely. The latter offers the high-speed connectivity needed for real-time, low-latency network administration and control [51].

2.2.5 Edge to Cloud Continuum

The edge-to-cloud compute continuum is seamless integration and interaction between cloud-hosted services along with data-driven edge components to maximize the value and benefits of edge computing [52]. Specifically, edge computing is a component of a computing continuum that extends from network-connected devices and equipment to the hybrid cloud, including public and private clouds. Workloads will run wherever it makes the most sense for the company to operate them in this new IT landscape. The different environments along the continuum will collaborate to deliver the appropriate resources for the task. This section delves into different cloud-edge continuum requirements and important aspects, including resource allocation, cyber foraging, data analytics, and computation offloading.

2.2.5.1 Resource Allocation

To make the most of the available resources, the allocation of combined computing and communication resources must be carefully considered. A single MEC server will serve applications that cannot be partitioned. At the same time, offloaded programs that can be divided into multiple sections will be allocated resources on different MEC servers. Suppose adequate resources are available when a job arrives at the MEC server. In that case, the scheduler should assign the VM for further processing, and

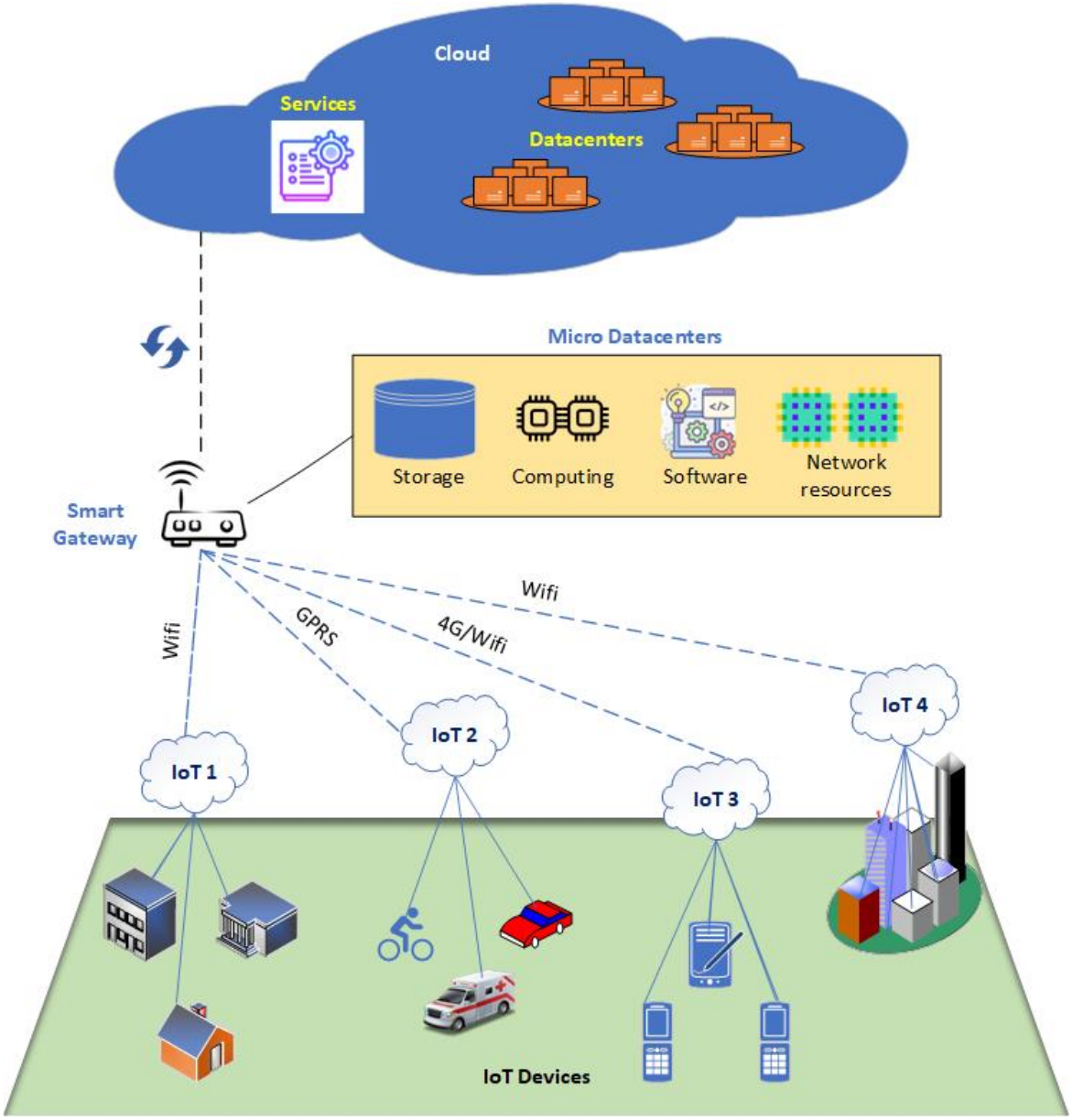

Figure 2.3: Mobile-Edge-Cloud Continuum architecture

it will delegate the job to the centralized cloud if the computational resources are insufficient.

Several essential resource allocation techniques have been developed for the cloud continuum for time-sensitive applications. However, one can distinguish between techniques that are based on the time model, consider workload tasks with difficult deadline requirements [53, 54] or that minimize response time [55, 56, 57] and other techniques based on the contention model that considers tasks to compete only for computational resources [58, 59, 60, 61].

For user application tasks and MEC service jobs, MEC servers must additionally distribute compute and communication resources. Examples include user mobility, network topology, scalability, and load balancing. The distribution of bandwidth will

become difficult when IoT gateways distribute limited bandwidth among several IoT devices that can handle video, audio, or bio-medical information [62]. Low-power wireless technologies used in IoT networks (e.g., BLE, ZigBee, low-power Wi-Fi, and LPWAN protocols like LoRa or Sigfox) have restricted bandwidth. Therefore, when IoT devices connect to the MEC server, which serves as an IoT gateway, they must use one of the low-power, low-bandwidth wireless connections.

2.2.5.2 Cyber Foraging

Cyber foraging opportunistically exploits fixed computing infrastructure in the surrounding environment to dynamically increase the computing power of mobile computers. Applications functionality is dynamically partitioned between the mobile computer and infrastructure servers that store data and do computation on behalf of mobile users in a cyber foraging system [63]. User mobility, platform features, and resource variations such as network bandwidth and CPU load all influence where program functionality is located. Cyber foraging creates a new surrogate computing layer between mobile users and cloud data centers. Surrogates are connected infrastructure servers with more computational capability than compact, battery-powered mobile devices. In addition, surrogates are geographically placed to be as close to mobile computers as possible, allowing them to deliver significantly faster network reaction times than servers in cloud data centers. Surrogates, for example, maybe found around wireless hotspots in coffee shops, airline lounges, and other public places [64].

In the context of computational mobility in the IoT, Cyber foraging is a critical skill for maximizing resource use. It allows for dynamic offloading of calculations to nearby mobile nodes or remote cloud-based servers, retrieval of results from the offloaded computations, and then the continuation of mobile business logic execution [65].

2.2.5.3 Data Analytics

The act of gathering, evaluating, managing, processing, and using vast collections of data from diverse sources and existing in many formats, structured, semi-structured, or unstructured, is known as data analytics or Big Data analytics [66]. Companies must know how to acquire, store, protect, manage, and analyze data efficiently when faced with ever-increasing amounts of data on-site, in the cloud, online, and offline. As a result, any company's capacity to exploit Big Data through data analytics has become a vital strategic concern. Organizations must leverage particular Big Data tools, technologies, systems, and infrastructures.

Many works present an overview of initiatives to use Big Data Analytics technologies in the Edge-to-Cloud Continuum [67, 68, 69, 70]. They show how to use Data Analytics platforms (such as Hadoop, Flink, Spark, Storm, Nifi, and others) and Machine Learning libraries (such as Spark MLlib, TensorFlow, Keras, Scikit-learn, and

others) to build a real-time Big Data pipeline from the Edge to the Cloud. Therefore, they explored three open challenges: interoperability, defining smart city applications, and privacy concerns.

2.2.5.4 Computation Offloading

Computation offloading is one of the primary goals of edge computing since it allows mobile devices to overcome their physical resource's limits such as processing capability, battery life, and storage capacity. Unfortunately, it's challenging to know when, and how, to offload compute cores. Various ways have been offered to address this problem in multiple contexts, including single users, multi-user, etc. The common design goal for the single-user situation is to save energy for the mobile device. However, the optimum offloading issue in the multi-user situation is frequently NP-hard [71]. By employing D2D communication in next-generation heterogeneous networks, compute workloads may be offloaded to servers but also devices [72].

2.2.6 Cloud Computing, VMs and Containers

One approach to improve security and isolation in the IoT is the use of virtualization. Indeed, virtualization becomes a technology enabler in several embedded systems space, because it has proven to enhance the quality of software and decrease design time [73]. Non-real-time components can still coexist in the same infrastructure without compromising the real-time nature of embedded systems, and still ensure the real-time integrity of the system. Furthermore, security by isolation is one among other methods that improve the security of the overall system, in which the hypervisor assures that one domain's execution does not interfere with that of other domains [74]. The hypervisor may achieve this isolation by ensuring that software operating in virtual machines (VMs) is oblivious to software running on different VMs, even if they are on the same physical hardware.

Simultaneously, advancements in the creation of IoT applications sparked debate regarding the security weaknesses posed by IoT devices. Additionally, billions of these devices will be connected to the cloud and share data. Hackers will be able to take advantage of such flaws, putting apps tied to such devices at risk. Due, virtualization must be considered a feasible strategy for IoT security [75].

Using container virtualization can be a lightweight alternative to hypervisors that meets the demands of most embedded virtualization applications. Furthermore, container virtualization is built on system calls, which is an effective strategy [76].

2.2.7 Keys Challenging Issues in Designing and Managing IoT Networks

The Internet of Things has capabilities that will meet future needs. However, it raises a new set of intriguing research questions about IoT architecture, trusted device connections, security problems, etc. The Internet of Things (IoT) design brings together various technologies and substantially influences IoT applications. This part will go through the critical challenges in IoT networks.

2.2.7.1 Scalability

Because of the large number of IoT devices, network scalability is a big concern in the IoT network. In addition, maintaining the state information of many IoT devices must also be considered. So, we'll want to ensure sure the IoT system can accept an ever-increasing number of endpoints as they arrive. This tremendous deluge of data traffic causes significant Quality-of-Service such as bandwidth difficulties and message delivery, although this system must not suffer from network growth, especially in mission-critical commercial and industrial use cases. The underlying wireless technology mainly dictates this. As a result, selecting the appropriate technology and design is critical to ensuring the long-term survival of any linked system.

2.2.7.2 Interoperability

Every IoT system is a jumble of disparate components and technology. Therefore, interconnecting diverse networks seamlessly is a vital issue and keeping interoperability a must for IoT scalability to avoid getting stuck with an outdated system that can't keep up with future innovation. In addition, many IoT devices will be connected to heterogeneous smart networks or applications via communication technologies to communicate, distribute, and gather crucial information.

The lack of interoperability prevents devices manufactured and designed by different companies from connecting autonomously, discovering each other, and collaborating with other smart devices and services, which is an obstacle to creating automated ecosystems such as for the deployment of smart homes or smart homes city.

2.2.7.3 Constrained Resource

The Internet of Things will provide challenges in resource efficiency in low-power, resource-constrained devices. For example, storage, computation, bandwidth, and energy are all resources in a typical battery-powered IoT device. This massive volume of data produced by IoT devices raises the need for processing and storage resources. Given the normal resource restrictions of IoT nodes, it becomes essential to incorporate

a few high-end nodes in the IoT ecosystem, such as edge devices, smart gateways, and cloud platforms. Despite these different ideas, resource constraints remain a significant barrier in IoT systems.

2.2.7.4 Security

The expansion in the volume of IoT devices means more devices are in operation and can always create more security threats. Thus, it is urgent to secure the network against malicious attacks. This is because each additional device increases the attack surface and increases the number of people accessing the network.

Botnets are a huge security concern to IoT devices because they are always looking for vulnerabilities. They can knock down large systems by employing a Distributed Denial of Service assault, which is a tactic that floods a system with messages and requests to take it down if they locate an open back door. A DDoS assault on an IoT fleet provider might be the final nail for a specific project or the entire company. Maintaining high levels of security between devices and the network as you scale up production and deployment may be a significant challenge.

2.2.7.5 Privacy-Preserving

Smart cities, industrial companies, health institutions, and governments are all data sources for IoT-based systems. These examples require various types of networks whose communication performance can handle billions of devices communicating with each other while guaranteeing privacy protection against attacks, knowing that IoT attackers always rely on data collected to launch their attacks.

IoT Privacy-Preserving, most of a critical challenge, protects a device's identification during data transfer while maintaining a steady level of network performance.

2.2.7.6 Energy Efficiency

The device's consumption is heavily influenced by how long it is continuously detecting, processing, and transmitting/receiving potential data. Simultaneously, IoT devices are often unable to broadcast across long distances because of their energy limits. Thus, having more sensor samples improves data interpretation, but they also increase power consumption. Therefore, before deciding on additional activities, it is vital to define how long the device will be sensing and the strategy for manipulating and communicating with the obtained data.

2.3 Software Defined Network and Network Function Virtualization

Today's IoT services and applications need an efficient network architecture to support scalable, interoperable, and predictable IoT systems. Such a network architecture must be flexible enough to be reprogrammed following any change in IoT application needs. Furthermore, IoT usually adopts different communication schemes than the traditional Internet, stemming primarily from various resource-constrained hardware platforms. Software-Defined Network (SDN) [77] has emerged as a flexible and agile network framework that enables reengineering of the network stack to deliver dramatic improvements in the network agility and flexibility and solve the resource management needs of the IoT environment [78].

Additionally, SDN enables the rapid movement of workloads across a network. For example, it allows slicing a virtual network into partitions, using Network Function Virtualization (NFV) to enable telecom providers to move IoT services to less expensive commodity servers. SDN makes it easier for any network to adapt and grow as network administrators. SDN's inherent speed and flexibility enable it to support emerging MEC and IoT trends, which require fast and simplified data transfer between remote sites.

In this section, we first introduce the fundamental concepts behind SDN (in Section 2.3.1) and show how it enables refactoring of the IoT network through the OpenFlow protocol in Section 2.3.2. Then, we introduce the SDN controller functionalities in Section 2.3.3 and SDN applications in Section 2.3.4. Next, in Section 2.3.5 we describe how NFV technology can be seen as a complementary technology to SDN and illustrate different functionalities of NFV. Finally, in Section 2.3.6 we delve into different approaches that leverage Network Softwarization in the context of IoT.

2.3.1 SDN Principles

In SDN, the control plane is detached from the forwarding plane, and communication between the two planes is accomplished through APIs such as OpenFlow [79]. On the other hand, SDN involves a centralized control plane regulating network traffic flows through application program interfaces (APIs). When a packet of data arrives at a particular switch in a network, for example, the switch's guidelines dictate where the packet is forwarded. As a result, network managers may deliver services throughout the network using SDN, regardless of the hardware components.

The layered architecture of an SDN is shown in Figure 2.4, which is made up of three levels: i) *control plane*, which serves as a brain for the network, managing it from a global perspective, ii) *data plane*, which is made up of dumb forwarding devices, such

as routers and switches, that only forward data in response to controller commands, and iii) *management plan*, which is responsible for the administration of the SDN network.

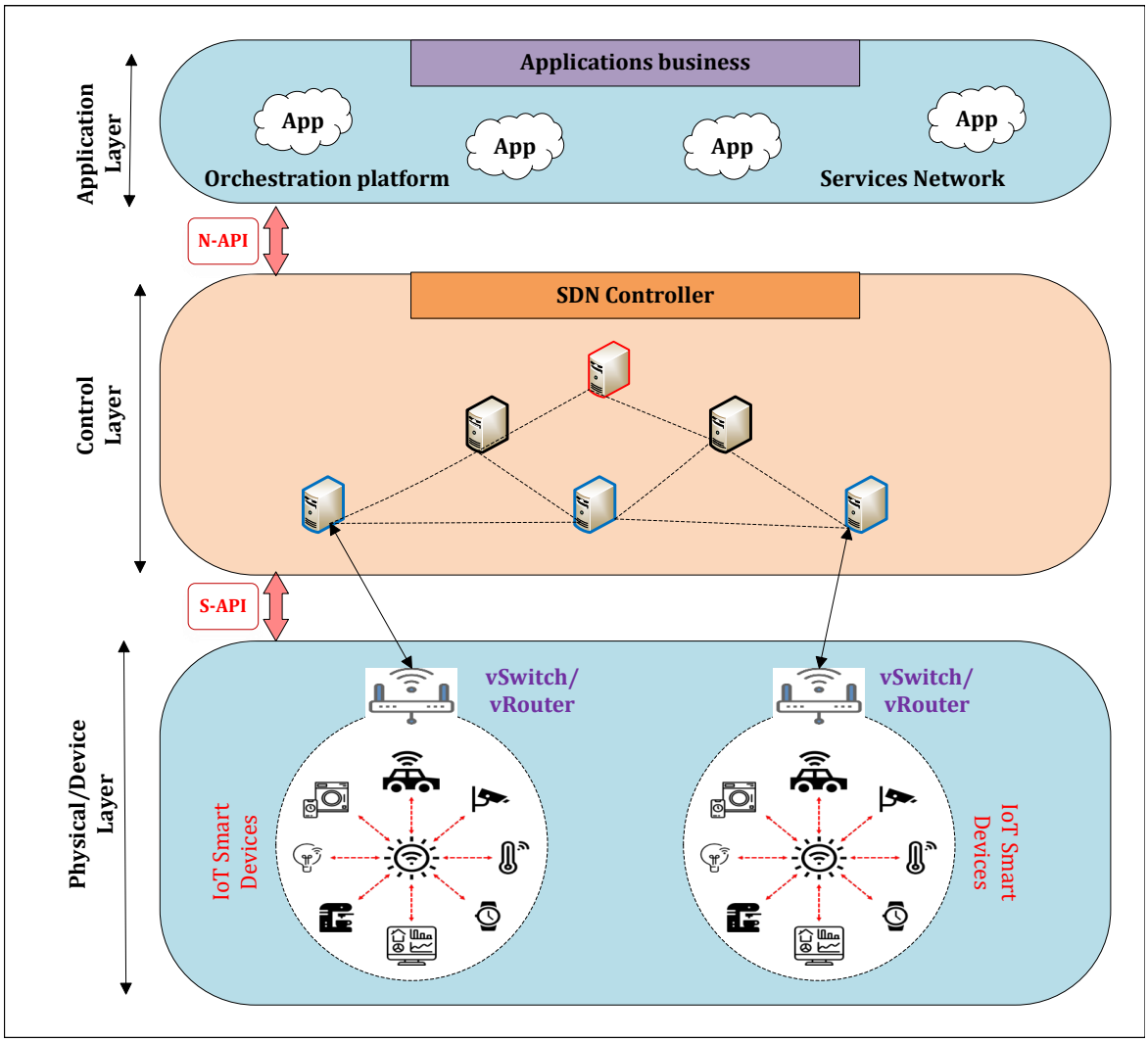

Figure 2.4: SDN layers

- The SDN data plan is responsible for routing packets from point A to point B through routers/switches based on information contained in tables. The data plan can also be defined as managing traffic that is not destined for the equipment itself, as opposed to the other two plans. It is an abstraction of the classical data plan, i.e., replacing hardware pipelines in classical routers with software pipelines in the form of flow tables.

- The SDN control plan is used to control the data plan by establishing the rules that the router must follow to route packets. Therefore, the routing protocols in traditional routers such as OSPF, STP, ARP, or BGP are implemented in software form in the SDN controller.

- The SDN management plan concerns all the tasks for network management. It is sometimes considered as a subset of the control plan [80].

SDN success is the centralized control plane design, which allows networks to be programmed utilizing several applications on a single entity. The centralized control plane of SDN architecture allows for network innovation and administration. On the other hand, large-scale networking environments cannot be controlled by a single central controller. As a result, creating a logically centralized but physically dispersed control plane is possible, which provides the most basic view of central control logic while also providing scalability and dependability [81]. The static mapping of network components with the controller, regardless of the overall load fluctuation of an individual controller and entire cluster, is one of the main concerns of network operators in a logically centralized SDN architecture.

SDN creates a central point that manages the control plane, while the physical switches/routers would only have to deal with the data plane. OpenFlow is a standard protocol that allows the controller to transmit instructions and program the flow table of a device. These instructions, so-called *OpenFlow flow entries*, are rules with a pattern (source or destination IP, mac address, TCP port, etc.) and a corresponding action (reject the packet, transmit on port x, add a VLAN header, etc.).

2.3.2 OpenFlow Protocol

An OpenFlow protocol [82] is generally composed of a message layer, a state machine, a data model, a system interface, and a configuration part. The following figure shows the different components of the OpenFlow protocol.

- *The flow table* and the *Group table* are in charge of processing and forwarding incoming packets,
- *OpenFlow Channel* is the interface that allows the interconnection of the underlying SDN routers/switches with the controller.

OpenFlow protocol components are illustrated in Figure 2.5 and is composed of the following components:

Message Layer: The message layer is the central part of the protocol stack. It defines the structure and semantics valid for all messages. In addition, it supports the ability to construct, copy, compare, print, and manipulate messages.

State machine: The state machine defines the basic low-level behavior of the protocol. Typically, it is used to describe actions such as negotiation, discoverability, flow control, delivery, etc.

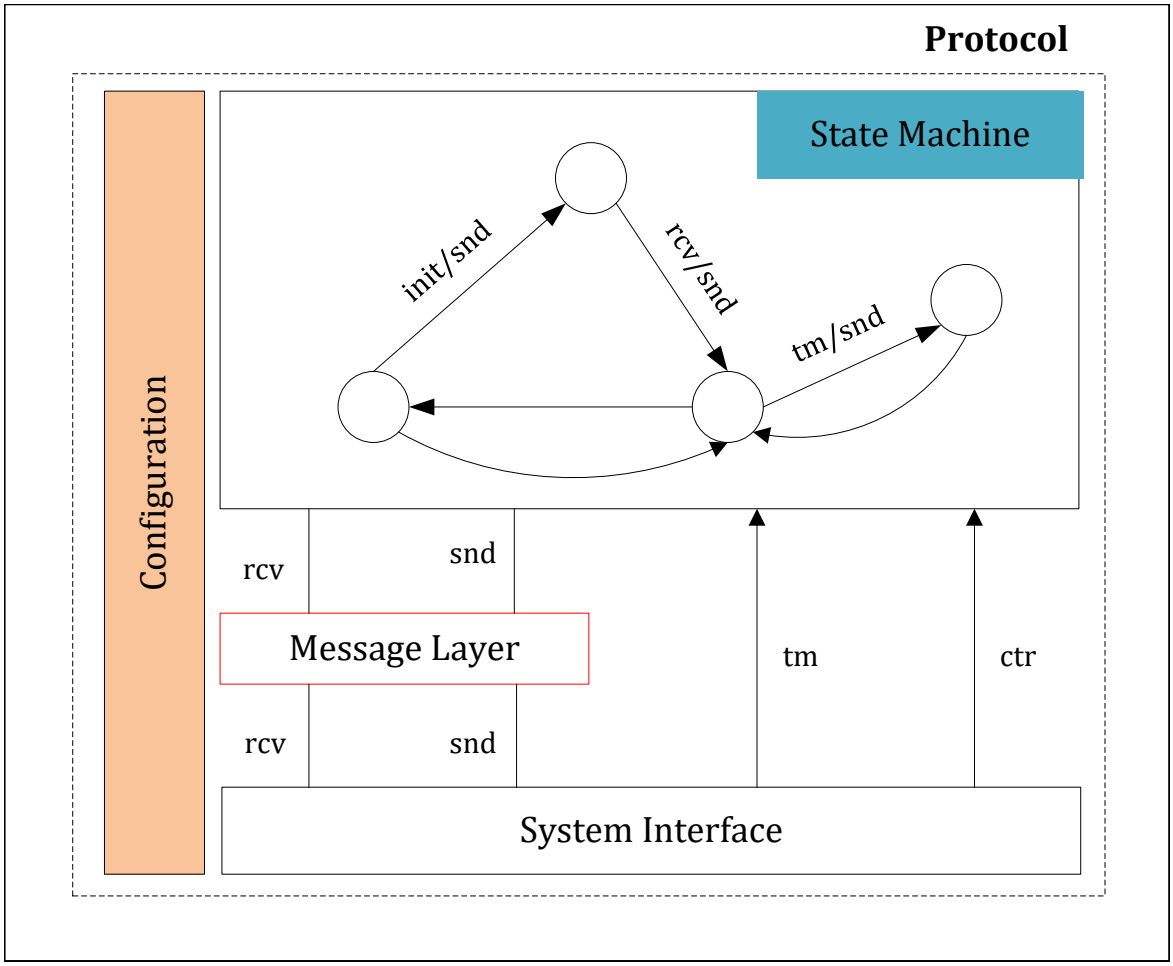

Figure 2.5: OpenFlow protocol components

System Interface: The system interface defines how the protocol interacts with the outside world. It typically identifies required and optional interfaces and their intended use, such as TLS and TCP as transport channels.

Configuration: Almost all aspects of the protocol have initial configurations or values. The configuration can cover anything from the size of the default buffer to responding to X.509 certificate intervals.

Data model: Another way to look at the OpenFlow protocol is to understand its underlying data model. Each switch maintains a relational data model containing attributes for each OpenFlow abstraction. These attributes describe either an abstraction capability, its configuration state, or a set of current statistics.

2.3.3 SDN Controller

The SDN controller offers services that enable the realization of a distributed control plane and ephemeral state management and centralization principles. An SDN controller is a software system or a set of software systems that collectively provide:

- A database that is used to maintain the network state and manage and distribute this state in some circumstances. The database serves as a store for data obtained by controlled network devices and related software and data managed by SDN applications, e.g., network status, ephemeral configuration data, learned topology, and control session data. In addition, the controller may have numerous purpose-driven data management procedures in various circumstances (e.g., relational and non-relational databases). Other in-memory database techniques can also be used in some instances.

- A high-level data model depicts the connections between the controller's, the underlying resources, policies, and other services. In many circumstances, the Yang modeling language [83] is used to create these data models.

- The controller services are exposed to an application using the RESTful API interface. This interface is produced from the data model representing the controller's services and features. The controller and its API may be part of a development environment that creates API code from the model in some situations. Some systems go even farther, providing powerful development environments that enable the growth of core capabilities and the subsequent publication of APIs for additional modules, such as those that offer dynamic controller expansion:

    1. A secure TCP control session between the controller and the network nodes' associated agents.
    2. A protocol for supplying application-driven network state on-network devices based on standards.
    3. A technique for discovering devices, topologies, and services; a path calculation system; and maybe additional network-centric or resource-centric information services.

2.3.4 SDN Applications

In a software-defined networking context, an SDN application is a software program that performs a task. SDN applications can replace and extend functionalities performed by firmware in traditional network hardware devices.

Northbound APIs, as shown in the above Figure 2.4 provide applications with an abstract view of the underlying network. Network parameters such as delay/throughput descriptors and resource availability can be included at this level of abstraction. Based on the relevant criteria, applications seek communication between nodes. The SDN controller configures network elements in the data plane once the optimum path has been determined. Therefore, SDN applications can include programs for network

virtualization, network monitoring, intrusion detection (IDS), and flow balancing (the SDN equivalent of load balancing), among many other possibilities.

You can write scripts and automate the network administration through the API, where several applications can be developed and accessed by the SDN controller. The scripts can be written in Java or Python and can also use the API to retrieve information from the SDN controller or even configure the network directly. Java and Python are the most used, but we can also develop with other languages like C++, and it all depends on the type of SDN controller is the case for the first NOX SDN controller.

2.3.5 Network Function Virtualization

One of the most interesting complementary technology of SDN, which has the potential to impact the future 5G networking dramatically and how to refactor the architecture of legacy networks, is virtualizing as many networks functions as possible, so-called Network Function Virtualization (NFV). NFV and SDN are two related but distinct technologies that are driving the telecom industry's digital transformation of network infrastructure [84]. NFV aims to virtualize a set of network functions by deploying them into software packages, which can be assembled and chained to create the same services provided in legacy networks. Along with SDN, NFV allows enforcing the security policies of the SDN network by enabling the deployment of virtualized network functions inside virtual appliances. NFV refers to replacing network appliance hardware with virtual machines that run software and processes under the control of a hypervisor.

NFV is a project that aims to replace proprietary hardware-based network services with virtual machines, with a virtual machine being defined as an operating system that imitates specialized hardware. Virtual machines provide network operations such as routing, load balancing, and firewalls in NFV. Resources are no longer tied to data centers when using NFV; instead, they pervade the network to increase the productivity of internal processes. Furthermore, NFV allows communication services to be separated from dedicated hardware components, such as routers and firewalls. Thus, network operations can dynamically provision new services without installing new hardware. In addition, virtualized services can run on generic servers instead of proprietary hardware. Other benefits of virtualizing network functions include implementing on-demand pay-per-use models, reduced operational costs due to fewer appliances, and rapidly scaling the network architecture.

The Standard Developing Organization ETSI is the reference that defines the NFV high-level functional architecture, as shown in Figure 2.6. High-level indicates that its specified job defines the available critical blocks, architecture, and inter-relationships. Virtual Network Functions (VNFs); NFV Infrastructure (NFVI), which includes the elements needed to run VNFs such as the hypervisor node and virtualization clusters;

and MANO (Management and Orchestration), which handles the operations required to run, migrate, and optimize VNF nodes and chains, possibly in collaboration with transport network orchestrators [85].

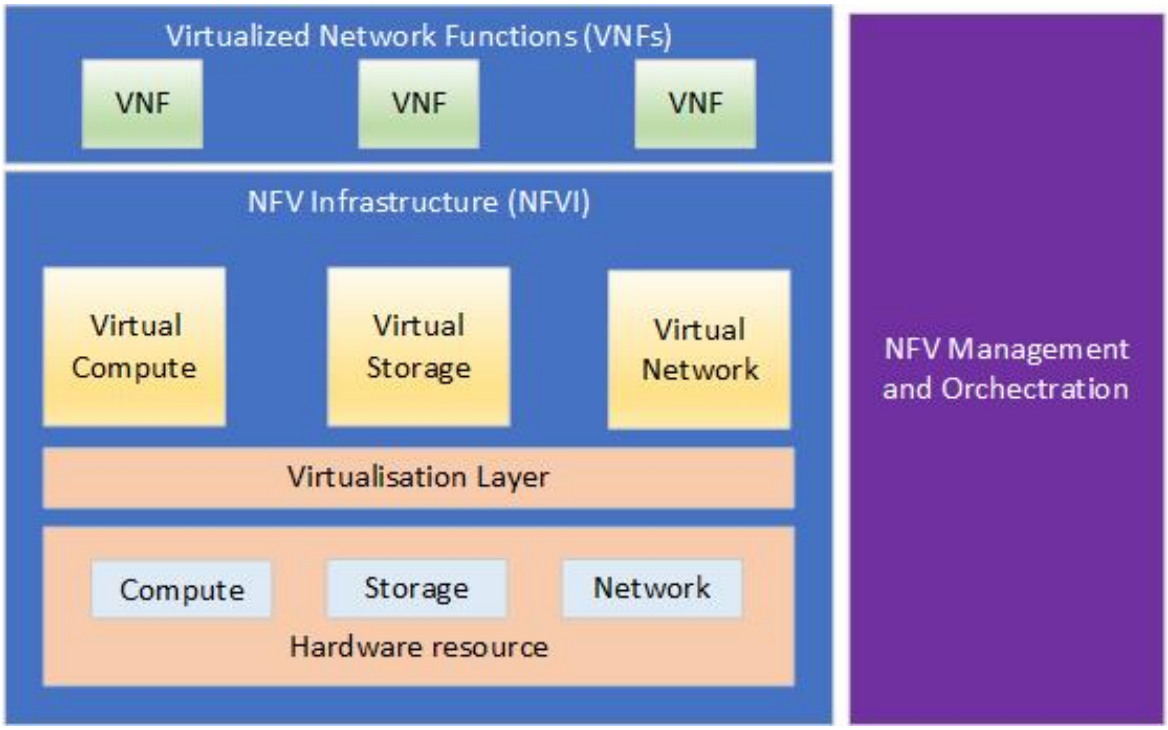

Figure 2.6: General NFV architecture by the ETSI

The NFVI is a set of software (e.g., virtualization hypervisor) and hardware (e.g., servers, storage, and network) resources that provide an NFV environment for the deployment of VNFs. A distributed collection of VNF nodes, also known as NFVI-PoPs, can be considered such infrastructure. Each NFVI-PoP is an abstracted physical location of the network infrastructure (e.g., data center, network node, or end-user premise) with a restricted computing capability to host the VNFs.

VNFs are NFVI-based software implementations of network functions (such as firewalls). A single VNF might be composed of numerous internal components executing on different virtual resources like VMs (Virtual Machines). For example, a network service in a telecommunications network is made up of one or more network functions. VNFs implement the network functions that make up the network service on virtual resources in the case of NFV. As a result, the network service's behavior in an NFV context is determined by the behavior of the constituent VNFs.

Finally, the NFV MANO provides the functionality for VNF provisioning and all management operations, such as placing and instantiating VNFs to meet user requests better, configuring VNFs according to active requests while meeting common traffic engineering objectives in IP transport networks, as well as minimizing the number of VNF instances to install.

2.3.5.1 Network Slicing

On top of a single physical infrastructure, network slicing allows operators to run several virtual networks. The main goal of network slicing is to construct and separate different services on a network so that operators can give the best possible support for those services. Even if they share the same underlying physical network, each slice manages its packet forwarding without interfering with other slices. Figure 2.7 depicts a multi-tenant architecture connected to a virtual overlay network through OpenFlow switches to deliver private or public cloud services. This use case demonstrates how partitioning network resources into several partitions allow tenants to use their virtual L2 slices without interfering with those of other tenants. The SDN controller can handle per-tenant/per-slice instantiation, while also providing policy-based network management and flexible resource allocation [86]. This indicates that the controller can adhere to its rule of providing robustness, stability, and scalability regarding the number of tenants, concurrent experiments, and controlled resources. NFV and SDN serve as a basis for network slicing [87] by allowing both physical and virtual resources to be used to deliver specialized services and providing the necessary components for network slicing [88].

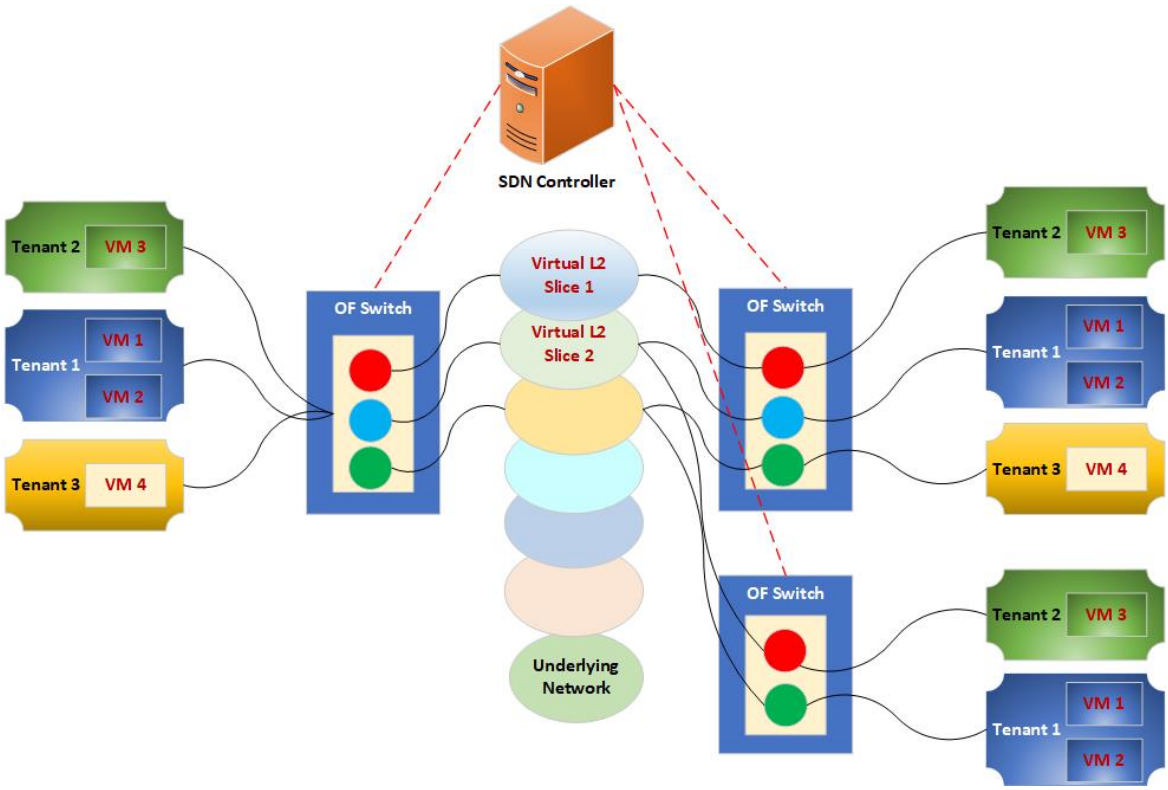

Figure 2.7: OpenFlow Network Slicing use case

Another area where NFV and SDN are likely to play a significant role in network slicing is radio access networks (RAN). Both NFV and SDN virtualize the network components of each slice in the core of a network to meet individual demands. In

the RAN, network slicing may be enabled by physical radio resources. NFV and SDN is projected to be used to support various RANs as well as the many service types that run over those RANs [89]. In addition, the commercialization of NFV and SDN is likely to accelerate in the future years, allowing network slicing to enable flexible network deployments to meet the demands of varied applications and services. SDN may also be utilized to create an overall framework for 5G to operate across a control the plane, deciding the best path for traffic flows throughout the network [88].

2.3.5.2 Service Chaining

Another concept that received a lot of attention with the evolution of SDN and NFV is the Network Service Chaining (NSC) [90]. The NSC aimed to help network operators to provide continuous delivery of services based on dynamic network function orchestration and automated deployment mechanisms to improve operational efficiency.

For example, end-to-end data flows are expected to pass through a succession of NFV functions, which is addressed by the idea of service chaining, such as the case of going through the firewall, encryption, load balancing, and decryption. In an NFV context, the technology's characteristics allow many virtual network functions to be joined together. These connections may be built up and broken down as needed with service chain provisioning through the NFV orchestration layer, since it's done in software utilizing virtual circuits.

2.3.5.3 VNFs Placement

MEC deployment may also make use of Network Function Virtualization (NFV) platforms, which removes the limitations imposed by older LTE networks, such as fixed network functionality placement [91]. NFV enables the development and placement of virtual network functions (VNFs) in a flexible and on-the-fly manner, supporting a wide range of application needs while also improving the management of heterogeneous (network, compute, and storage) resources. As a result, application and network capabilities are handled as VNFs and controlled by an NFV Orchestrator [92] and may be executed in a distributed system's many locations, including, i.e0., NFV-enabled MEC nodes.

Integrating SDN, NFV, and MEC technologies is critical for 5G development. Together with SDN and NFV, MEC will play a role in addressing the issues that 5G hopes to overcome. Hence, EC may be thought of as a kind of NFV. Any NFV-capable network node on which MEC services can be hosted with assigned VNFs is considered a MEC node. MEC entails putting computational power in new places. Network functionality and end-user applications may be involved in this computation. None of the studies on MEC and SDN integration look into using NFV to install specific VNFs to provide end-user services like VoIP, video streaming, or web surfing.

The benefits of SDN, NFV, and MEC, VNFs must be supplied with enough resources on edge servers to avoid affecting network Quality of Service (QoS).

2.3.5.4 Migration of VNFs

The network evolves due to the dynamic nature of edges. For example, the network topology may change, a machine's resource capacity may change at some time, and the bandwidth between two computers may rise or decrease owing to varying traffic scenarios. As a result, some VNFs may require more computing resources during the day due to high traffic situations, and they must be migrated [93] to central network node machines with massive resources; during the night, due to lower demand for computing resources, some VNFs must be migrated back to edge network nodes with fewer resources, allowing the machines with massive resources to be vacated for large-scale tasks [94].

As a result, we must promptly accomplish the VNF chain migration and the VNF connection recovery duty after migration.

2.3.5.5 Dynamic Management of Service Chaining

Physical devices are replaced with virtual machines in SDN and NFV, which reduces an operator's time to market for a new service. The reason for this is that under SDN, the centralized controller has a global view of the network, allowing for the creation of new service chains with a simple change in policy associated with a flow [95].

Dynamic service chaining is likely to revolutionize how network operators offer services to their subscribers, allowing them unprecedented flexibility in designing, deploying, managing, and upgrading services that are individually customized to the demands of their clients.

2.3.6 Network Softwarization in IoT

The Internet of Things sector is always searching for new capabilities to enable cutting-edge service developments while also allowing rapid deployment and corporate agility. Software-Defined Networking (SDN) and Network Function Virtualization (NFV), as major vital drivers in the evolution of IoT [96], are being positioned as central technology enablers for decoupling IoT hardware from service deployment, leveraging an increasing number and type of services supported over a single deployed IoT platform.

Adopting the virtualization idea within IoT devices is expected to result in significant cost savings in the service providing, while also allowing for the rapid introduction of new and innovative services to the market. This allows IoT device vendors to open up their platforms to a large extent, while also allowing IoT service providers to avoid

vendor lock-in with proprietary hardware and software [97]. Furthermore, by simplifying network administration with centralized management and control of IoT devices from many manufacturers and offering chances for cooperation and interoperability, the IoT service provider will have greater control over the IoT devices.

Adoption of SDN and NFV opens up a world of possibilities in terms of control and monitoring, distributed QoS enforcement, network resource management, in-network data processing and storage, fusion and detection of complex events, and so on [97].

2.4 Deep Reinforcement Learning for Edge Computing Applications

Deep learning has been successfully used in a variety of fields [98], including computer vision, natural language processing (NLP), and artificial intelligence (AI). Deep learning has distinctive attributes compared to traditional machine learning in terms of feature learning, model building, and model training, as shown in Figure 2.8. It integrates the learning of features and building models in a single model by selecting different kernels or tuning parameters through end-to-end optimization. Deep learning has shown tremendous information extraction and processing skills when compared to typical machine learning approaches [99]. Furthermore, Deep learning advances have considerably increased edge computing applications in numerous scenarios, enhancing performance, efficiency, and management.

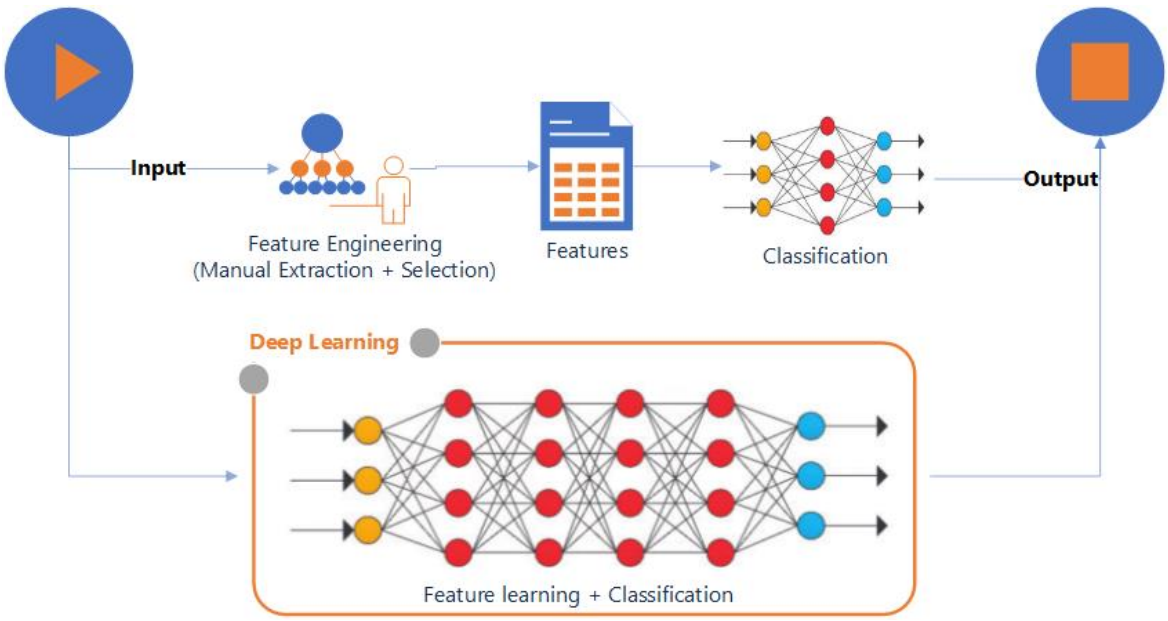

Figure 2.8: Comparing deep learning with machine learning

This section presents different DRL algorithms used and examined in this thesis. We first introduce the exploitation, Restricted Boltzmann machine (RBM), and diverse

neural network approaches used in DRL.

2.4.1 Exploration

Exploration is one of the most challenging aspects of Reinforcement Learning. Indeed, learning agents must control investigation even if no useful information is obtained. For example, the agent has done some environmental observation but has not attained the goal. He can't determine if the goal has been found just by looking [100]. Whereas, if the agent never learns how to gain points, he will never be able to improve. This is already an issue in games. It becomes considerably more so in real life. A simple concept would be to reduce the amount of exploration gradually. However, when faced with real-world activities, something in the environment may change throughout the run, necessitating a re-exploration.

2.4.2 Restricted Boltzmann machine (RBM)

Restricted Boltzmann (RBM) machines are probabilistic graphical models that resemble stochastic neural networks. A typical two-layer RBM consists of a visible layer with the known input and a hidden layer with the latent variables [101]. RBMs are arranged as a bipartite graph, with each visible neuron connecting to all hidden neurons and vice versa, but no two units in the same layer are connected. RBMs have found use in various fields, including collaborative filtering and network anomaly detection. A deep belief network (DBN) comprises a visible layer, and many hidden layers made up of multiple stacked RBM layers. A DBN is trained layer by layer, with each layer being viewed as an RBM trained on top of the previously trained layer [102]. The structure of DBNs may be used for a variety of applications, including defect detection categorization in industrial contexts, threat identification in security warning systems, and picture emotional feature extraction. RBM has been used for diverse IoT applications [103], such as intrusion detection in smart cities [104], predictive maintenance in smart factory [105], localization [106], image recognition [107], and healthcare [108].

2.4.3 Autoencoder

AutoEncoders (AEs) are neural networks that aim at coping their input layer to their output layers, coupled by one or more hidden layers [109]. First, the input is compressed into a latent-space representation to reconstruct the output.

The AE is made up of two components: an encoder and a decoder, as is shown in Figure 2.9. The former learns the input's representative qualities, compresses the input into a latent-space representation, and converts them to other latent features, i.e., usually in a compressing way. On the other hand, the latter takes the encoder's

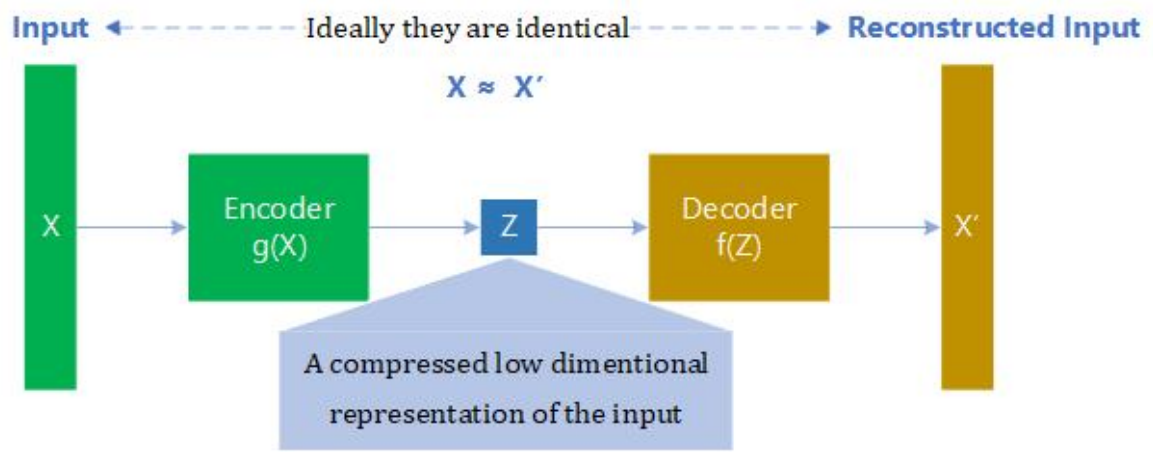

Figure 2.9: The architecture of Autoencoder

latent characteristics, reconstructs the input from the latent space representation, and recreates the input data in its original form while minimizing the reconstruction error. Similarly, an AE may be built as a deep architecture by stacking numerous layers into the hidden layer.

2.4.4 Deep Neural Networks

Deep neural networks (DNN) have a deeper layer structure than typical artificial neural networks (ANN), which have a shallow structure for more sophisticated learning tasks. A DNN is made up of an input layer [110], numerous hidden layers, and an output layer, with each layer's output being fed to the next through activation functions. The final output, which represents the model forecast, is generated at the last layer. In the training process, optimization methods such as Stochastic Gradient Decent (SGD) [111] and back propagation [112] are commonly utilized. In feature extraction, classification, and function approximation, DNNs are commonly utilized.

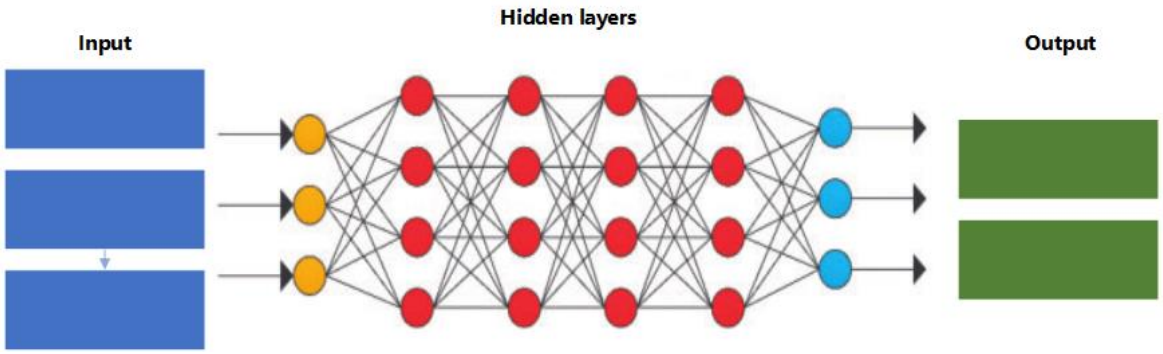

Figure 2.10: Deep Neural Network

2.4.5 Convolutional Neural Networks

Convolutional neural networks (CNNs) are used to analyze data in several arrays, such as a color picture of three 2D arrays storing pixel intensities in each color channel. The foundation of CNN design is convolutional layers, which receive 2D data structures and extract high-level features, as shown in Figure 2.11. abstraction [113].

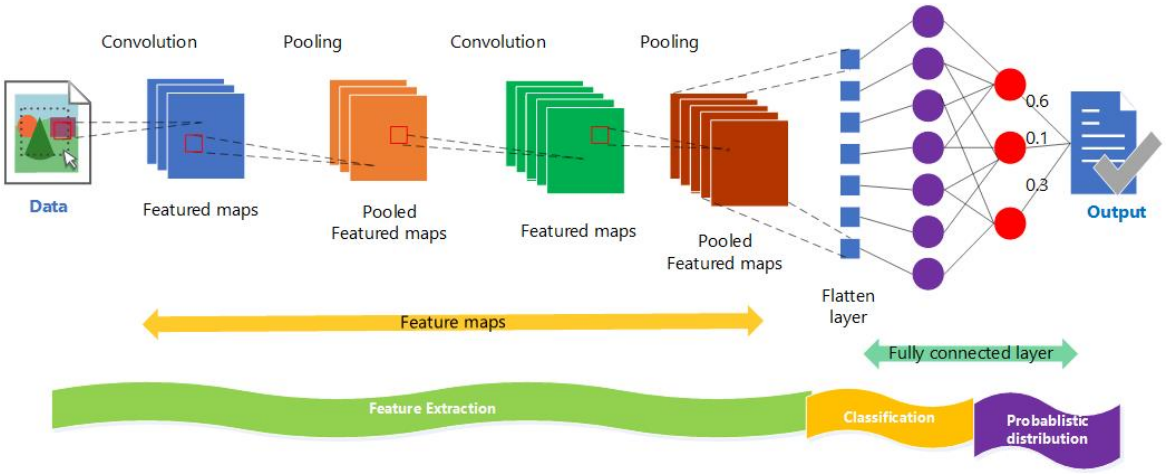

Figure 2.11: Convolutional Neural Network

CNN extracts spatial correlations between neighboring data by computing the inner product of the input and the filter as it goes through the 2D data with a set of moving filters and pooling functions. The output is then sent to a pooling block, which reduces the spatial dimensions and generates a high level.

2.4.6 Deep Reinforcement Learning (DRL)

DRL is a hybrid of deep learning and reinforcement learning (RL) [114]. Its goal is to create an agent that can learn the optimum action choices over a collection of states by interacting with the environment to maximize long-term accrued rewards, as shown in Figure 2.12. Unlike classic RL, DRL represents the policy using a deep neural network because of its high representation capacity to approximate the value function or direct strategy. Deep Q-Learning (DQL), Double DQL, and Duel DQL are examples of value-based DRL. In contrast, deep deterministic policy gradient (DDPG) and asynchronous advantage actor-critic (A3C) are policy-gradient-based DRL. The DRL has been applied to various fields, including computer gaming, chess games, and rate adaptation.

2.4.6.1 Deep Q Learning

The Q-Learning method generates an accurate matrix for the working agent, which it may ”refer to” to optimize its long-term reward [115]. Although this strategy is

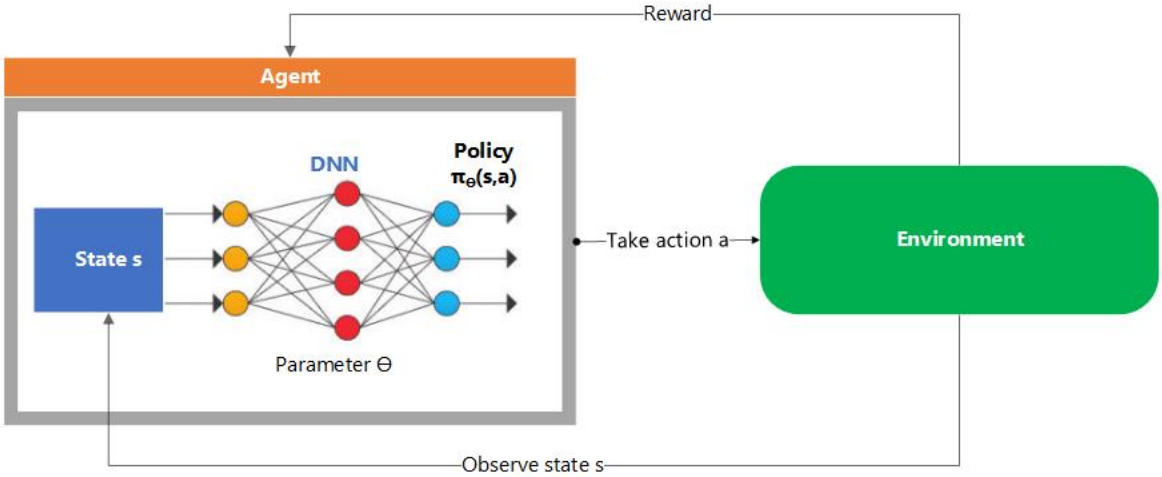

Figure 2.12: Deep Reinforcement Learning

not inherently flawed, it is only practicable in relatively small settings and soon loses viability as the number of states and actions in the environment grows. The answer to the preceding dilemma comes from realizing that the values in the matrix only have relative relevance. That is, they only matter concerning the other values. As a result of this reasoning, we have Deep Q-Learning, which approximates values using a deep neural network. As long as the relative significance is maintained, this approximation of values is not harmful. Deep Q-Learning's working step is to feed the starting state into the neural network, which then returns the Q-value of all potential actions as an output.

2.4.6.2 Dueling Deep Q Learning

The Dueling DQL's network architecture can be broken down into two key components [116]: value function and advantage function, as shown in Figure 2.13. The value function is used to describe how nice it is to be in a specific condition, while the advantage function may be used to compare the relative relevance of one action to another. Finally, the outputs of the value function and the advantage function are coupled back to the last layer to determine the final Q-value.

2.4.6.3 Asynchronous Advantage Actor Critic (A3C)

A3C is a multi-thread variant of the actor-critic approach that seeks to speed up convergence, expand the search area, and solve the problem of highly correlated samples collected by a single agent [117] as shown in the Figure 2.14. Multiple actor agents interact with replicas of the environment concurrently in the A3C method, collecting experiences used by numerous learners to train the model.

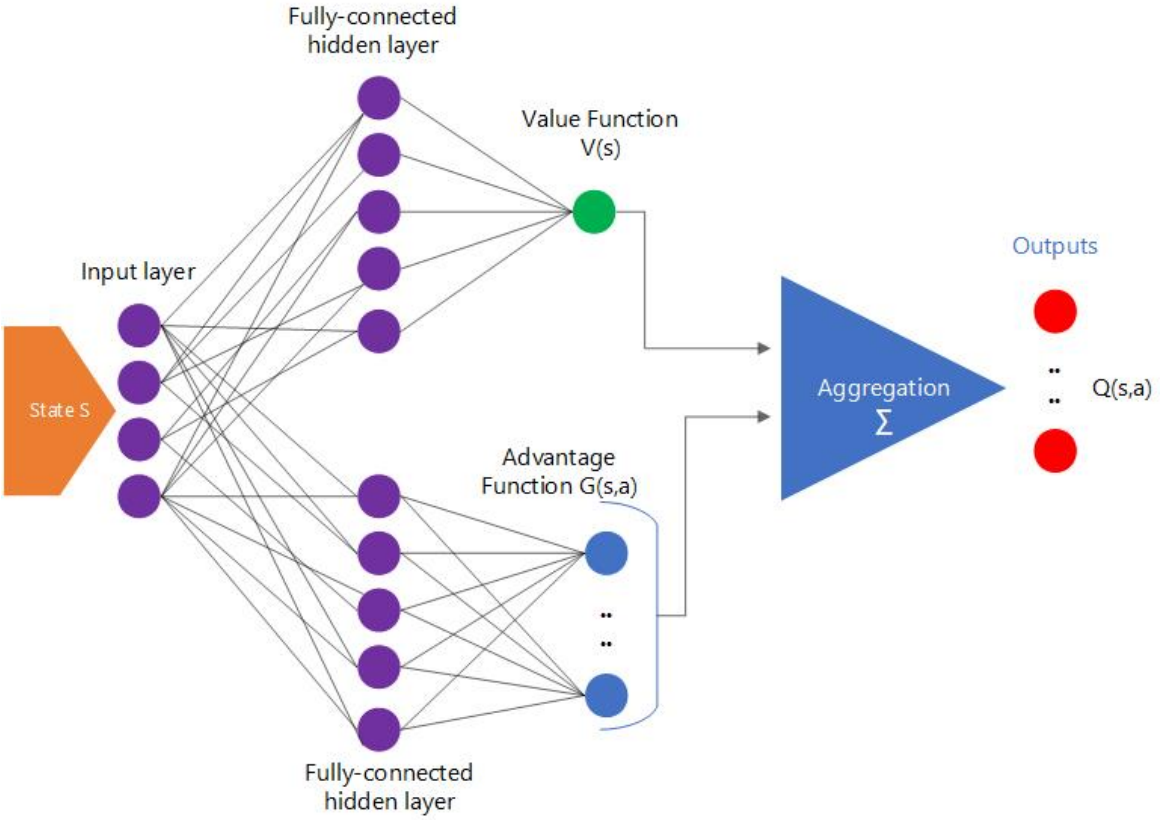

Figure 2.13: Dueling Deep Q Learning

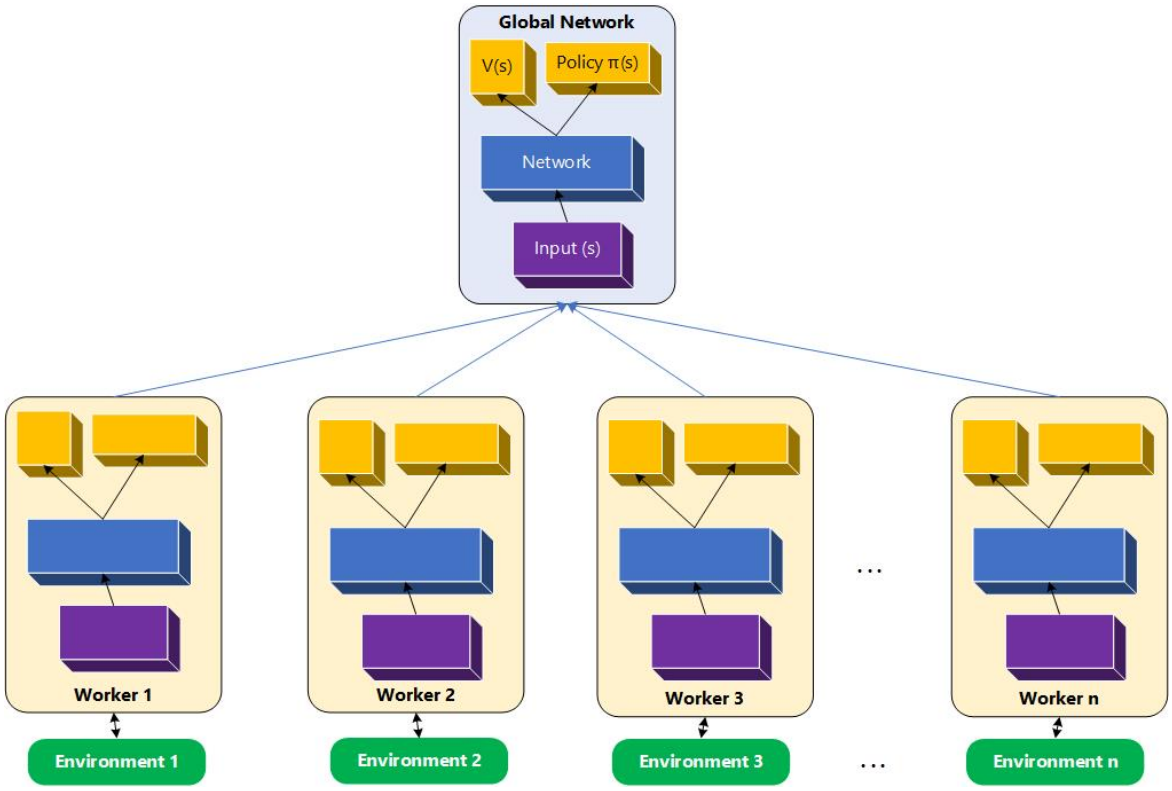

Figure 2.14: Asynchronous Advantage Actor Critic (A3C)

2.4.7 Empowering Edge Applications With Deep Learning

In recent years, edge computing and deep learning have experienced fast progress and considerable success in their respective sectors. The massive amount of valuable data generated and collected at the edge, on the other hand, necessitates more robust and intelligent processing capabilities on the local level to fully unleash the underlying potentials of big data and meet the ever-increasing demands of various applications. Recent advances in deep learning have thrown light on edge application scenarios, demonstrating extraordinary abilities in information interpretation, data management, decision-making, and other areas. The merger of these two technologies has the potential to open up many more doors, enabling the creation of a slew of new applications. Edge computing has already been steadily combined with artificial intelligence (AI) to achieve edge intelligence.

The combination of edge computing with deep learning allows computing intelligence to penetrate every nook and cranny of a city, resulting in a smart city that can deliver more efficient, cost-effective, energy-saving, and convenient services. Computer vision and DRM have been used in smart cities for public safety, is to identify the slums and traffic rate prediction [118] [119]. Similarly, diverse machine-learning approaches have been used for building efficient communication on the Internet of Vehicles (IoV) [120] and allowing more intelligent transportation management, such as autonomous driving, traffic prediction, and traffic signal control. Likewise, there are several approaches toward enabling the intelligent smart building, which helps to minimize energy consumption, increase building security, and improve building sensing capacity [121]. Additionally, recent trends in smart manufacturing rely on deep learning's intelligent processing to offer manufacturing inspection and monitoring [122] and reduce equipment's failure by providing near real-time diagnostic analytics [123].

2.5 Overview of Blockchain for Trust IoT Management

2.5.1 Introduction to Blockchain

Blockchain has emerged from the current Bitcoin [124] cryptocurrency to become an appealing technology for decentralized resource management and transactions lookup. Beyond cryptocurrency use, Blockchain is a decentralized ledger that stores records and transactions carried out between users since their setup, where each member can check the validity of the chain by itself. A distributed ledger is a shared database, fully replicated at multiple nodes or sites, without any centralized third-party authority. Updating the database needs the agreements of most distributed nodes that use a

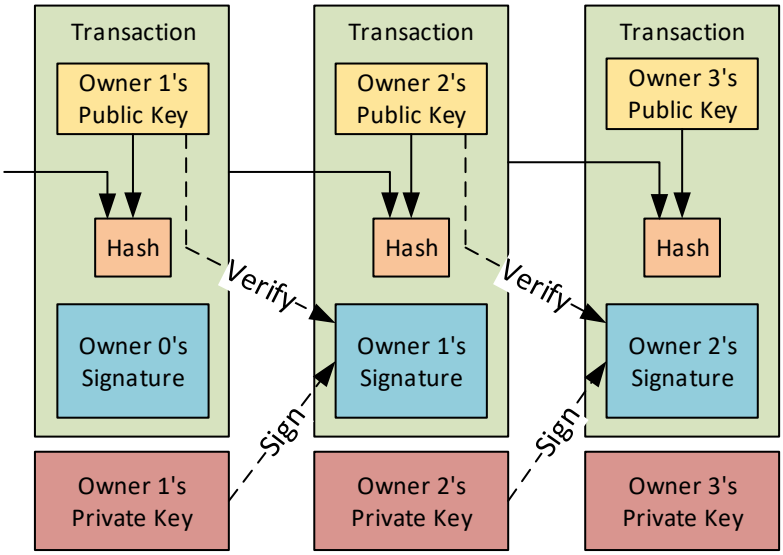

Figure 2.15: The Blockchain Data Layout

specific decision protocol.

Figure 2.15 illustrates the blockchain data layout, where transactions are represented by addresses, private and public keys, data, and hash values. Transaction data store user records, while hash values store coded, and encrypted information generated from the previous block. The first block does not have a hash value to any previous block, called the genesis block. Instead, the blocks contain data in a secure form, e.g., timestamps and hash tree (i.e., Merkle tree) to store data blocks and hash the previous transactions. Additionally, a blockchain can be seen as a distributed database that maintains a doubly linked list of ordered blocks. Each block averages *1 megabyte* and contains control data of approximately *200 bytes*, such as a timestamp, a link to a previous block, some other fields, and *1* to N transactions as can fit in the remaining space. The blocks, once recorded, are designed to be resistant to modification; the data in a block cannot be altered retroactively [125]. Using a peer-to-peer network and a distributed timestamping server, a public blockchain database is managed autonomously. The block validation system is designed to be immutable so that all transactions, old and new, are preserved forever with no ability to delete.

The adoption of blockchain technology has increased in recent years in many financial [126] and government [127] sectors, which moved from non-trusted transactions towards decentralized, cryptographically secure, and immutable ecosystems without any centralized authority. As a result, a blockchain-enabled IoT implementation can improve the application's overall speed by allowing devices to record, validate, and then activate transactions [128].

The following are the basics of blockchain technology as explained by Zheng et al. [129] and Tasca and Tessone [130]:

- **Decentralization:** A supervisor is usually present in all standard transaction

systems, validating the transaction's compliance and recording it in the system. Because such a central unit is responsible for processing and approving all requests, it is frequently a bottleneck that affects the overall system's efficiency. The decentralized structure of the blockchain system, which is based on a consensus algorithm, eliminates the need for a problematic central supervisor because each transaction is mutually confirmed, ensuring data consistency.

- **Persistency:** Because the transactions are verified, any attempt to approve transactions that are incompatible with the specified policies are promptly discovered by confirming/mining nodes, as are blocks containing inaccurate data.
- **Anonymity:** Each network user is given a randomly generated address (hash) with which they may execute actions. This address does not allow for the unambiguous identification of a genuine user.
- **Auditability:** Because each transaction must relate to a prior transaction, it is possible to trace and verify what occurred to the processed data. For example, one may see how a certain user's balance has evolved since the system's inception on the bitcoin network.
- **Transparency :** Every user who can access the blockchain can view all transactions (i.e., an encrypted format of the transactions) from any public address; everyone on the public network has the same rights.
- **Security :** Due to one-way cryptographic hash algorithms, the chain of blocks is shared, tamper-proof, and cannot be faked. The employment of cryptographic technologies ensures the security of transactions. Users can only submit data if they have a private key. The private key is used to create a signature, certifying that a specific user requested the transaction and preventing it from being altered.
- **Immutability :** Blockchain data is immutable. That is to say. The network must approve each ledger entry. Therefore, it can't be an undercover operation. Each block includes the preceding block's hash, which is calculated using the contents of the block. As a result, even a slight change in the data causes a change in the hash, resulting in the alteration being intercepted and rejected by the other nodes.

2.5.1.1 Blockchain P2P Properties

Blockchain infrastructure is inherently decentralized so that all nodes are distributed in an authenticated peer-to-peer (P2P) network. Furthermore, all communication is direct, and synchronization is realized through a distributed timestamping server so

that all public distributed ledgers are managed autonomously without any intermediate entity. This P2P architecture helps distributed ledgers achieve large-scale and systematic cooperation in an entirely distributed and decentralized manner. Furthermore, blockchain use Interplanetary File System (IPFS) [131], which is a P2P distributed file system used to connect all blockchain nodes with the same system of files. IPFS is also used for storing and accessing files, finding information by its contents (i.e., content addressing, not location addressing), and distributing and versioning extensive data on many computers.

IPFS implements a distributed hash table (DHT) that offers a lookup service similar to the hash table (key, value) pairs [132]. All participating nodes in the blockchain are responsible for maintaining the mapping of keys to values so that any change in a single blockchain node cannot disrupt all of the networks. Such a property allows the blockchain network to scale up and down while continue handling the arrival of new joining nodes, the departure of other nodes, or even the failure of some other nodes [133].

2.5.1.2 Trust with Consensus Algorithms

Blockchains are secure by design, and they use a consensus protocol to prevent a single node or entity from controlling the whole blockchain network or distorting the truth of what should be stored in the distributed ledger [134]. Furthermore, the consensus algorithm is a fault-tolerant mechanism that is used to achieve the necessary agreement to ensure that a failure of one or more blockchain nodes will not alter or cause the failure of the entire network [135]. Additionally, the blockchain uses the consensus protocol to keep the network consistent by keeping decentralized records like a centralized database.

2.5.1.2.1 Proof of Work (PoW) The PoW consensus is a technique initially employed in bitcoin. The main idea behind this algorithm is to solve a complicated mathematical puzzle quickly and efficiently. Because this mathematical challenge necessitates a lot of computer power, the node that solves it first gets to mine the next block. Indeed, this technique makes the nodes in competition compete to produce a hash value for the block header that is equal to or less than a goal value [129]. When a node reaches the goal value, it broadcasts the block to other nodes, and the other nodes must mutually validate that the hash value is accurate. Other miners will attach this new block to their blockchain if the block is verified.

2.5.1.2.2 Proof of Stack (PoS) Instead of investing in expensive technology to solve a complicated issue like PoW consensus, the PoS makes validators invest in the system's currency by locking up part of their coins as a stake in this form of a consensus

process. Only those are allowed to participate. All the validators will then validate the blocks. Validators will validate blocks by betting on them if they find one that they believe can be added to the chain. Validators receive a return proportionate to their wagers based on the actual blocks uploaded to the Blockchain, and their stake increases proportionately. Thus, depending on their economic stake in the network, a validator is picked to produce a new block. As a result, PoS encourages validators to establish a consensus through an incentive mechanism.

2.5.1.2.3 Proof of Authority PoA technique was first suggested as part of the Ethereum ecosystem for private networks and was integrated into the Aura and Clique clients. The authorities are a group of N trustworthy nodes that PoA algorithms rely on. Each authority is assigned a unique id, and most of them are presumed to be trustworthy. This principle leverages the value of identities, meaning block validators are not staking coins but their reputation instead. To order the transactions issued by clients, the authorities conduct a consensus. The mining rotation schema, a commonly used way to spread the burden of block creation across the authority, is used to achieve consensus in PoA algorithms. Time is split into stages, each with a mining leader elected by the people.

2.5.1.2.4 Proof of Elapsed Time PoET is one of the most equitable consensus algorithms commonly employed In permissioned Blockchain networks, choosing the next block only based on fairness. Every validator on the network has a fair chance to produce their block using this process. All nodes achieve this by waiting for an arbitrary period of time and then adding proof of their waiting time to the block. The barriers that have been generated are broadcast to the network for others to examine. The validator with the lowest timer value in the proof portion wins. The winning validator node's block is added to the Blockchain. Additional checks are built into the algorithm to prevent nodes from always winning the election or providing the lowest timer value.

2.5.1.2.5 Practical Byzantine Fault Tolerance (PBFT) PBFT was created to function well in asynchronous systems with no time limit for receiving a response to a request. In addition, it has been designed to have a minimal amount of overhead time. Finally, its purpose was to address many issues with existing Byzantine Fault Tolerance (BFT) solutions. BFT (Byzantine Fault Tolerance) was derived from Byzantine Generals' Problem. A distributed network characteristic allows it to establish consensus (agreement on the same value) even when some nodes in the network fail to reply or respond with inaccurate information. The goal of a BFT mechanism is to protect against system failures by utilizing collective decision-making (both accurate and faulty nodes), intending to reduce the effect of faulty nodes. The Byzantine

Generals' Problem inspired BFT.

2.5.1.2.6 Proof of Burn (PoB) Instead of investing in expensive hardware, validators in PoB 'burn' coins by sending them to an address where they are permanently lost. Validators acquire the right to mine on the system based on a random selection procedure by committing the coins to an unreachable address. As a result, validators have a long-term commitment in return for a short-term loss when they burn tokens. Miners may burn the native money of the Blockchain application or the currency of an alternate chain, such as bitcoin, depending on how the PoB is implemented. The more coins they burn, the more likely they will be chosen to mine the next block.

2.5.1.2.7 Proof of Capacity With PoC consensus, the validators are meant to contribute their hard drive space instead than investing in expensive gear or burning coins in the Proof of Capacity consensus. As a result, validators with greater hard drive space have a better chance of being chosen to mine the next block and winning the block reward.

2.5.1.3 Smart Contracts

A smart contract is a piece of executable code stored on a blockchain executed in response to a particular set of circumstances. Smart contracts cannot be performed until their calling transactions are included in a new block. Transactions are structured into blocks to eliminate non-determinism, which may otherwise affect the output results. IoT devices gain autonomy from blockchain contracts by directly assessing if agreements meet contractual criteria. As a result, a smart contract can reduce regulatory costs, while blockchain functionalities serve as a ledger form, validating that transactions were completed.

Business logic is executed automatically using smart contracts in a blockchain-based IoT system without exposing the underlying mechanism to dangers like denial-of-service assaults. When processing transactions and connections, a smart contract provides a high degree of collaboration and cohesiveness. As a result, a smart contract allows the ledger's service to contain the language of the transaction's terms as well as the computations to determine whether those criteria have been met [20].

As blockchain ledgers are still emerging, smart contracts take diverse forms depending on the use case where they will be deployed [136] such as financial, notary, Game, wallet, library, etc. Smart contracts can also pave the way toward new business models and facilitate resource leading of diverse IoT services [137].

2.5.1.4 Decentralized Applications (DApps)

Current blockchain-based applications are still confined to using smart contracts for essential data and functionality that must be resistant to changes. To complete the application, smart contract users must still execute their apps locally. One of the main reasons is the performance limitations of current blockchain systems, which cannot suit many applications' needs. This raises concerns about operational security and application upkeep. For example, deliberate cheating behaviors may be hidden from the public audit in local components. To this end, ideal blockchain applications use decentralized applications (DApps) that are entirely hosted by a peer-to-peer blockchain network. A deployed DApps require a low level of maintenance and control from the original developers. Smart contracts enable DApps to interact with third-party applications on a smartphone, remote servers, hosted on the cloud, or even embedded into small IoT devices. DApps can also access IPFS through secured TLS (Transport Layer Security) endpoints and exchange information with them using JSON-RPC API.

Indeed, blockchain DApps are like any other modern web application (but are P2P supported) that typically consists of a user interface (UI) that interacts with backend services, e.g., reading and writing to persistent storage, processing, complex logic, etc. The backend services use the blockchain core functionalities and libraries, such as the Ethereum platform and diverse smart contracts technologies deployed in it. DApps interact with the other functionalities of the Blockchain, allowing for transparent, verifiable, and immutable records of each transaction, as opposed to the client-server architecture that drives conventional internet apps. In addition, DApps can have their own set of smart contracts that encapsulate business logic and provide persistent storage of state when implemented on blockchain networks.

According to the description of DApps in Raval et al. [138], typical DApps include the following four characteristics:

1. *Open Source:* Due to the trustworthiness of blockchain, DApps must make their code open source so that third-party audits may be performed. For example, Web3 is a collection of libraries that contain functionality for the Ethereum blockchain [139], which can be used to allow DApps to interact with a local or remote Ethereum node using HTTP, IPC, or WebSocket. In addition, web3 can be used as a developer-facing programming framework [140] to build cross-chain DApps.

2. *Internal Cryptocurrency Support:* Internal currency is the engine that drives a DApps's ecosystem. DApps can use tokens to quantify all credits and transactions between system users, including content suppliers and consumers.

3. *Decentralized Consensus:* The cornerstone of transparency is decentralized consensus among nodes.

4. *No single point of failure:* Because all components of the apps will be housed and operated on the blockchain network rather than on a single computer, a completely decentralized system should have no single point of failure.

2.5.1.5 Types of Blockchains

Even though Blockchain is a relatively new technology, there are a few alternative Blockchain taxonomies identified in the literature [141] [142] [143] [144] [145] [146], such as public and private Blockchains. The distinction between public and private Blockchains is not based on a difference between public (states, local governments, etc.) and private (companies, NGOs, etc.) Blockchains, but on the open or closed nature of the Blockchain protocol, because Blockchain protocols can be distinguished by whether they allow free writing and reading or whether one or both of these operations must be accepted by a third party.

According to Lin and Liao [147] blockchain technology can be classified into three categories based on the availability of data as follows:

2.5.1.5.1 Public Blockchains are all those whose transaction records may be accessed by everyone on the earth at any time and can participate in the process of obtaining a consensus. Indeed, the transactions stored in these ledgers are not adjustable, unerasable, accessible for reading by all, ultra-secure based on cryptography, without a central governing body, and decentralized. This is also known as a permissionless Blockchain. Bitcoin [124] and Ethereum [148] are two examples of such a network.

2.5.1.5.2 Private Blockchain Companies can install these solutions in their own internal environments, and they can choose to employ them in a completely centralized governance model (permissioned). The implementation occurs within a group of partners who have decided to manage a business process jointly; they may know each other, but they don't always trust each other.

2.5.1.5.3 Consortium Blockchain where each member has a network node in its infrastructure, and a sufficient number of members must validate transactions for a block to be added to the chain. At this level, consensus no longer works because of the remuneration of one of the nodes according to the computing power involved, but only based on a majority acquired within the members on the validity of the transactions. Beyond the consensus, the operating mode is the same as for public Blockchains, particularly governance.

2.5.1.5.4 Comparing blockchains Table 2.1 compares between the aforementioned blockchain categories. It is worth noting that each category f blockchain has its unique set of features. For public blockchains, the advantages are that the communication is organized in a P2P fashion without any intermediate node between participants, and the blockchain is considered pseudo-anonymous. However, public blockchains still suffer from scalability, lower transaction speed, and higher energy consumption. Conversely, a private blockchain is the solution of choice to develop and execute its own blockchain company. Nevertheless, a private blockchain is less secured and decentralized than a public one. On the other hand, the consortium blockchain is likely to appeal to businesses and organizations looking to simplify communication and organizational collaboration, e.g., banking, financial, insurance, etc. It offers scalability and is much more secure and can be seen as more efficient when compared to public blockchains since it has better customizability and control over resources. Nonetheless, it is less transparent and less anonymous compared to other blockchains.

Table 2.1: Comparison of types of Blockchain

	Features				
	Accessibility	**Participants**	**Consensus**	**Transaction Speed**	**Decentralization**
Public	Anyone	Permissionless Anyone	PoS/PoW	Slow	Completed Decentralized
Private	Single organization	Permissioned known identities	Voting multi-party consensus	Lighter and Faster	Less Decentralized
Consortium	Multiple selected organizations	Permissioned known identities	Voting multi-party consensus	Lighter and Faster	Less Decentralized

According to the business requirements and the advantages that each blockchain category has to offer, the choice of which category fits better for given business logic is still under debate. Before deciding on the best blockchain for use, it is essential to reconsider each use case's features, advantages, usage, and requirements.

2.5.1.6 Blockchain Practical IoT Use Cases

2.5.1.6.1 Smart Cities As a major pillar of the smart city ecosystem, human behavior, technology nodes, and institutional administration are integrated into a single service architecture. This model specifies essential characteristics of successful BC data management to satisfy the system's service delivery requirements: automated data collection, distributed data security, openness and privacy, trust-free governance, and democratization [149].

The Blockchain solution, according to Zheng et al. [150] is a technology-supported intervention capable of scalable, secure, and efficient data management across cloud-based or decentralized network channels that can be coordinated to provide a robust output of informational resources to meet this wide range of expectations.

Academics are being pushed to establish and apply a more productive, efficient security standard based on a centralized blockchain solution by the immediate ram-

ifications of IoT security issues. First, efficiency and system performance are proved through small-scale experiments, according to Chen et al. [151], and then gradually scaled up to meet the needs of large-scale systems. A blockchain-based energy grid was developed in Brooklyn to allow solar-paneled houses to track their energy output and consumption, making system credits and debits easier [152]. While comparable solutions have been proposed for other service-level billing possibilities, such as healthcare, establishing a decentralized, intermediary-free pricing system would eventually create the efficiencies required to reduce network costs. By aggregating data on healthcare spending, Kundu [153] predicts that insurance companies and service providers would be able to connect with client data, monitoring demand and giving discounts based on health, payment performance, and network engagement (e.g., visiting their primary care provider). Similarly, the consolidation of data management services via centralized cloud-based, network-routed authorizations ensures seamless integration as consumers add additional layers of technology and information resources to their network connections in the integrated IoT solution developed by Bruneo et al. [154].

2.5.1.6.2 Supply chain Modern supply chains have grown into complicated networks due to recent advances. As a result, supply chain management systems confront several difficulties. These include a lack of visibility from the upstream party (Provider) to the downstream party (Client); a lack of flexibility in the face of rapid swings in demand and cost control; a lack of reliance on safety stakeholders; and poor supply chain risk management. Blockchain is employed throughout the supply chain to meet the rising demand for goods.

Many of these prerequisites for a successful and efficient supply chain are met by default by the Blockchain data structure. Thus, it is a natural choice for industries and their financial partners to use it to manage their supply chains [155].

Tracing the origins of a product is especially important for the quality management of sensitive products, such as food or medicine. The timestamped registration of all information regarding the production, shipment, and sales of any particular product on a blockchain would instantly identify the roots of such cases instantly [156].

The technology's distributed nature is one of the features that make blockchains attractive for supply chain implementations. There is no central organization managing the transactions and storing the supply chain activities. The fact that all transactions and information are stored on all network nodes also inspires trust and security.

The traceability property is a direct consequence of the fact that it cannot be altered once information is registered on the chain. Of course, perfect immutability does not exist since. Theoretically, most networks can coordinate to tamper with the chain. However, this is practically infeasible.

2.5.1.6.3 Real Estates The traditional real estate system entails a great deal of risk and time commitment. It also goes through many levels of legal procedure, requiring numerous paper signatures and manual document verification. Blockchain can help address these challenging issues by offering a decentralized for the purchase and sale of properties without the involvement of a third party. In addition, files and documents can be digitally checked and authenticated, and all papers can be kept in a distributed digital ledger database accessible to everyone involved. smart contracts can play a significantly larger role in this market [157] to connect buyers and sellers more quickly and directly. Moreover, tokenizing real property can facilitate digital exchange and eliminate the risk of manipulation by third parties.

Additionally, key real estate transactions such as purchasing, selling, finance, leasing, and management can be transformed by blockchain technology. Blockchain can help the real estate market overcome inefficiencies and inconsistencies, improve transparency and reduce the risk of fraud [158]. For example, Karamitsos et al. [159] created a blockchain system for real estate to increase trust among entities involved in real estate development, and it eliminates the need for intermediaries because transactions are independently verified and validated automatically.

2.5.1.6.4 Identity Management Identity management solutions are commonly used in real-world applications and make managing digital identities and activities like authentication easier. Recently, there have been attempts to offer blockchain-based identity management solutions that allow users to take control of their own identities. For example, Tobin et al. [160] created the Sovrin system based on Hyperledger to employ digital credentials in the real world. Sovrin has a self-contained identity independent of any centralized authority and cannot be destroyed. Similarly, Naik et al. [161] created uPort, a self-sovereign identity system. Based on Ethereum, the uPort identity is defined by the Ethereum account address with which users communicate, and the identity is permanent. The uPort table is the smart contract that underpins all uPort identities and serves as the foundation for authentication and sharing offline data access. uPort enhances Ethereum-based apps from the user's perspective, allowing them to engage with actual people rather than hexadecimal addresses. Likewise, Soltani [162] designed ShoCard, a blockchain-based identity management solution that allows individuals to store and defend their own digital identities. The user's identification data will always be combined with the user's key to preserving privacy. A third-party database is no longer necessary. ShoCard stores the user data authentication code on the blockchain, ensuring personal identification authenticity and facilitating third-party verification. ShoCard also accepts payments with National Financial System (SFN) currencies.

2.5.1.7 Blockchain Implementations

2.5.1.7.1 Bitcoin is a solution demonstrated by Satoshi Nakamoto to address the double-spending problem [124] particularly. Nakamoto developed digital signatures for bitcoin cryptocurrency and a method that uses a P2P distributed timestamp server as a generator of the computational evidence of transaction chronological ordering. The project is maintained by an open community [163]. Each bitcoin transaction is defined as a combination of the previous transaction's digitally signed hash and the next owner's public key. The transaction is signed using the private key, and the transaction is verified with the public key. The public key is stored in a wallet, i.e., a piece of software or hardware or cloud-hosted application. The Bitcoin ledger is a state transition system that consists of a state that displays the ownership status of all existing bitcoins and a state transition function, which takes the form of a transaction. A new state is the result of the state transition function [164]. If the sender has enough bitcoins to conduct a transaction, this procedure produces state changes for the sender and receiver; otherwise, it produces an error.

2.5.1.7.2 Ethereum The Ethereum public blockchain is a distributed computing platform that emphasizes the use of smart contracts. Engineers may quickly develop decentralized apps at a high level and make use of the distributions provided by Blockchain technology [165]. Compared to Bitcoin, which only maintains Bitcoin transactions between addresses, the Ethereum blockchain contains addresses with EVM codes. Ethereum's transactions are recorded on the blockchain and contain data about the information passed to the program as input. These transactions are translated by the Ethereum Virtual Machine (EVM) and stated in the appropriate language. Ethereum is based on a Turing complete machine that makes the smart contract more interactive to understand and implement any future agreement. We provide more details about the Ethereum blockchain in Section 2.5.2 below.

2.5.1.7.3 Hyperledger is an open source blockchain project jointly developed by the Linux Foundation and IBM to promote the building of distributed blockchain ledgers by everyone. Hyperledger is a consortium blockchain. The goal is to foster cross-industry collaboration by creating blockchains and distributed leads, to enhance system efficiency and dependability [166]. Five Hyperledger projects have developed, including Indy, Factory, Iroha, Burrow, and Sawtooth [167]. Instead of the smart contract, Hyperledger introduced a *chaincode*, which is the program written in Go, node.js, or Java, to implement different users and applications' interfaces and handle business logic agreed to by members of the network.

2.5.1.7.4 Corda is a distributed ledger platform comprised of mutually distrusting nodes that provides a single worldwide database to track the status of transactions between organizations and individuals [168]. Corda distributed blockchain nodes deployed over a P2P network, but the communication that happens only between nodes is point-to-point, where data are encrypted using transport-layer security. As a result, there is no global broadcast of transactions to all parties on the network in the Corda blockchain. Instead, all nodes can send messages to all other nodes on the one-to-one communication. Furthermore, for any node in Corda to join the network, it should obtain a certificate from the blockchain operator that identifies the node's identity through its public key. Such a deployment model enforces rules that stipulate the information each node is allowed to send and/or receive. In addition, Corda offers a mapping service that matches each node identity to an IP address so that each node uses this address to send data to other joining nodes in the Corda blockchain. This mapping service allows nodes to discover and know each other.

Additionally, Corda provides a distributed ledger that is replicated and shared across all nodes so that each node can keep its copy of the ledger. Still, each node has a different view of the ledger depending on the facts they share in the network. This removes a lot of the time-consuming work necessary to maintain all ledgers' synchronization. Moreover, The hash values in the Corda blockchain, which are coupled with the node encryption consensus, are used to ensure that transactions are only viewable to those who are valid participants in the transaction. Corda's significant features include automatic smart contracts and document time stamping to ensure uniqueness.

Nodes that want to share data and transactions in the blockchain should reach a consensus before committing their data to the ledger. Consensus entails gathering existing values, merging them with smart contracts, and establishing new outcomes or states. Transaction validity and uniqueness are the two most essential factors in achieving agreement [169]. The validity of consensus is maintained by verifying the validity of the smart contract codes and ensuring that it was executed with the required signatures. Smart contracts use notarization, time stamping, and other requirements to maintain the transaction's uniqueness. Further, Corda uses the notion of an immutable state, which consists of digitally signed secure transactions. Transactions are validated through the consensus protocol, which is executed inside a sandbox environment to enforce the system's security. In addition, the consensus protocol invokes a verification function to verify if a transaction is digitally signed by all parties, ensuring that a transaction will be conducted if it is confirmed and validated by all participants.

2.5.2 Introduction to Ethereum Technology

2.5.2.1 Ethereum Accounts

The account is Ethereum's fundamental unit. Anyone who wishes to send a transaction to the blockchain must first create an account. Externally Owned Accounts (EOA), through which users may submit transactions directly, and Contract Accounts, which are based on the contract code and can be used to contact another contract and send an inside transaction if necessary. In Ethereum, each account is separated into two parts: a private key and a public key. Each account address is created from the first *20 bytes* of a public key, which is an essential component of every transaction. It's crucial to know the distinction between a transaction and EOA's private key, the transaction's sender, and that following confirmation of the hash value's return, we can monitor all blockchain transactions. For internal transactions, many sources utilize call or message conditions [170].

2.5.2.2 Ethereum Work

Ethereum functions as an open software platform based on blockchain technology. This blockchain is hosted on many computers around the world, making it completely decentralized. Each computer has a copy of the blockchain, and there must be a majority of agreement before any changes can be implemented on the network. However, the Ethereum network also allows developers to build and deploy Decentralized Applications (DApps). These are also stored on the blockchain along with transaction records.

Ethereum functions similarly to bitcoin, with the exception that it provides "Smart contracts" because of its built-in turning language. Transaction fees are required because every published transaction necessitates the cost of verifying and downloading transactions through the network. Ethereum virtual machine code may encode any sort of computation, including loops. Ethereum has built a simplified version of GHOST with five tiers.

2.5.2.3 Decentralized Applications

Decentralized Blockchain applications, or "DApps", are ultimately quite similar to the applications we all know, with a few differences. DApps are linked to blockchain networks such as Ethereum, a copy of the data is stored on each computer in the blockchain network. This means that, by definition, no individual or group controls a DApp.

Ethereum enables you to build centralized apps, also known as decentralized applications. As demonstrated in Figure 2.16, a DApp is made up of a backend code

that operates on a distributed peer-to-peer network. Also, it is software built to run in the Ethereum network without being managed by a centralized system, and this is the major difference against the centralized classic applications: it allows end-users to communicate directly with decentralized application providers.

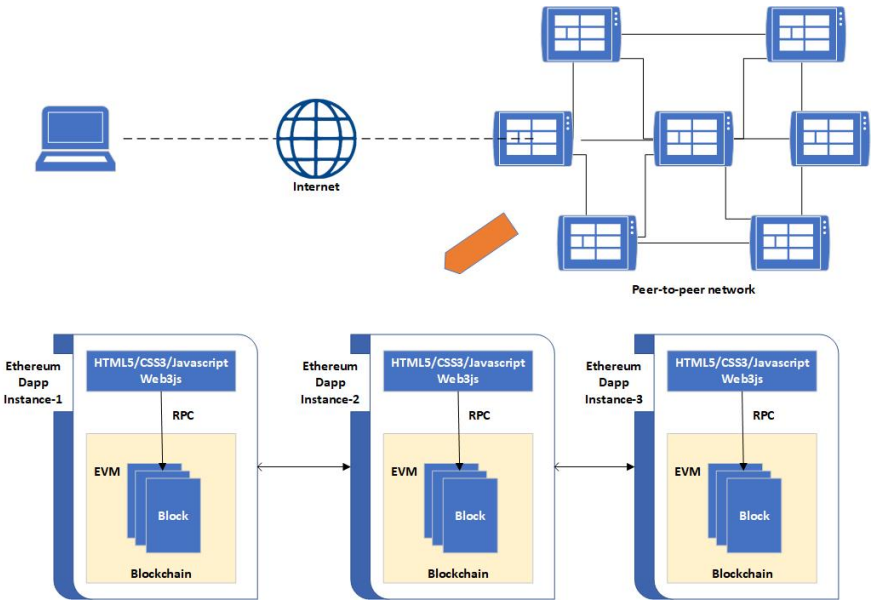

Figure 2.16: Decentralized Application Architecture

2.5.2.4 Solidity

The programming language used to code smart contracts on Ethereum is called Solidity. This language was proposed in August 2014 by Gavin Wood, the co-founder of Ethereum. Developers then developed Solidity Christian Reitwiessner, Alex Beregszaszi and Yoichi Hirai. At its core, solidity is an object-oriented, high-level language for implementing Ethereum smart contracts, which can interact with different programming languages such as C++, Java, Python, and JavaScript. Similar to those traditional languages, solidity is statically typed that can support inheritance as well as complex types and libraries.

Solidity is close to JavaScript and is considered easy to learn by people with existing programming skills. Once the smart contract conditions are coded, they are compiled into OPCODES or operation codes. An operation code is an instruction that specifies the operation to be performed in machine language.

These opcodes are then deployed on the Ethereum network, where the Ethereum Virtual Machine (EVM) executes them. The EVM is the first sub-layer of the network. It is the machine that will interpret the smart contracts and implement them. Similar to Java JVM, in EVM the smart contract code is generated in a form of *byte-code* that can be compiled from source code into low-level code, which in turn will be executed for the software interpreter. The EVM can also support another smart contract language

called Vyper [171], which is a pythonic programming language and strong typing, but has fewer features compared to Solidity.

2.6 Conclusion

This chapter describes state of the art in IoT and edge computing. We discussed how SDN and NFV could be used to enable massively distributed 5G IoT networks. Additionally, we highlighted the difficulty of task offloading and resource allocation via reinforcement learning techniques. Besides, we introduced the fundamental concepts behind blockchain technology and delved into its different deployment models and software frameworks used to implement distributed and P2P infrastructure. We also described different cognitive and autonomic control and administration of IoT services and the critical, challenging issues in managing trusted IoT Networks such as scalability, energy efficiency, security, etc. Finally, we relied on research directions on blockchain and the Internet of Things convergence and the empowering of blockchain-based IoT networks with SDN/NFV.

The work described in this thesis investigates different technologies that enable supporting distributed and energy aware IoT systems for network edge communication. The first contribution, which will be described in Chapter 3, develops a novel IoT network virtualization approach based on SDN/NFV that uses a context-awareness model for managing data distribution and energy efficiency in smart buildings' environment. In our second contribution, which will be introduced in chapter 4, we delve into distributed deep learning model-based task offloading and foraging for IoT network edge systems. Our last contribution, which will be discussed in chapter 5, proposes to extend the blockchain system to support secure and decentralized resource allocation in fog computing networks.

Chapter 3

A Context-Awareness Energy-Efficient Framework for SDN-enabled IoT Network

3.1 Introduction

This chapter describes the issues associated with making buildings smart, energy-efficient, and more sustainable realized as part of the IPERCITIES project, which is based on an IoT network's architectural design that uses SDN and NFV. This chapter will focus on the design of a novel IoT network virtualization approach for offering a high level of automation and service chaining.

The objective of this chapter is, therefore, to describe in section 3.2 the implementation details of our IoT architecture in smart buildings, which enables offloading of expensive computation at the network edge. First, we provide a novel approach to realize context information enhanced with Time Series Data Repository (TSDR) approach to model hierarchical structures and relationships for collecting, storing, querying, and maintaining time series data in the SDN controller. Then, we introduce in Section 3.2.1 a new IoT data model that provides a data collection facility to accommodate HVAC sensors and actuators, where data are generated and consumed locally in smart campus buildings. Next, in Section 3.2.3 we delve into the design of our Context-Awareness Model, which we introduce to represent the functional intelligence that identifies a set of contexts, transition rules, dependencies, and relations between contexts, to control energy appliances in smart buildings.

After that, we present in Section 3.3 a prototype implementation to show the proposed architecture framework can be deployed in both educational and residential smart campus buildings using low-cost hardware and lightweight Docker virtualization. To this end, we used an intuitive system that is less computationally intensive, simple to implement on small modular, low-cost Single Board Computers such as the

Raspberry Pi, and capable of online adaptation to changes in ambient temperature and solar heat input through windows, and other factors. Finally, some concluding remarks will be given in Section 3.4 at the end of this chapter.

3.2 Architecture Overview

Figure 3.1 illustrates the architecture of our framework, which is composed of three layers: at the bottom is the *perception layer*, which comprises the sensing layer and the aggregator. The sensing layer encompasses all smart IoT sensors, e.g., temperature, humidity, air quality sensors, and the HVAC actuators, which directly connect to the network via short-range PAN technologies such as RF, Bluetooth, ZigBee, etc. Aggregator sinks collect data from the sensing layer and act as bridges between sensor nodes and IoT gateways, which act as a network proxy with the rest of the network.

IoT sensors and actuators use message queuing clients (MQTT) for publishing and subscribing data to/from IoT gateways. For example, IoT sensors gather temperature readings periodically, listen for network events through MQTT protocol, and send commands to IoT gateways to control fans and HVAC systems. These IoT devices use four MQTT messages to publish-subscribe data:

- i) *Connect* the message allows a client to connect to a remote M2M broker inside IoT gateways. It triggers a callback function to handle any incoming MQTT messages on the subscribed topics;

- ii) *Send* message is used by sensors to publish data to IoT brokers and receive command data back to it; *Store* message allows MQTT traffic incoming to brokers to be stored in a time-series database for retrieval in the future, and

- iii) *Use* message allows using Restful API for client applications to make use of their stored data. Furthermore, these IoT devices make up three QoS levels (i.e., QoS 0, QoS 1, and QoS 2) to create different priority levels for the published data.

Next, we have the *fog layer*, which is located in single-hop proximity of the IoT sensing devices and includes all the network equipment to realize the micro-grid communication infrastructure. It consists of IoT gateways (shown in Figure 3.2) that interface with the perception layer to receive raw data from the sensors and send commands to control the actuators. The gateway concept is prevalent in home ADSL models and WiFi access points. However, the design of the IoT gateway is different since it should be able to integrate heterogeneous smart objects, expose their resources, and make them available to the rest of the IoT network. Additionally, IoT gateways are connected to the SDN network through SDN routers (i.e., virtual and physical),

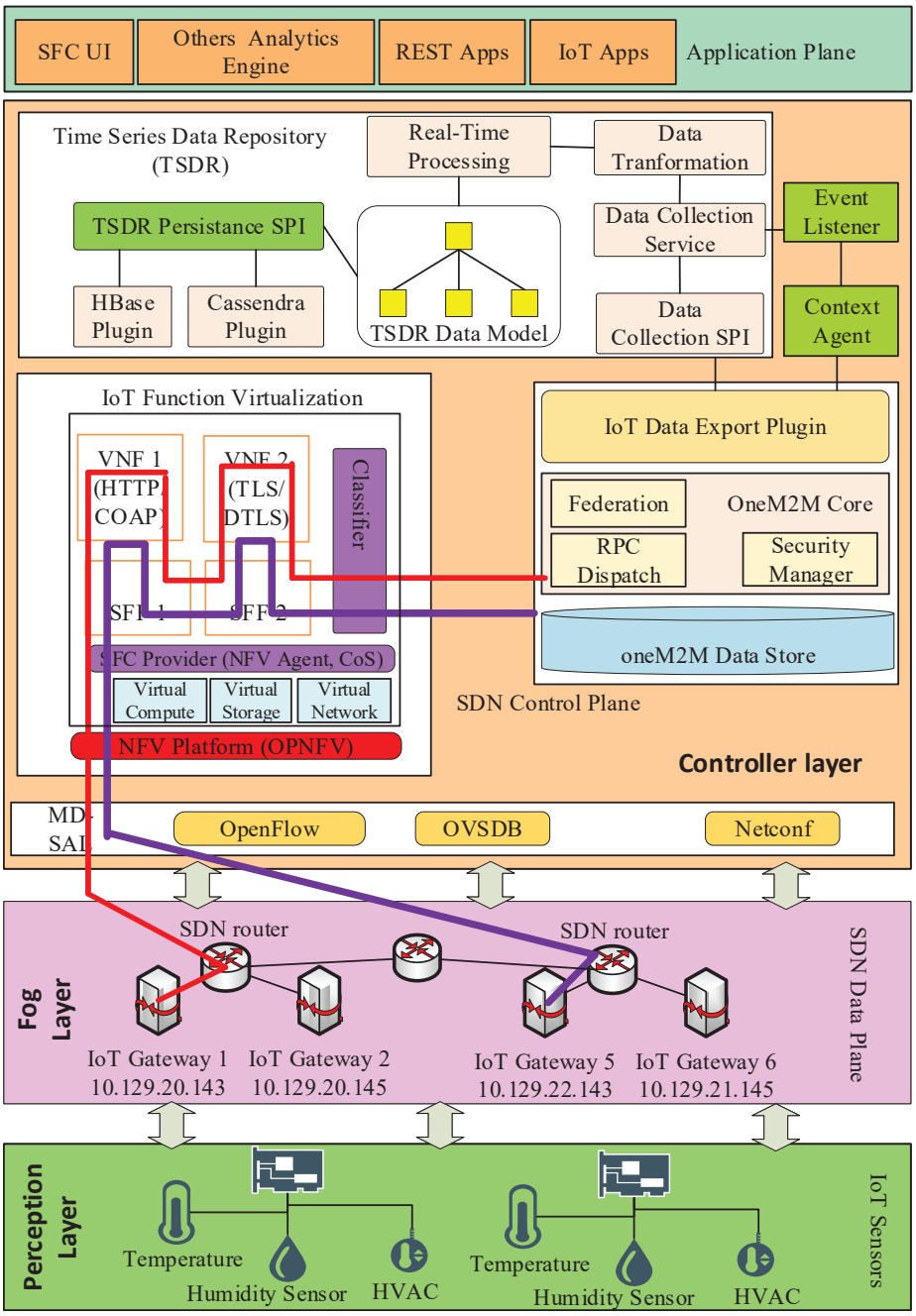

Figure 3.1: Layered Architectural Overview of the SDN-enabled Framework

which embed an OpenFlow agent that can add, remove, update, and delete packets inside these routing devices.

The SDN routers (OpenVSwitch virtual routers in Figure 3.2) are connected to a *SDN control layer* as shown in Figure 3.1, i.e., the SDN controller, which embeds all the intelligence and maintains the network-wide view of the data path elements and links that connect them. The controller contains several modules we develop to integrate into the smart micro-grid network. First, the *IoT function virtualization module* (will be described in Section 3.2.2), which expands the micro-grids network capacity by deploying virtualized IoT functions into software packages that can be assembled and chained whenever required to deliver chained services to IoT devices. Thanks to the

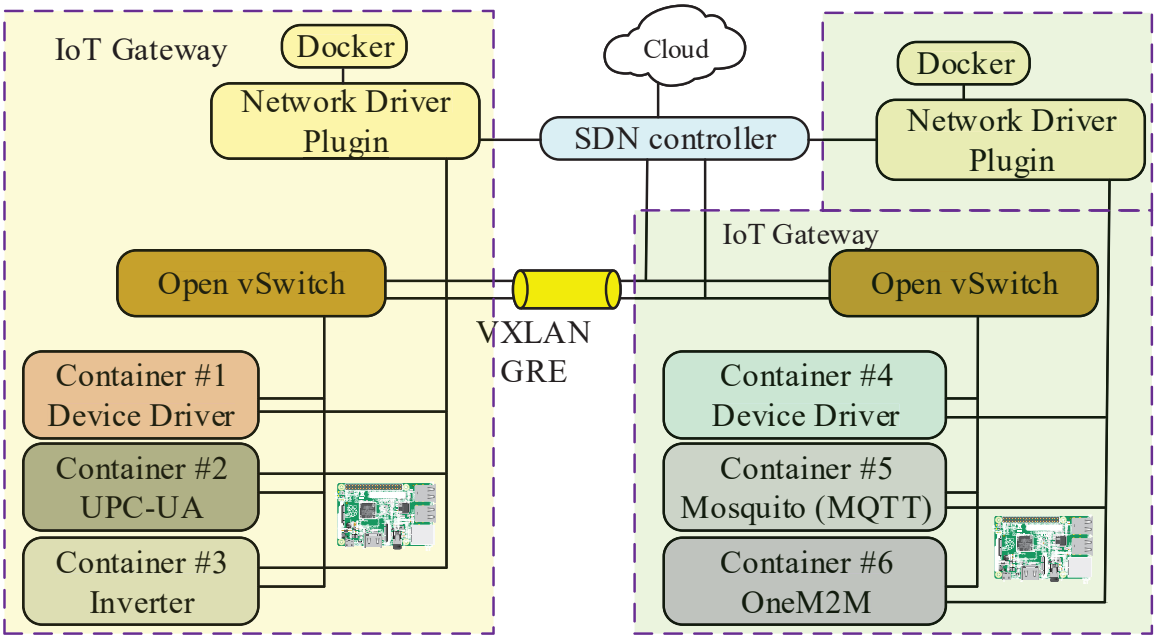

Figure 3.2: IoT Gateways

NFV platform (e.g., OPNFV in Figure 3.1) that encompasses an orchestrator (i.e., SFC provider), which can add new services without interrupting existing ones or upgrading the network with new devices. The orchestrator is the NFV management and network orchestration (MANO) tool responsible for controlling and managing NFV compute, storage, and network resources.

Second, the SDN control layer contains the IoT management model to perform communication with IoT sensors through IoT data management and service capability's module, i.e., *IoT Data Export Plugin* (will be described in Section 3.2.1) for accomplishing M2M operations at scale. It also includes a *context agent* (will be discussed in Section 3.2.3) embeds the context-aware model to perform context reasoning needed for data processing, filtering, and aggregation and to capture the knowledge and generate high-level abstracted context information. Third, the controller layer comprises a data processing module that makes use of a Time Series Data Repository (TSDR) module to perform real-time data processing and analytic, data transformation, and collection services: upon received at the data store, the IoT Data Export Plugin triggers CRUD handling operations to enable writing data in the data-store before connecting them to the context-aware middleware to perform data processing. The TSDR module connects to Cassandra and HBase NoSQL database management system through plugins.

3.2.1 IoT Data Management Model

The IoT data management model is based on a data-centric IoT message broker that uses a standardized oneM2M API to allow authorized sensors and applications to

retrieve the data stored by other devices. In particular, the model implements a hierarchical containment tree where each node in the tree represents n IoT resource. As depicted in Figure 3.3, the tree contains different data and measurements of IoT devices and their associated attributes. Each node in the tree represents a specific resource an IoT device can interact with using the Message Queuing Telemetry Transport (MQTT) broker or direct HTTP-like message exchange. Those attributes provide a resource's description in the form of meta-data that includes information about the resource creation, access rights, resource creator, content size, creation time and date, etc.

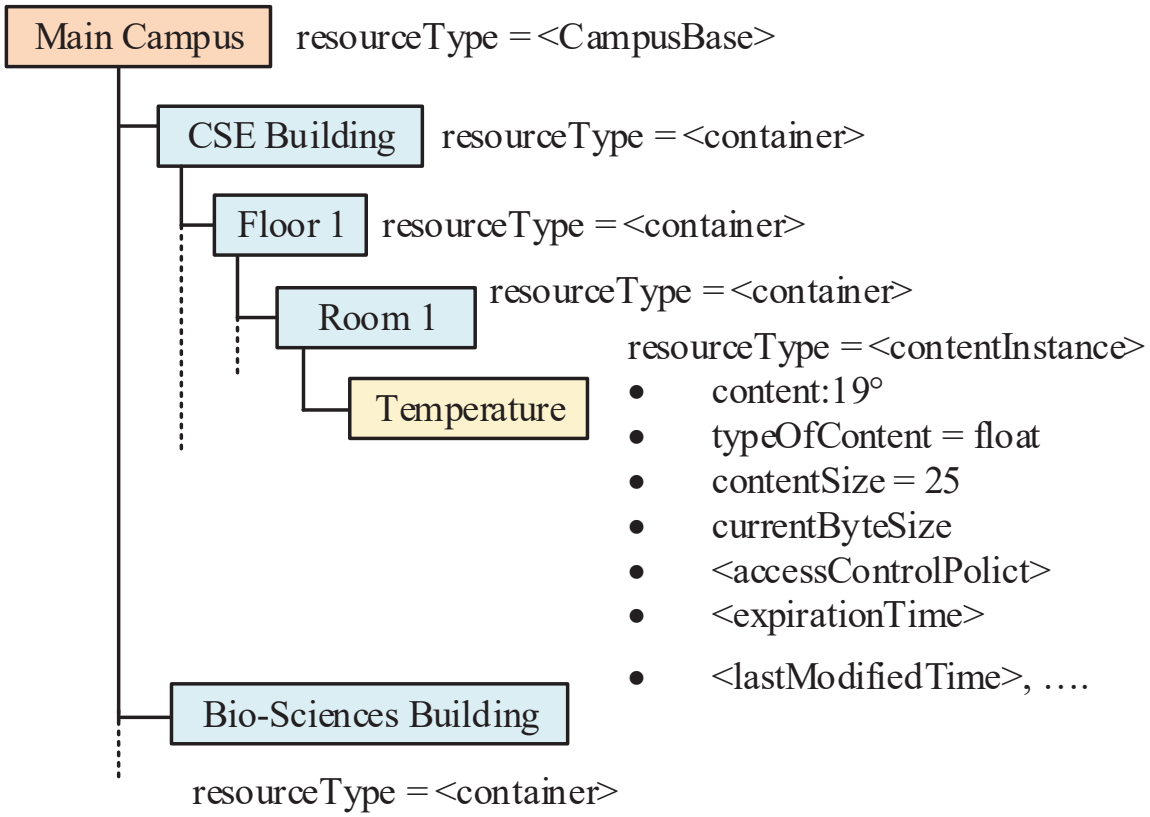

Figure 3.3: Resources tree comprising IoT data.

As shown in Figure 3.1, the oneM2M Data Store represents the back-end database where raw sensor data are stored. Furthermore, the proposed architecture supports CRUD (create, retrieve, update, and delete) operations to collect remote IoT sensors' sensing data. Such an approach allows us to efficiently perform inventory with life-cycle management of IoT devices and perform big data analytics on raw sensor data, retrieving and transforming them into appropriate context representation.

3.2.2 IoT Service Chaining

In Figure 3.1, the *IoT Function Virtualization (IoT NFV)* module shows our proposal for mapping IoT sensors into virtualized functions that follow the ETSI guideline for NFV architectural framework. In addition, the SDN controller is merged with the IoT

management platform (i.e., NFV Platform) to communicate between IoT gateways and their remote sensors and enforce state information security. The IoT-NFV module uses lightweight containers to enable the creation of multiple isolated virtual IoT gateways inside a single physical one. For example, VNF1 and VNF2 in Figure 3.1 represent two independent virtualized functions chained to form a single IoT service. Constrained Application Protocol (CoAP) messages in VNF1 coming from sensors in the sensing layer are chained with DTLS service to enforce the security of IoT resources. Similarly, another group of sensors in the sensing layer that use HTTP/REST services in VNF1 can see their messages chained and secured with TLS in VNF2 to optimize the cooperation between IoT devices and intermediate infrastructure, and the rest of the IoT network. The virtualization layer is based on lightweight containers using Docker. Thus, it becomes fast to create, install, run and deploy independent micro-services and provide simple service composition facilities.

Routing the packets among these VNF components is wholly managed and controlled by the SDN controller. Thanks to Pipework and the "overlay" mode of the OpenVSwitch software router. The former allows connecting multiple containers in arbitrarily complex scenarios. The latter provides a private IP address that is only valid internally. Each IP address P identifies a service deployment in a separate chain so that the SDN controller can program the flow table with the required flow entries F_P to define the following component $B = F_P(A)$ in the chain for which the traffic will be forwarded. The controller creates for each flow entry F_P the forward table entries that match received packets against the forwarding ports A they should follow with P as the destination address.

3.2.3 Context-Awareness Model

The context reasoning model identifies a set of contexts, transition rules, dependencies, and relations between contexts. Figure 3.4 illustrates the context-awareness model which includes a 3-tuple (A_k, T_k, D_k), Where is A_k is the action knowledge, T_k is Transitional knowledge, and D_k declarative knowledge. The action knowledge represents the functional intelligence for a given environment, coded using logic rules or machine learning algorithms. For example, given a current temperature inside a room, the system should switch on or off the HVAC system of the building.

The transition knowledge specifies when a passage to another context should be performed. It can be expressed, for example, as IF (conditions) then (activation) transition rules, fuzzy rules, or neural network probability conditions. Finally, declarative knowledge describes some aspects of the context, including some pre-acquired knowledge for the context. For example, the number of guests in the convocation hall, room size, room usage schedule, etc.

The context reasoning model is used to subdivide the knowledge into multi-level

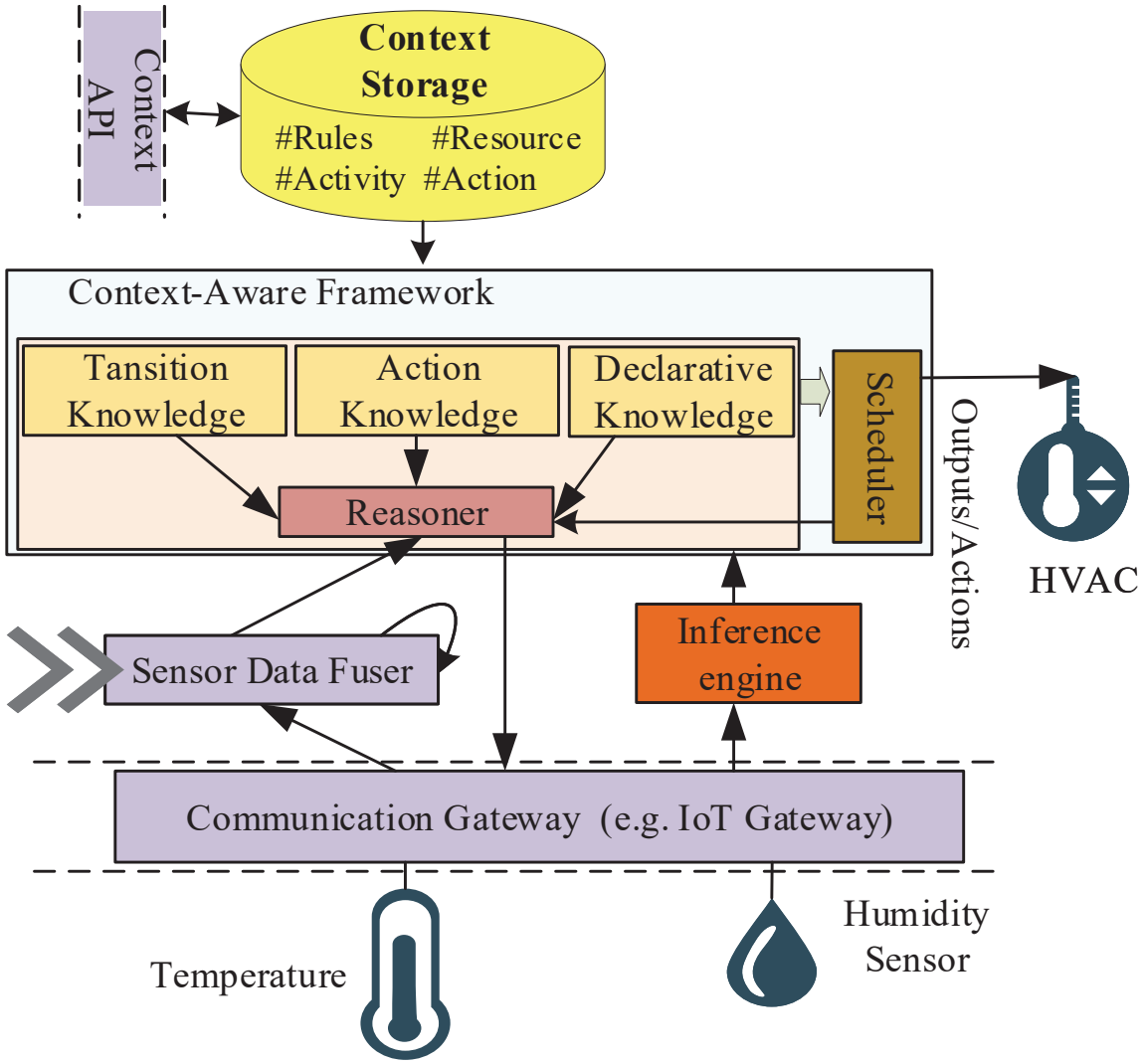

Figure 3.4: Context-Awareness Model

hierarchies. Each level is used to represent a vertical relationship between sub-levels. Figure 3.5 illustrates an example context hierarchy for an educational building. The vertical multi-level hierarchy describes the relationship between groups in a given set $G = \{G_1, G_2, \ldots, G_n\}$, where a given group $G_i \in G$ contains a set of mutually exclusive contexts $C_i = C_0^i, \ldots, C_a^i, \ldots, C_n^i$. An active context C_a^i in a subset of group G_i is active within the context of its parents. That is, to make the inheritance active, transitional and declarative knowledge from selected contexts in groups that are hierarchically above G_i.

In our case, Figure 3.5 illustrates the reasoning model where Campus buildings are the first group G_1. Then, in the second group G_2, we identify different types of buildings, e.g., educational, residential, administrative, laboratory buildings, etc. To apply this context-aware reasoning model to our case, we should activate the set of contexts that best suits a given situation. The active context will control all the execution processes, define behaviors of our actuators, and specify the constraints

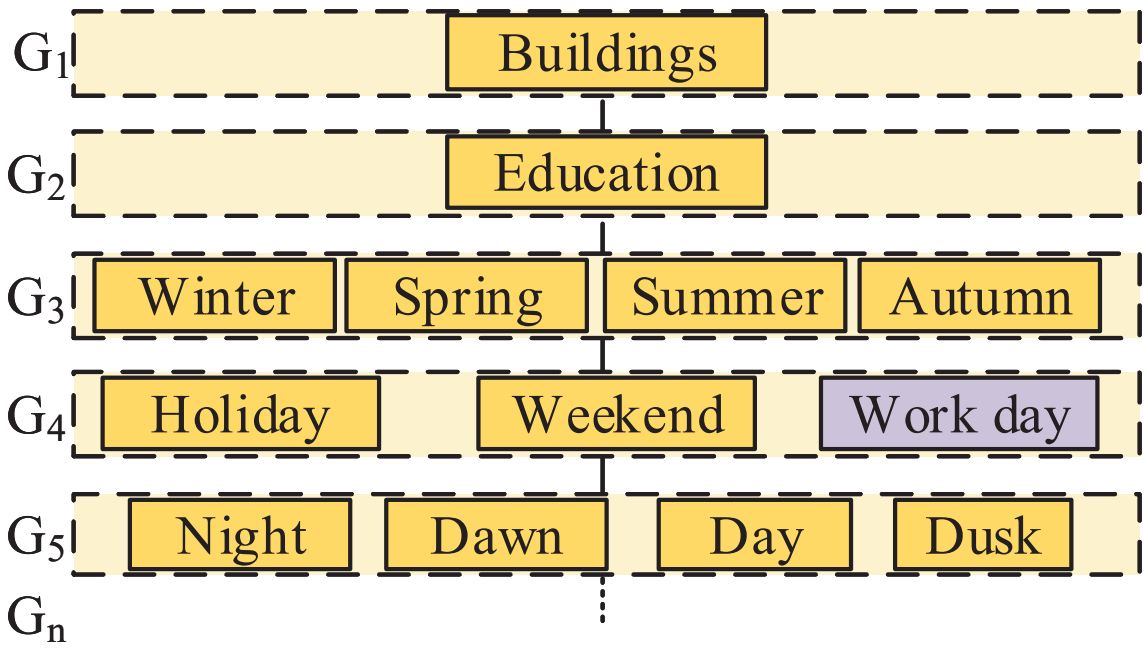

Figure 3.5: Example of context hierarchy levels for an education building

and all other context-dependent characteristics. To seek of simplicity, we describe an example of our context reasoning model from Figure 3.5 as follows:

- C^{t}_{Active} = {Buildings, Educational, Spring, Working-Day, Day}.
- C^{t+1}_{Active} = {Buildings, Residential, Spring, Working-Day, Night}.
- C^{t+2}_{Active} = {Buildings, Educational, Winter, Holiday, Day}.

Specifically, given active contexts C^{t}_{Active} and C^{t+1}_{Active} at certain instances t and $t+1$, it might be seen as easy or even not much complicated to identify the process of managing context operation, in particular given the slow evolution of sensor variables such as temperature, humidity and CO2 level inside rooms in education buildings. However, given different situation that occur in a given group G_i, we identify a domain of services s_i which has its own execution thread(s) and its control is a function of context C^{t}_{Active}. The control of service s_i can be defined by equation 3.1:

$$\texttt{Control of } s_i = \Gamma(C^{t}_{Active}) \tag{3.1}$$

Where Γ is the context reasoning framework operating within s_i. For example, room temperature behaves differently if a door/window is open/closed or if the room is empty or occupied. Electrical lights and fans act accordingly to these situations.

3.2.4 Machine Learning Engine

The machine learning engine capabilities are twofold: first, they help the context reasoning framework to infer the context decisions. Second, the SDN controller's feedback on the environment changes to perform automatic traffic steering and policy placement. For the former, the machine learning engine monitors the current sensor's data delivery, predicts future data, and learns the optimal network management policy. In particular, the energy demand is variable in time and space since its consumption varies qualitatively and quantitatively on the time of days, e.g., working days, holidays, weekends, etc., or the location where IoT devices are deployed, e.g., laboratories, classrooms, office building, etc. The machine learning engine provides a better-personalized experience and prioritizes specific IoT devices that should communicate relevant information to the SDN controller. It also enables storing and processing the collected data to analyze the datasets based on specific parameters such as location, time, and historical data.

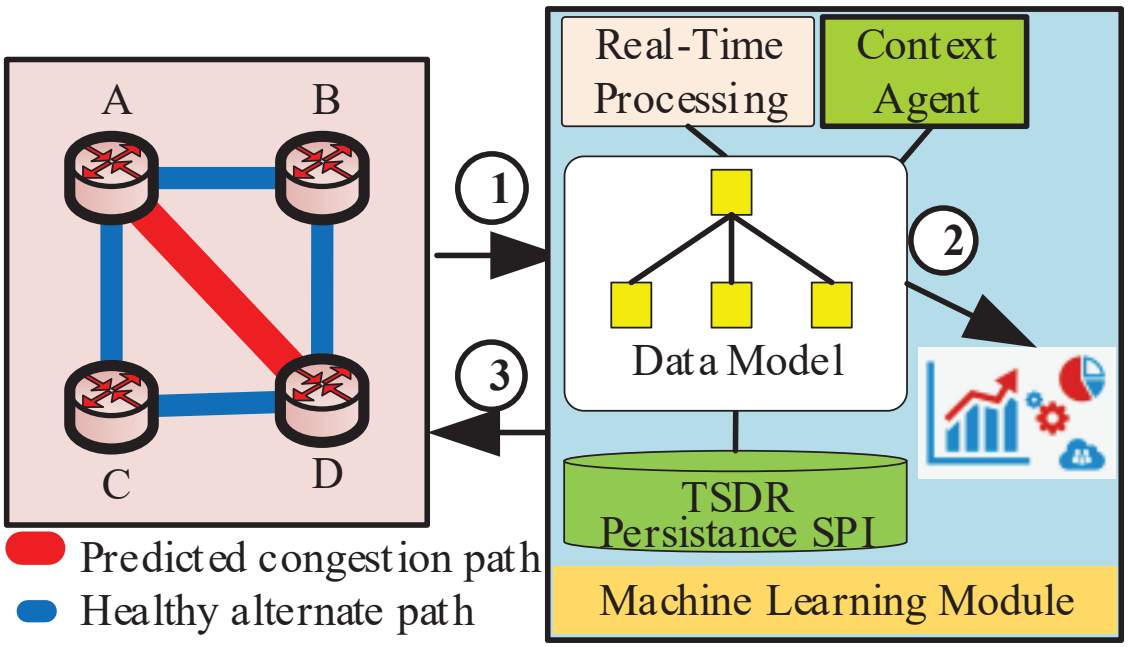

Figure 3.6: Traffic congestion prediction with automated control.

For the latter, Figure 3.6 depicts the machine learning module, where the SDN controller continuously learns from data generated by virtual routers and becomes aware of the runtime status of the network. The controller collects OpenFlow statistics (circle 1), applies ML algorithm, i.e., multi-layer perceptron (circle 2), and takes the right decisions that adjust the policies (i.e., traffic flow redirection from $A \Rightarrow D$ to $A \Rightarrow B \Rightarrow D$ or $A \Rightarrow C \Rightarrow D$) for traffic classification and traffic shaping, dynamically change these policies according to the analytics results, and feedback these results to the forwarding path for automatic steering and policy placement. In addition, the

controller can also use the machine learning capabilities to establish normalized profiles to predict traffic patterns and perform routing optimization based on predetermined or dynamically updated learned rules.

3.3 Proof of Concept Implementation

We have realized a primary prototype implementation to verify the feasibility of running the platform. We first introduce the platform implementation, then highlight two application test cases, and show our approach can operate in actual smart buildings environments such as smart campus buildings and residential buildings.

3.3.1 Testbed Setup

Figure 3.7 depicts our target application. IoT gateways are based on a single-board computer RPi 3 with 1 GB of memory and a quad-core ARMv8 BCM2837B0 Cortex-A53 ARM Cortex-A53 CPU running at 1.2 GHz. The gateway connects to the network using its integrated 2.4GHz 802.11n interface. It also contains *hostapd* userspace daemon software, which we used to create virtual wireless networks inside the same physical one. IoT gateways are equipped with 40 pins GPIO interfaces to connect to a wide range of sensors and actuators.

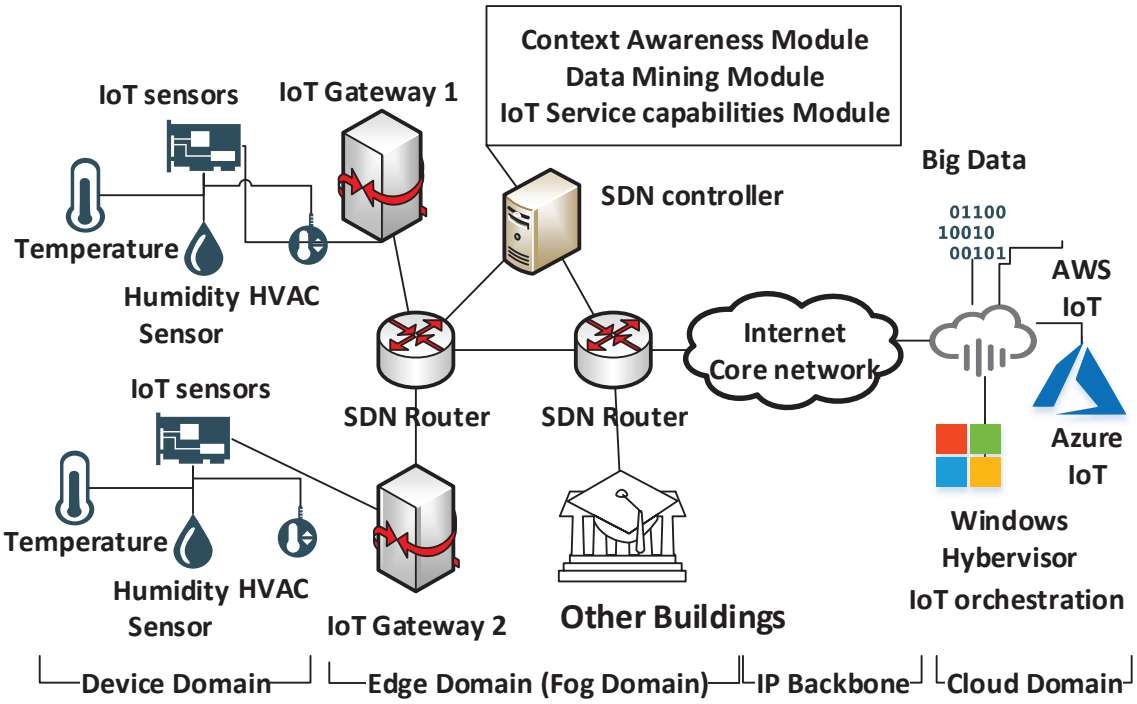

Figure 3.7: Network topology used as the target application

We deployed several sensor boards that act as Cluster Head (CH) nodes to collect raw sensor data and measure the surrounding air quality from all Cluster Members

(CMs). CMs sensor nodes periodically measure temperature, humidity, and Co2 levels and compare them with the last measurement. If results have changed, they send an advertisement message (ADV) to the cluster head. Examples of CMs we used include *DHT22* temperature and humidity sensors; the *MQ-135* Co2-Gas sensor; and the *K30 CO2* module that gathers the level of oxygen inside our campus buildings and student housing residence. We also used a smart energy meter to detect and report power consumption to CH nodes and IoT gateways. As Cluster Heads (CHs), we used multiple NodeMCU IoT platforms running on top of the ESP8266 Wi-Fi SoC, which integrates a TCP/IP protocol stack that allows access to our virtual Wi-Fi network running on IoT gateways. We successfully connected these CHs through the MQTT broker running on IoT gateways. We also used CoAP API to connect some other sensors to the ESP8266 board. COAP servers are deployed inside IoT gateways in the form of lightweight containers.

Additionally, our SDN controller is based on the OpenDayLight (ODL) Project, which has several SDN and application management capabilities. The ODL platform can be configured to run as a highly scaled up and out distributed cluster with IoT, SDN, and NFV functions. It can also be integrated with OpenStack and deployed to communicate with a high-traffic data center. We used OpenVSwitch (OVS) as our distributed virtual multi-layer SDN switch managed by our SDN controller. We used two OVS modes: NAT and bridge. The NAT mode connects the SDN routers with the outside world. The OVS bridge mode offers virtual interfaces we used to communicate with docker instances to multi-host networking, i.e., we used both internal IP addresses as our pseudo-IP and MAC addresses.

3.3.2 Service Function Chain Composition in Education Building

We consider various CH nodes in charge of collecting temperature, humidity levels, and CO2 concentration from CM sensors of occupied indoor spaces for the education buildings. We consider three VNFs implemented inside docker images: $Tensor_{VNF}$, $Mosquitto_{VNF}$, and $OneM2M_{VNF}$ as described in Figure 3.8. The first VNF monitors the current sensor's data delivery, uses TensorFlow machine learning models to predict future data from correlated sensor readings, and learns the optimal policy for the network management by avoiding redundant readings. The second VNF gathers sensor readings periodically, listens for network events through MQTT protocol, and sends commands to control fans and the HVAC system. Finally, the third VNF establishes the access to IoT resources through the hierarchical containment tree described in Section 3.2.1.

To interconnect different micro-services running in containers, we used Docker

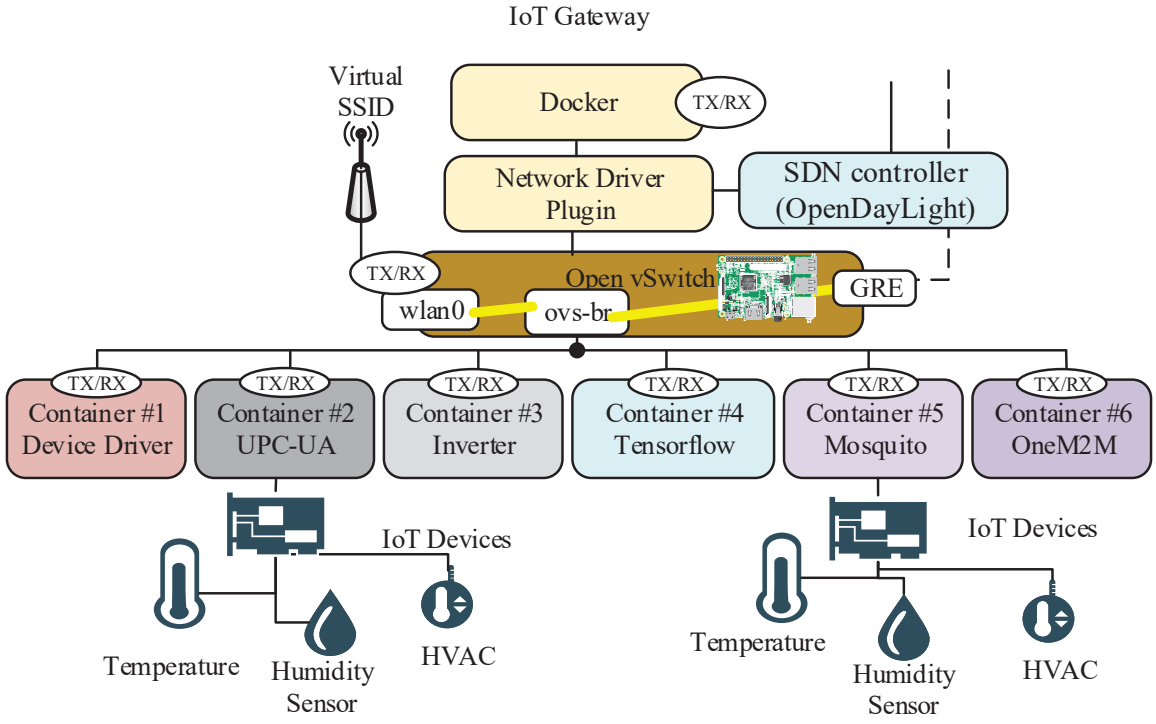

Figure 3.8: Network topology used as the target application.

engine v19.03. Docker engine offers advanced network capabilities to manage the connectivity between containers. As shown in Figure 3.8, we used Docker Network Driver Plugin to create a virtual docker network that connects to virtual SDN routers (OpenVSwitch) and handles all coordination between virtual hosts. We also configured persistent docker volume to store alarms and other debugging events outside docker containers. We also used Vagrant for automating the creation of virtual machine instances and building and maintaining portable virtual Docker containers.

We deployed Docker swarm as our container's orchestration tool to manage different VNFs we deployed in our tests. Docker swarm offers a high level of availability for the running application. In addition, we defined one of our containers as a leader to manage membership and delegation among the other containers, which we configured as workers. It is worth noting that we can use the Kubernetes (K3s) system to automate the deployment, scaling, and management of VNFs. We configured Docker Swarm to use docker-compose configuration files and scripts we created to tell the docker daemon (running inside each IoT gateway) how to pull the appropriate container image from the Docker Hub repository, how to establish networking between VNFs, how to mount storage volumes, and where to store logs for a given container.

Figure 3.9 describes the approach we used to configure the OpenDayLight SDN controller Service Function Chaining (SFC) to define an ordered list of network services (i.e., $Tensor_{VNF}$, $Mosquitto_{VNF}$, and $OneM2M_{VNF}$) which we stitched together to create a service chain. This SFC is arranged as i) physical network function (PNF) that

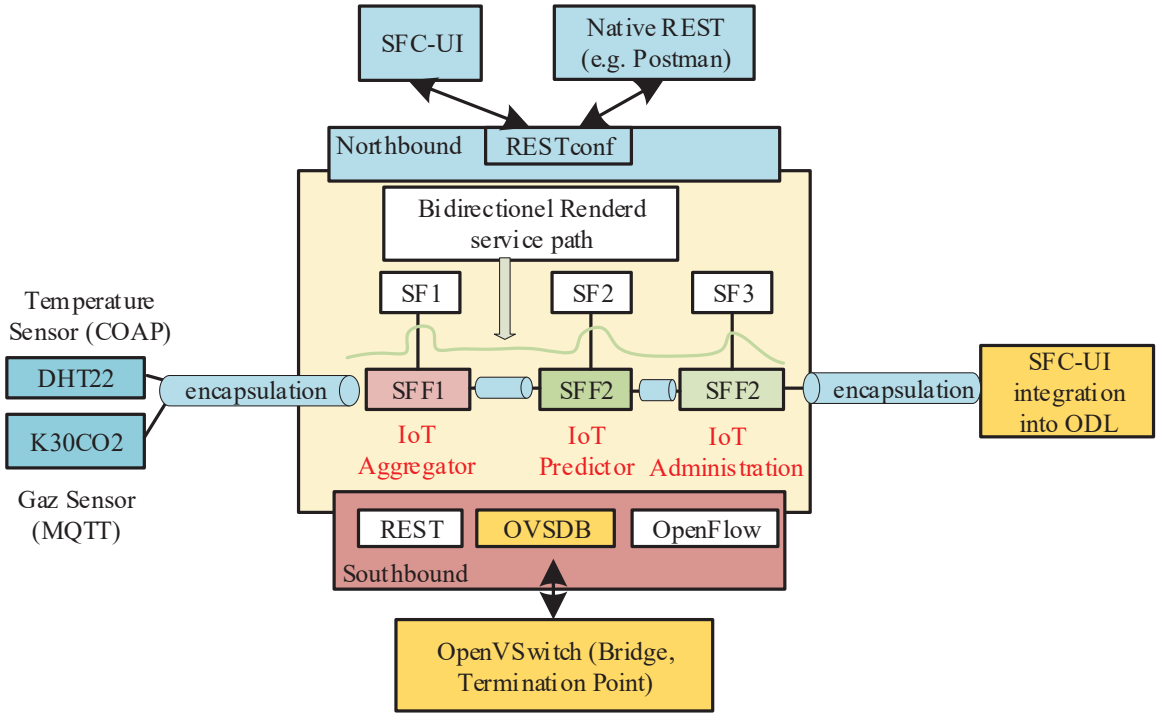

Figure 3.9: SFC composition for Temperature and CO2 measurements as an IoT service.

contains the IoT temperature and gas sensors, ii) three IoT VNFs depicted as Service Function Forwarders (SFF) in Figure 3.9, and iii) the SFC-UI integration into the SDN controller. When the IoT SF1 is updated with new sensor measurements, an updated SFC composition is triggered by the docker swarm orchestrator to pull and install a new image (or migrate an existing one) for the required VNF ($Mosquitto_{VNF}$, $Tensor_{VNF}$, or $OneM2M_{VNF}$), respectively. Our implementation shows that our approach can create, instantiate and deploy new customized on-demand virtualized IoT services that gather, process, estimate, and supervise the air conditioning inside campus buildings. Furthermore, our approach successfully collects room temperature. It sends these values to IoT VNFs through IoT gateways, which forward them to the SDN controller to switch ON or OFF the HVAC appliances based on temperature and CO2 threshold. This result is very flexible and reconfigurable as it can instantiate and deploy VNFs as needed for scalability and saves up to 70% of the energy consumption in the campus buildings.

3.3.3 Activity Management in Residential Building

Monitoring users' activity in residential buildings is critical to understanding how they interact with IoT home appliances (e.g., TV boxes, PlayStation (PS), washing machines, laptops, Lights, etc.) because different user activities usually call for additional services. Therefore, we add an Activity Recognition (AR) model to the

context-awareness model described in Section 3.2.3.

Table 3.1: Daily Occurring User Activity in smart residential building

Activity	Sleep	Use Laptop	Study in office	Watch TV	Read Book	Play PS	Prepare food	Washing clothes	No One at home
Occurrence (%)	33	10	8.3	12.5	6.8	7.1	9.2	4.3	8.8

The AR model describes the user behavior inside a smart residential home as shown in table 3.1. The AR model is trained using a semi-supervised learning model (which is included in the ML engine in Figure 3.10) to forecast the appliances a user is using during the activities described in table 3.1. For example, if user activity is "*Sleep*," only the night light of the bedroom should be ON, and all other appliances should be OFF. Similarly, if the weather is getting warmer, the HVAC or fans should be ON automatically; if no one is at home, all lights and appliances should be OFF. Figure 3.10 depicts the IoT gateway, which can be easily turned into a virtual SDN

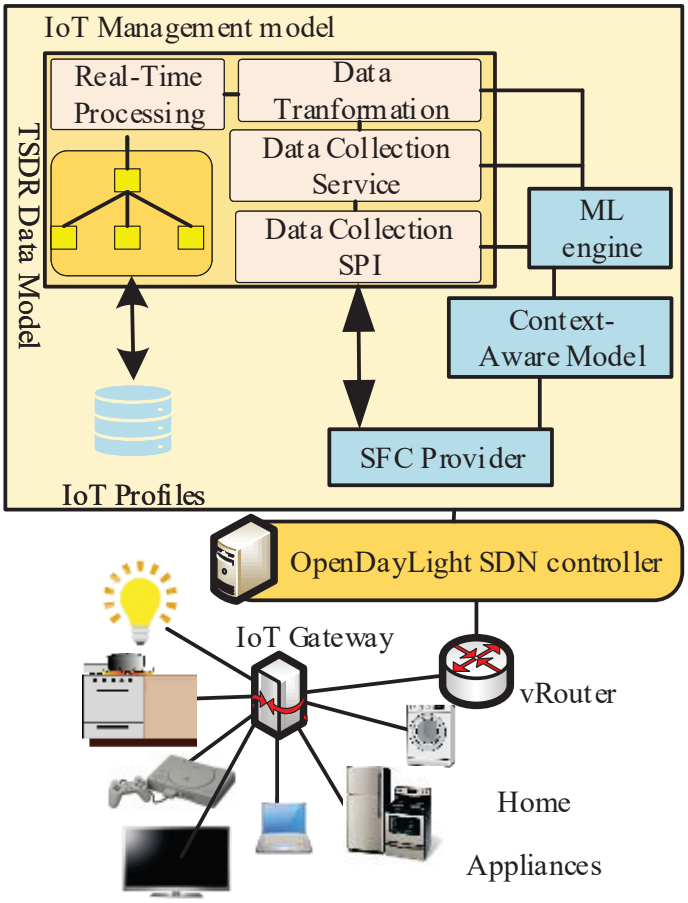

Figure 3.10: Typical smart home network as an IoT service.

router using OpenVSwitch and Docker Network Driver Plugin services. The SDN controller can manage the whole Home Area network. The controller monitors link states periodically via the link layer discovery protocol and creates the network topology. Additionally, the approach we develop in this paper allows slicing the Home Area network

into several separated logical networks. Each network partition can deal with different QoS requirements. For example, a network partition with strict QoS requirements can be configured for CCTV cameras, video streaming, etc. This result of architecture can successfully discover many valuable data from the AR model. It has also successfully created several separated logical networks, which improve the network's agility, availability, and performance. Moreover, the proposed architecture saves energy by automatically switching ON/OFF unnecessary home appliances based on the context-awareness and AR models. The SDN controller collects valuable data for context-aware service provisioning and manages the forwarding tables of SDN routers to provide comfort and assistance to building occupants and apply powerful learning models on collected data to derive behaviors that impact high energy consumption.

3.4 Conclusion

In this chapter, we have presented our contributions to answering the first problem of making a building more energy-efficient by implementing the smart energy management system that is beneficial to smart micro-grids in campus and residential buildings. Therefore, we proposed a comprehensive design of SDN architecture for green IoT networks, which is designed to empower SDN-enabled Context-Aware IoT systems and networks to create a flexible, agile, and reconfigurable framework for improving energy efficiency in smart buildings and enabling automated building operations and control. To this end, in the first section of this chapter, We have proposed the design of a novel IoT network virtualization approach that offers a high level of automation and service chaining. First, we have provided a technique for collecting, storing, querying, and managing time-series data in the SDN controller using a Time Series Data Repository (TSDR) approach. Then, we have offered a novel IoT data model that provides a data collecting facility for HVAC sensors and actuators in smart campus buildings, where data is created and consumed locally.

Then, we investigated many factors that influence energy consumption in campus buildings and created a context-awareness model that allows for the most accurate prediction of users' actions based on their daily energy consumption profiles in educational and residential facilities. Newt, for designing and delivering customized IoT services on-demand, we launched an IoT service chaining solution. Finally, we included a machine learning engine that aids the context reasoning framework in inferring context judgments and feeding back modifications to the SDN controller for automated traffic steering and policy insertion. The architecture will also offer up new avenues for the widespread adoption of energy management solutions that are based on reliable and scalable IoT services.

The following chapters will be devoted to the task allocation and resource planning

methodologies while guaranteeing consumable energy optimization reduces delay, and we will investigate the use of Blockchain and Reinforcement Learning (DRL) in SDN-enabled IoT systems to accomplish energy-aware task cyber foraging to increase the dependability, low latency, and energy efficiency.

Chapter 4

Reinforcement Learning-based Framework for Task Offloading in IoT Network Edge

4.1 Introduction

This chapter aims to introduce a novel approach for SDN-enabled IoT task offloading based on Deep Reinforcement Learning. First, in Section 4.2 we present the architectural details of our approach to supporting task assignment and scheduling while ensuring dynamic and flexible resource management of the underlying SDN-aware IoT network edge. Then, we describe in Section 4.2.2 the problem statement to dispute the optimally scheduling approach of IoT tasks and highlight our proposed approach to transfer resource-intensive computational tasks to separate IoT devices deployed at the network edge. Next, we delve into the design of our intelligent IoT network communication system based on the DRL learning algorithm. After that, we evaluate our approach to illustrate the effectiveness of our approach and validate our claims of flexible data delivery, low latency communication overhead, and a low-energy footprint. Finally, we present the concluding remarks of this chapter.

4.2 Model for task assignment and scheduling problem

This section delves into the architectural details that enable us to support task assignment and scheduling and dynamic and flexible resource management with our SDN-based framework. It presents the problem statement and the algorithms to instantiate service intelligence at the network edge.

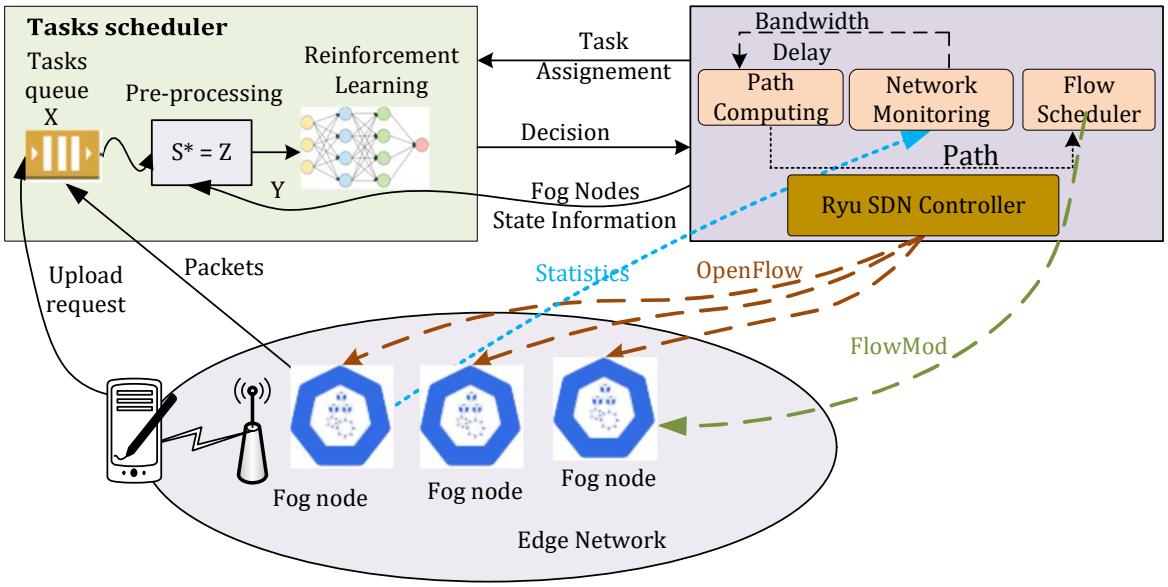

Figure 4.1: Task scheduling architecture at a glance

4.2.1 System's Architecture

Figure 4.1 glances at the architectural design of our deep reinforcement assignment and scheduling solution to address the task scheduling problem in the IoT network. We added the task scheduler at the SDN controller level to find and select the best scheduling decision policy. The algorithm that describes the task scheduler consists of a queue containing task processing requests from mobile IoT applications, a learning-based input representative, and a planning decision-maker based on learning. Expressly, the dashed lines represent OpenFlow messages (i.e., FlowMod, OpenFlow Packet-IN, and Packet-OUT, statistics such as OFPMP_FLOW that carry out Information about individual flow entries, as described by the OpenFlow Switch Specification [172]) exchanged between the Ryu SDN controller and the underlying Fog nodes, which are the data plane in our example. The messages are the continuous lines between the task scheduler and the SDN controller.

The SDN controller collaborates with our DRL algorithm to usher the process of carrying out intelligent network resource scheduling and management. The SDN controller uses a planner algorithm to manage task processing requests and create a historical dataset from incoming task requests. Figure 4.1 shows the internal controller modules: i) the path computing module assigns optimal route paths for different types of traffic generated by different IoT tasks and highly improves the QoS settings (bandwidth and delay); ii) the network monitoring module polls fog nodes to collect flow statistics to determine throughput, packet loss, and delay and; iii) the controller uses the flow scheduling module to exploit multiple paths to select the best QoS-aware path accordingly and uses queuing mechanisms to achieve optimal bandwidth utilization and supervise traffic activities. The SDN controller acts as the brain of

the network by controlling the underlying fog nodes through the OpenFlow secure channel. Additionally, our agent in the controller SDN learns to use the dataset to represent the state information of all the fog nodes and the task requests to create a latent representation model. Once the information available on the dataset is well filtered and represented in a network graph, the SDN controller starts learning how to generate learning policies to input programming decision values (Q-value). Then, it selects the best fog node and sends the decision to OpenFlow rules for handling the tasks. Finally, the SDN controller assigns the fog nodes to process the task requests. After that, the mobile device can download data from the designated fog node.

4.2.2 Problem Statement

The task planning in fog computing is represented by N different tasks $\mathcal{T} = T_1, T_2, \ldots, T_N$, which will be assigned to distinct fog nodes $\mathcal{F} = F_1, F_2, \ldots, F_M$ to minimize energy consumption and time delay by optimizing the use of the transmission channel. Let us consider:

- $X_{ij}(t)$: denotes the assignment of task T_i on fog node F_j in respect of the time t.

$$X_{ij}(t) = \begin{cases} 1, & \text{if } T_i \textit{ runs on } F_j \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

- Whenever a task is assigned to a fog node to run on it, it takes some latency time to execute in that node. We introduce the execution time of task T_i when the task T_i is assigned to a fog node F_j, which can be obtained through the following equation.

$$TTC_{ij}(t) = D_i / C_j \tag{4.2}$$

  In which D_i stands for the number of instructions of a task T_i over time, and C_j is the CPU processing rate at fog node F_j. The execution time cost in equation 4.2 describes the operating costs for the task assignment and indicates the complexity of the processing whenever the number of connected devices (i.e., requests) increases.

- The transmission time delay for task T_i on the fog node F_j is shown by equation 4.3:

$$\begin{aligned} TTR_{ij}(t) &= \frac{D_i(t)}{r_{ij}(t)} \; ; \\ r_{ij}(t) &= w_{ij}(t) \times \log(1 + \frac{h_{ij}(t) \times p_{ij}(t)}{\sigma}) \end{aligned} \tag{4.3}$$

Where $w_{ij}(t)$ is the bandwidth, $h_{ij}(t)$ is the channel power-gain, $p_{ij}(t)$ is the transmission power, and σ is the noise power. The fog transmission service rate in equation 4.3 indicates the communication time of task offloading while taking into account the wireless path fading based on the channel characteristics and the noise power spectrum density, as well as the available bandwidth per fog node.

- The queuing delay for the task T_i is denoted by equation 4.4:

$$\begin{aligned} TTW_{ij}(t) &= \frac{D_i(t)}{w_{ij}(t)} \frac{\tau}{1-\tau} \; ; \\ \tau &= \frac{D_i(t) \times a_{ij}(t)}{w_{ij}(t)} \end{aligned} \tag{4.4}$$

  Where $a_{ij}(t)$ is the average packet rate. The queuing delay, *aka residence time*, indicates how many persistent tasks in the queue to offload to a neighboring node based on the remaining queue size and incoming tasks requests

- The total time delay is given by equation 4.5:

$$TT_{ij}(t) = TTR_{ij}(t) + TTC_{ij}(t) + TTW_{ij}(t) \tag{4.5}$$

  Where $TTW_{ij}(t)$ is the queuing delay for task scheduling.

- Similarly, the energy consumption is denoted by equation 4.6:

$$EC_{ij} = TTR_{ij}(t) \times p_{ir}(t) + TTC_{ij}(t) \times p_{ie}(t) \tag{4.6}$$

  Where $p_{ir}(t)$ is the transmission power, and $p_{ie}(t)$ is the idle power.

We model the task scheduling problem as a nonlinear multi-objective combinatorial optimization problem with several objectives. The objective function is multi-variables and multi-constraints. It is tricky to find an optimal solution using a polynomial method. Thus, a compelling need is to design a hybrid heuristic algorithm proposed in this section to build a task scheduling strategy. In simplifying the complexity of the problem into a single objective problem and reducing the difficulty of solving, we consider the following hypotheses:

- Each task is independent, and there is no constraint between the tasks.
- Each task can be assigned to only a fog node.
- No task can be allocated repeatedly.
- In the calculation process, the task doesn't consider the impact of the mobility of the terminal equipment.

- All nodes are static, and the current task cannot be interrupted.

The objective function of task scheduling in fog nodes is shown in equation 4.7, where both time delay and energy consumption constraints are formulated as follows:

$$f = \min \sum_{i=1}^{N} (W_{it} \sum_{j=1}^{M} [X_{ij}(t) \times TT_{ij}(t)] + W_{ie} \sum_{j=1}^{M} [X_{ij}(t) \times EC_{ij}(t)]) \; ; \tag{4.7}$$

Subject to:

$$\sum_{i=1}^{N} \sum_{j=1}^{M} EC_{ij}(t) \leq EC_{max} \; ;$$

$$\sum_{i=1}^{N} \sum_{j=1}^{M} TT_{ij}(t) \leq TT_{max}$$

Where EC_{max} is the maximum energy consumption of our system, which represents the sum of the batteries available, TT_{max} is the maximum delay of our system, W_{it} is the weight of delay, and W_{ie} is the weight of energy consumption. Both weighting factors emphasize the importance of each type of constraint. In other words, the choice of weights in such multi-objective optimization approaches alludes to the decision-maker's preference.

4.3 Deep Reinforcement Learning for resolving Task Scheduling Problem

The DRL learning algorithm consists of objective (goal) oriented training and learning technique. An agent learns the optimal policy actions to interact with the environment and rewards it for its actions. Specifically, we define an action a belonging to a set of actions $A = \{serve, forward, discard\}$, that corresponds to the action of serving the current request, forwarding it to the neighbor node, and discard the request whenever resources are no longer available. A fog node tries to maximize its cumulative rewards and adapt to its environment to achieve the goal. The observation space (i.e., the state space) representing the environment at a given step describes the state of task requests to access a given service, e.g., number of task requests, number of granted requests, and number of allocated resources for that task request.

Figure 4.2 illustrates our approach to tackling the issue scheduling problem using deep reinforcement learning. Because the task planning module contains a small amount of information about the future arriving tasks, such as arrival time and task size, the SDN controller uses historical tasks to build the deployment decisions. In addition, the DRL algorithms we implemented inside the controller can analyze the performance of all connected fog-enabled IoT nodes. We build efficient scheduling to

execute several simultaneous tasks and predict optimal scheduling on the network that meets both low-latency and efficient-energy requirements.

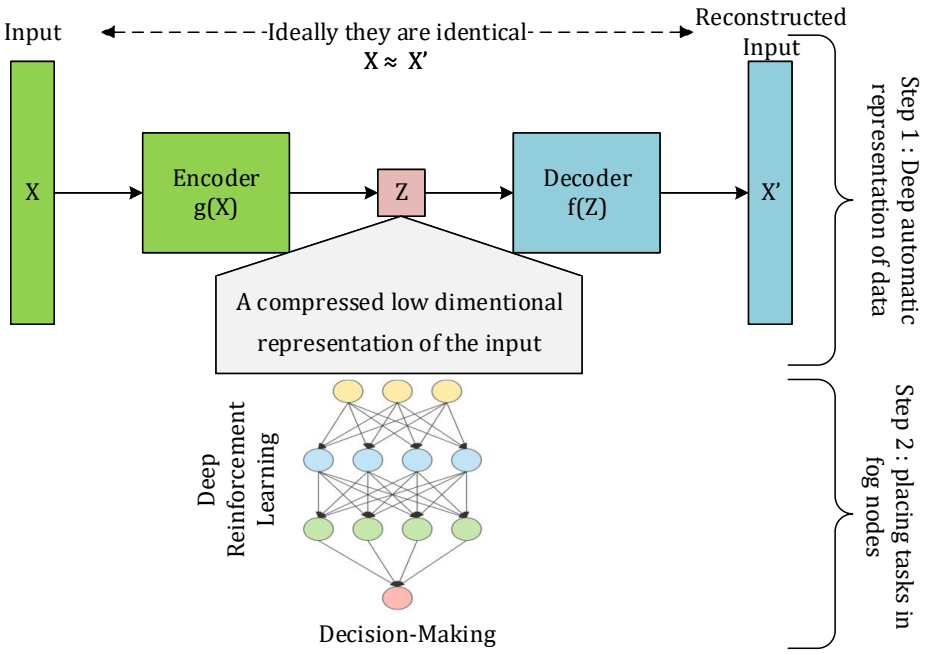

Figure 4.2: Deep Learning for Task Scheduling

First, we train the SDN controller to represent the datasets of all tasks and fog nodes for performing task assignments using the best optimal way under the constraints mentioned above, minimizing the network latency and reducing the energy consumption. Therefore, we introduce our DRL algorithm to select and apply the best decision that distributes the tasks on available fog nodes. As shown in Figure 4.2, the compressed low-dimensional representation of the input is used to find a latent representation of the data between tasks that will be executed and fog node states ready to execute these tasks. Then, an auto-encoder model was developed, which aims to find a latent representation Z from data X using an encoder and decoder networks. The main goal of this model is to compute the following function $g(x) = sg(Wx+b)$, where sg is an activation function as $sigmoid()$, W is the weight, and b is the bias. After that, a bottleneck layer $Z = g(x)$ filters the incoming data from the encoder layer. Then, a decoder function defined as follows $f(x) = sg(W'z+b')$ is used to reconstruct the input X from Z (representation of latent space). The auto-encoder model is trained using the mean squared error (MSE), which minimizes the reconstruction error between the input X and the reconstructed input (output) X'. Furthermore, the obtained latent representation Z, obtained by the encoder network, i.e., $S^* = Z = g(S)$, is used to train the SDN controller to assign the task T_i to the node F_j and generate the optimal decision to schedule the tasks.

4.3.1 Task Assignment

Algorithm 1 illustrates the task assignment we implemented inside the SDN controller, which collects information from the underlying SDN routers about the available fog node capacities, including their available energy. Thence, the algorithm receives a list of tasks and their characteristics and then assigns them to the available fog nodes. The DRL algorithm selects the fog nodes based on their available energy and current occupation rates to reduce the delay in processing time. Once the controller has assigned tasks to their relevant nodes, it keeps track of a log dataset of the current node's processing and available energy.

Algorithm 1: Task Assignment to Fog Nodes

Input:

1. Detection nodes $N = \{n_1, n_2, \ldots, n_j\}$ with their available energies,
2. Set of tasks $T = \{t_1, t_2, \ldots, t_i\}$ with their characteristics

Output: Assign a task t_i to a node n_j

```
1 while true do // infinite loop
      // learn according to cases
2     Replay (n, t) ;
      // Predict the value of the reward
3     act-values = predict (n, t) ;
      // Choose the action according to the expected reward
4     a = arg max(act-values[0]) ;
5     Execute a // Send t to n
```

Whenever a task is successfully assigned to a fog node, the controller increases the value of the local reward and selects the forthcoming action according to the expected reward. Then, to maximize the objective function (c.f., equation 4.7), the algorithm applies the *argmax* operator to find the maximum values that fulfill low-energy consumption constraints and lower network latency.

4.3.2 Deep Q-Learning Policy Algorithm

First, we introduce our Deep Q-Learning policy using experience replay to learn a small data block to avoid biasing the dataset distribution. Then, the algorithm approximates the Q-value function to explore all the states and determine all possible actions to reach the optimal solution. The algorithm uses continuous learning through the experience replay to update the algorithm parameters based on the previous actions. As described in Algorithm 2, the SDN controller implements a Deep Q-learning algorithm to mimic a learning agent that maps states of the environment to actions. The agent considers these actions to move from one state to another to maximize a numerical reward over

time.

Algorithm 2: Deep Q-learning algorithm with experience replay

Input: Initialize replay memory M
Initialize action-value Q with random weights
Observe the initial state s
Output: A trained model to optimally assign a task to node

repeat
  Select an action a with probability ϵ
  Select a random action;
    Otherwise select $a = \arg\max_{a'} Q(s, a')$
  execute action a
  observe reward r and new state s'
  `// Store experience in memory`
  store exp $< s, a, r, s' >$ in memory M
  sample random transitions $< ss, aa, rr, ss' >$ from memory M
  compute target for each minibatch transition
  if *ss' is terminal state* **then**
    $tt \leftarrow rr$
  else $tt \leftarrow rr + \gamma \max'_a Q(ss', aa')$

  Train the Q network using $(tt - Q(ss, aa))^2$ as loss
  $s = s'$
until *terminated*

Specifically, the SDN controller selects these actions during run-time, even if an agent doesn't complete the knowledge of rewards and state transition functions. In each state, the agent can choose between two types of behavior: (i) the controller can continue exploring the state space to find optimal decision policy; or (ii) it can leverage the information already given by the Q values defined by equation 4.8:

$$\begin{aligned} Q(S^*) &= R + \gamma \max Q(s', a') \\ Action\ a &= \arg\max \_a' Q(s, a') \end{aligned} \tag{4.8}$$

The total reward is given by equation 4.9:

$$R = \sum_{i=1}^{n} (W_{it} \sum_{j=1}^{m} \beta[X_{ij}(t) \times TT_{ij}(t)] + W_{ie} \sum_{j=1}^{m} \beta[X_{ij}(t) \times EC_{ij}(t)]) \tag{4.9}$$

Where W_{it} is the delay weight and W_{ie} is the weight of energy consumption. The parameter β takes a positive sign if we execute the assigned task in the node, whereas a negative sign otherwise remains pending in the queue. We implemented the training algorithm that uses a regression loss function to reduce the total training data error. Worthy of mention, the deep learning neural network loss function, to predict the states of Q values, given by equation 4.10 with a hyper-parameter setting γ as a discount

factor :

$$L = \frac{1}{2}[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]^2 \tag{4.10}$$

Algorithm 2 learns the allocation policy to provide an optimal decision regarding both the constraints mentioned above, i.e., in terms of latency and energy and performance of the system. First, to start the learning process, the algorithm initializes a decision matrix with weights (Q-values) of random policies and observes the initial states of the SDN network. Then, nodes will be chosen with their smallest probability values from the filled matrix and assigned the tasks randomly to its distinct fog nodes. Once the first step is completed, the controller can move to the following states, i.e., returning a reward and performing the calculation, and transitioning from one state to another. Each newly calculated step is saved in the matrix, and we compare the existing policy with the previous one. If the newer policy is better, it will be considered locally, optimal, etc. Thus, we repeatedly operate until we get an optimal global assignment for each task. Then, the operation is repeated until all tasks in the waiting queue are processed and assigned to the best available fog nodes.

4.3.3 Random Learning Policy

We introduced dense random neural networks that randomly process a data block while simultaneously improving the learning policy's robustness and accuracy. Algorithm 3 represents the assignment of tasks using a random agent. The latter is based on a strategy that randomly assigns the tasks to the various available fog nodes without considering the quality of service in terms of latency. It assigns each task to a randomly chosen fog node and updates its energies each time. In addition, it also updates the accumulated rewards.

4.3.4 Deterministic Learning Policy

We developed a deterministic learning policy to dictate what action to take given a particular state. Indeed, we consider a different situation where incoming IoT tasks are known to forecast the next event precisely from the current event. The value of the state is the expected reward if we start from it and continue using the same policy. Nonetheless, the deterministic policy does not involve that the reward remains the same. Algorithm 4 represents the assignment of tasks to the corresponding nodes using a deterministic agent. The latter will schedule the tasks one by one according to their order of arrival and assign them to nodes close to their minimum latency. At each iteration, an update will be performed on the energy state of the node that executed the task.

Algorithm 3: AGENT RANDOM

Input:

1. Detection nodes $N = \{n_1, n_2, \ldots, n_n\}$ with their available energies,
2. Set of tasks $T = \{t_1, t_2, \ldots, t_m\}$ with their characteristics

Output: Assign task t_i to node n_j

$i \leftarrow 1$
$reward \leftarrow 0$
$totrewards \leftarrow 0$
while $i <= m$ **do** `// m is the number of tasks`
    $action \leftarrow env.action_space.sample()$ `// randomly choose an action`
    $ob, reward \leftarrow env.step(action)$
    update(N,ob) `// update of nodes`
    $totrewards+ = reward$
    $i++$

Algorithm 4: AGENT DETERMINISTIC

Input:

1. Detect nodes $N = \{n_1, n_2, \ldots, n_n\}$ with their available energies,
2. Set of tasks $T = \{t_1, t_2, \ldots, t_m\}$ with their characteristics

Output: Assign task t_i to node n_j

Sorted(N) $i \leftarrow 1$
$assign \leftarrow false$
$reward \leftarrow 0$
$totrewards \leftarrow 0$
while $i <= m$ **do** `// m is the number of tasks`
    for $j \leftarrow 1$ ***to*** n **do** `// n is the number of nodes`
        if $N[j]["energ"] > T[i]["energ"]$ **then**
            $N[j]["energ"] \leftarrow N[j]["energ"] - T[i]["energ"]$
            $ob, reward \leftarrow Execute(T[i], N[j])$ `// execute task in node`
            $assign \leftarrow true$
            break
    if $!assign$ **then**
        $ob, reward \leftarrow env.step()$
    update(N,ob) `// update of nodes`
    Sort(N)
    $totrewards+ = reward$
    $i++$

4.3.5 Asynchronous Actor-Critic Agent (A3C) Algorithm

We design Asynchronous Advantage Actor-Critic (A3C) Algorithm to involve global network optimization in parallel by using multiple agents. These agents have their own set of parameters to create different situations to interact with the fog nodes distributed in the environment. Each agent harvests a different learning experience and adds it to the overall learning experience. Algorithm 5 illustrates the task assignment and scheduling scheme using the A3C approach, which relies on multiple agents that are likely to explore different states and transitions. Each agent has its network parameters and a copy of the environment.

Algorithm 5: Asynchronous Advantage Actor-Critic Agent (A3C)

Output: Model trained with workers to assign task to node.

for $i \leftarrow 1$ ***to*** n **do** // n is the number of workers
  Wi.run() // Start worker thread
$step \leftarrow 1$ // ForEach worker Wi, initialize step counter
$T \leftarrow 0$ // Initialize episode counter
repeat
  $d\theta \leftarrow 0$; $d\theta_v \leftarrow 0$ // reset gradients
  $\theta' \leftarrow \theta$; $\theta'_v \leftarrow \theta_v$; $t \leftarrow step$ // Synchronize thread-specific parameters
  $s \leftarrow s_t$ // Initialize iteration
  // Get observation state
  while *s is not terminal and* $step - t < t_{max}$ **do**
    Simulate action a_t according to $\pi(a_t|s;\theta\ )$
    Receive reward r_t and next state s_{t+1}
    $step++$;
    $T++$;
  if s_t *is terminal state* **then**
    $R \leftarrow 0$
  else $R \leftarrow V(s_t, \theta'_v)$ // Bootstrap from last state

  for $i \leftarrow step - 1$ ***to*** t **do**
    $R \leftarrow r_i + \gamma R$
    $d\theta \leftarrow d\theta + \Delta_{\theta'} log(\pi(a_i|si;\theta')(R - V(s_i;\theta'_v)))$
    // Accumulate gradients
    $d\theta_v \leftarrow d\theta_v + \frac{\partial((R-V(s_i;\theta'_v))^2)}{\partial\theta'_v}$
  // Perform asynchronous update of θ and θ_v
  $\theta = \theta + d\theta$
  $\theta_v = \theta + d\theta_v$
until $T > T_{max}$

These agents interact with their respective environments asynchronously while learning at each iteration. Thus, each agent gets its copy of the environment and

processes the gathered data samples at their arrival. The main thrust of A3C is that the network controls each agent to gain more acknowledge and contribute to the complete knowledge of the network.

Algorithm 5 (*lines 20-24*) illustrates the policy update we developed using the A3C approach. As shown in *line 21*, the A3C agents select different actions in order to maximize the discounted reward R by updating the hyper-parameter settings such as discount factor γ. The agents try to maximize the immediate rewards by taking greedy actions using the policy function π and the Value function V to impact future global parameters vectors $d\theta$, as shown in *line 22*. The update operation is performed until reaching the maximum number of predefined iterations T_{max}.

4.4 Performance analysis

This section describes the testbed setup and shows the evaluation of our solution. Specifically, we describe the results for commonly used evaluation metrics, such as latency, energy efficiency, and network scalability.

4.4.1 Testbed Setup

We implemented our framework using an emulated SDN environment comprising Mininet [173] as our network emulators along with OpenFlow SDN switches for creating different IoT scenarios. Furthermore, we extended Mininet to support Open AI Gym toolkit [174] for reinforcement learning and deployed IoT nodes in the form of virtualized micro-services using Docker containers in emulated Kubernetes clusters. We also implemented the SDN northbound application using Python-based Ryu [175] SDN controller, which performs global traffic management, load balancing, global topology discovery, and monitoring. We developed our solution using the TensorFlow python interface for interacting with our SDN environment. In the latter, we used to run tests on more than 100 nodes, each running over $1,000$ tasks simultaneously. Furthermore, we assessed our solution versus deterministic algorithms, random, and A3C approaches. The deterministic agent plans tasks according to their order of arrival and assigns them to the nearest nodes regarding their minimum latency. Whereas the random agent assigns tasks to the available nodes in a stochastic order, i.e., it assigns tasks to the available nodes' strategy less. If a selected node does not have the required capacity to execute an incoming task, then the random agent leaves its in-hold state in the waiting queue and assigns tasks to alternative nodes. Thus, the A3C approach uses multiple agents who independently learn a policy from their environment and then collaborate with other agents to create global knowledge to choose the best decision. The detailed simulation hyperparameters are given in Table 4.1.

Hyperparameter	Value
Learning rate	0.001
Gamma	0.95
Batch size	32
Exploration Max	1.0
Exploration Min	0.01
Exploration Decay	0.995

Table 4.1: Hyperparameter values in simulation

4.4.2 Pre-processing

The first step we performed on our datasets comprises pre-processing input data to carry out the training of our SDN controller (i.e., the Ryu controller). Carrying out data refinement allows properly representing and preparing data for our deep Q-learning model to perform task assignments and scheduling. Therefore, we implemented different techniques to reduce the dimension of our datasets to find the best representation of our data.

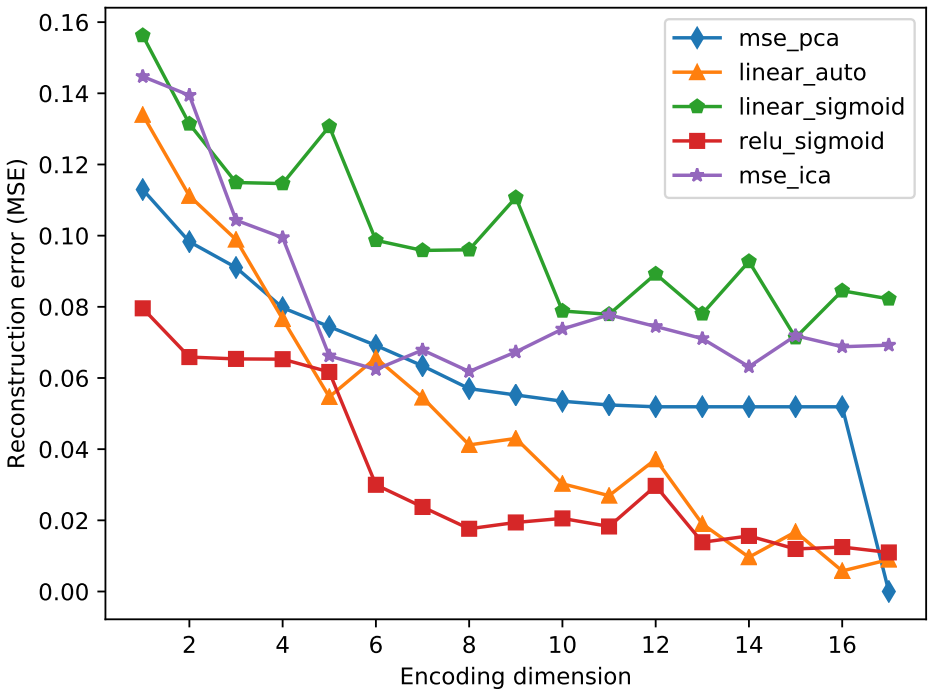

Figure 4.3: Data Pre-Processing

Figure 4.3 illustrates the Mean Square Error (MSE) regression loss function. This yield was obtained for different refinement techniques, including Principal Component Analysis (PCA), Independent Component Analysis (ICA), Deep auto-encoder with Sigmoid function, rectified linear activation function (reLU), and linear function. As underscored by Figure 4.3, the Deep auto-encoder with Sigmoid activation function (i.e., the Relu-Sigmoid) performs better filtering and refinement results while keeping

the MSE error minimum. We create multiple local minima to find optimal task assignment strategies. The sigmoid function makes the loss function non-convex, rather than creating a single global minimum for our training.

4.4.3 Discounted Cumulative reward

The SDN agent in run-time collects states from the environment and sends back information to the controller. By trying different actions, the agent learns to optimize the reward from the environment. The controller can either decide to take the current policy as the best decision to place tasks on the selected fog-enabled nodes, or continue learning from the available distributed nodes to find a better candidate to place the current task requests. All agents are driven by the same goal, to maximize the expected discounted return.

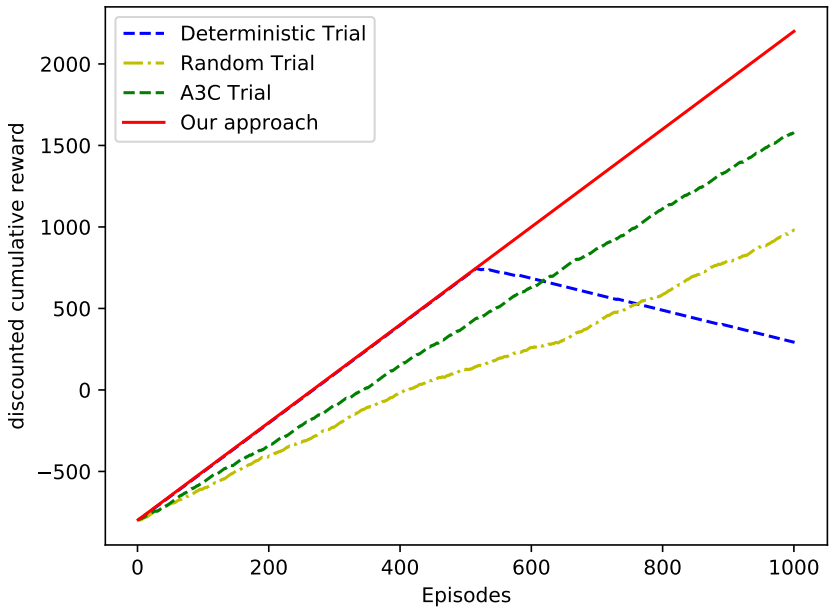

Figure 4.4: Cumulative Rewards for Our approach against the three other approaches.

Figure 4.4 compares the accumulated reward got by our approach versus the deterministic, random, and A3C approaches. After the deterministic agent increases to almost 550 earned rewards, it decreases and starts losing rewards. It ushers, losing its computation power and its ability to complete the planning of newer coming tasks. As a result, his cumulative reward curve slowly increases for the random agent, which means some tasks have not been assigned, and we must put them on hold state. For the A3C agent, its cumulative rewards slowly increase compared to our approach. Our approach performs better cumulative rewards than the deterministic and random case, selecting the available node based on the energy level. For the A3C trial agent, it has

a lower accumulated reward compared to our approach. The update of the cumulative reward curve of our agent is increasing rapidly compared to other agents. Thus, the agent has a very optimal investment strategy where the action is selected each time. It motivates it to maximize the rewarded return.

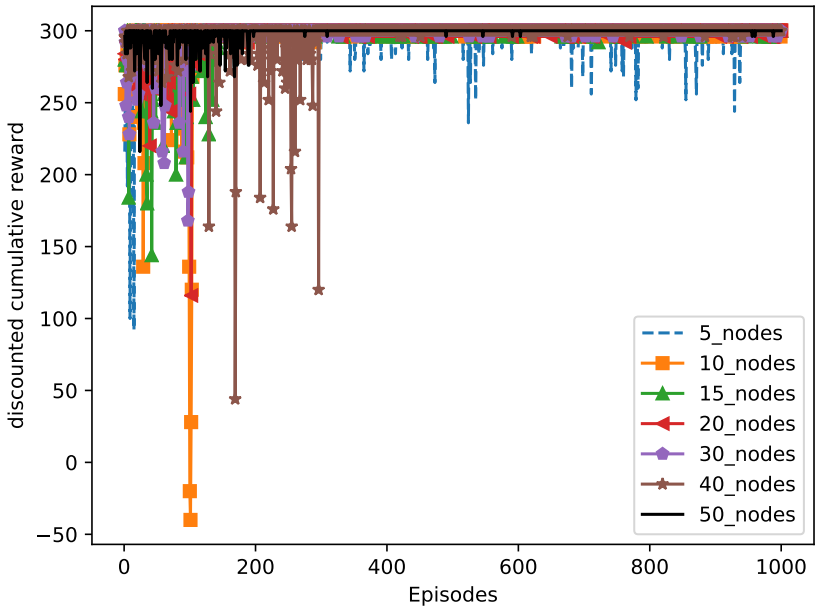

Figure 4.5: Local Rewards with Increasing number of nodes

To evaluate the stability and the scalability of our approach, we increased the number of fog nodes up to 50, as shown in Figure 4.5. Our SDN-enabled decision-maker agent could quickly learn from the SDN topology network to make optimal decisions. As a result, the local reward, i.e., optimal local assignment, rapidly reaches almost 300 in a few dozen episodes, as illustrated in Figure 4.5, which means: that optimal local minimum, i.e., local optimization, can be performed rapidly. We also experimented with our approach with over 100 in other scenarios (none shown in Figure 4.5), and we observed the same behavior. Thus, we claim that our deep learning approach helps implement local, optimal, and global task assignments and schedules for SDN-enabled IoT networks while respecting QoS constraints.

4.4.4 Energy-Efficiency

We aim to reduce the energy consumption of running fog-enabled IoT nodes as we described in equation 4.6 and perform better energy efficiency as we consider all nodes as batteries-powered ones. To evaluate the energy efficiency of our SDN-enabled solution, the SDN controller trained the agent by $1,000$ episodes. As a result, the agent should be able to plan 100 tasks for energy-constrained fog nodes. However, each fog

node has a limited power capacity, i.e., their battery level during this planning process is close to $5,000wh$.

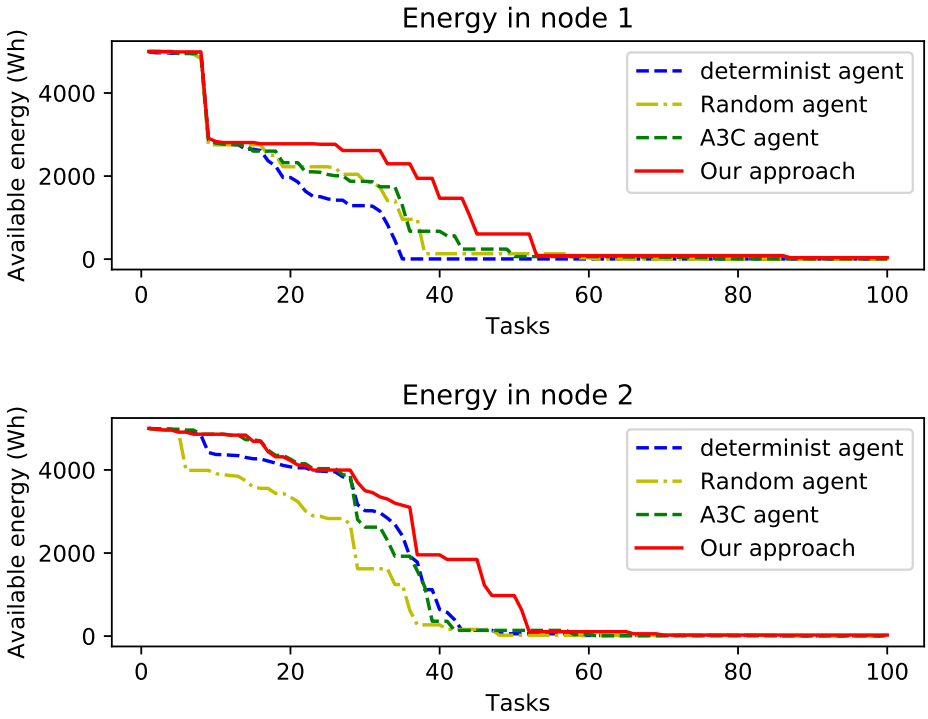

Figure 4.6: Energy Consumption in both tasks' executing fog nodes

Figure 4.6 illustrates the energy consumption of two available fog nodes, each running up to 100 tasks simultaneously, using our training approach against deterministic, random, and A3C training agents. Throughout the planning strategy of these tasks, the DRL agent in our approach keeps a better battery level in both nodes compared to the three other algorithms.

Recall that our primary objective is to minimize the overall energy consumption of our SDN-enabled fog network, as we described in equation 4.6. Figure 4.7 shows that our approach flags out better energy efficiency, i.e., up to 87% compared against both deterministic agents, which perform 48%, and the random agent that performs 58%, and the A3C agent performs 76%.

Our results confirm our claims that the solution we propose can readily be used to optimize task scheduling and dynamically assignment of complex jobs with task dependencies in distributed fog IoT networks.

4.4.5 Assessing the Latency performance

Our optimization approach aims at minimizing the network latency for available nodes during the task executions, as we described in equation 4.5. We gauged the total time delay expected by available fog-enabled nodes to process the current task's requests and communicate the results to remote IoT senders.

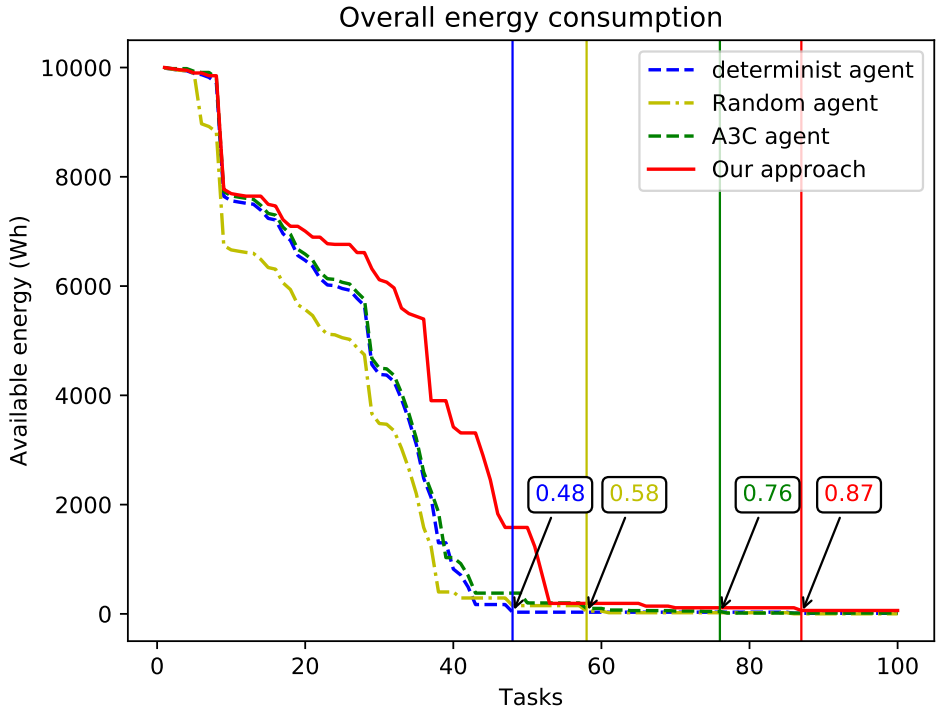

Figure 4.7: Energy Saving and Efficiency for all available Fog nodes

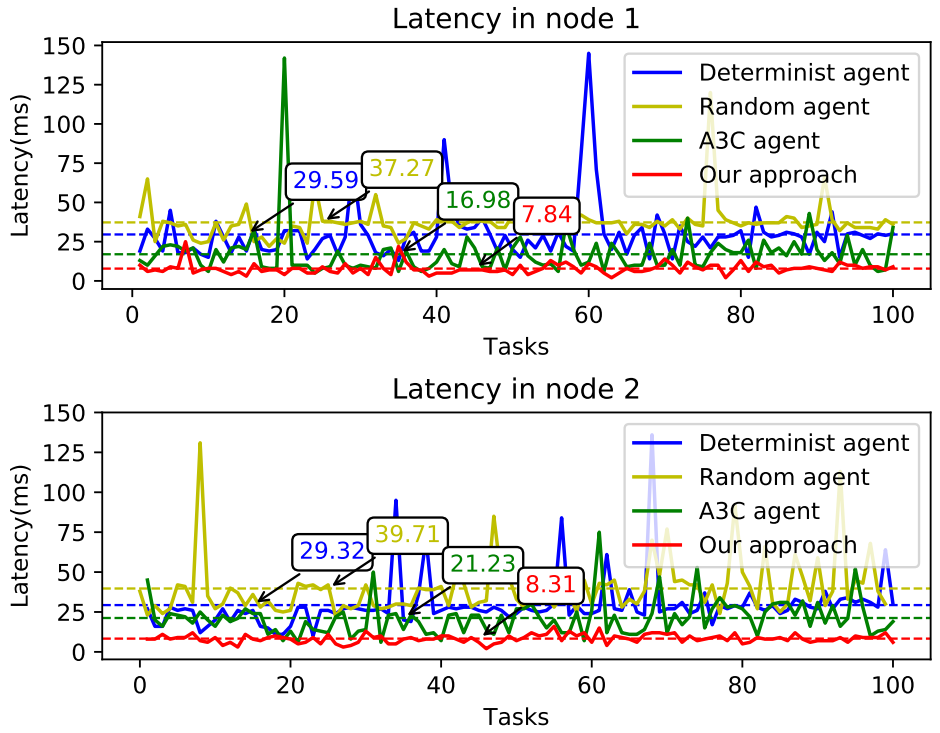

Figure 4.8: Evaluating Latency during Task Scheduling

We measured the total delay the available fog nodes expected to handle the pending task requests. The set of latency values collected during task scheduling of our approach, as well as with other approaches to scheduling tasks in two battery-powered nodes, as shown in Figure 4.8.

The latency results of the different approaches have been summarized in Table 4.2.

The deterministic approach performed a time latency of 29.59 ms in node 1 and 39.32 ms in node 2. Likewise, the random algorithm carried out 37.27 ms in node 1 and 39.71 ms in node 2. Finally, the A3C algorithm presents an average latency of 16.98 ms at node 1 and 21.23 ms at node 2. We repeated these experiments multiple times, and we found the average latency expected by our approach is close to 7.84ms in node 1 and 8.31 ms in node 2. Therefore, our approach ensured a minimum latency of all fog nodes and showed significant latency and energy consumption performances.

Agent	node 1	node 2
A3C	16.98	21.23
Random	37.27	39.71
Deterministic	29.59	39.32
Our approach	7.84	8.31

Table 4.2: Average latency

4.4.6 Evaluating the Bandwidth performance

To assess the performance of our approach, we studied the bandwidth usage of our IoT network during the task scheduling process. Figure 4.9 illustrates the bandwidth usage during the scheduling by the different approaches described in section 5.2.2. According to Figure 4.9, our approach outperforms all the other approaches during task scheduling. Compared to both random and deterministic approaches, which performed *29.6* Gbits/s and *32.15* Gbits/ respectively, our approach outperformed both of them. The reason is that the deterministic approach uses a deterministic greedy policy to make the locally optimal choice at each stage without exploration. Furthermore, in our IoT network, the deterministic algorithm will stick to the current state during that task assignment step when it is better than the observed states. Nonetheless, the deterministic greedy policy will find itself trapped in a local optimum, failing to explore certain other states, which may hold a better local optimum solution. Similarly, the randomized exploration approach (randomized policy) will explore different states based on a specific probability distribution, making it slightly slower. Expectedly, our approach reaches *34.70* Gbits/s, which outperforms the *33.4* Gbits/s obtained with the A3C approach, even when we use many parallel workers (agents) that interact asynchronously with separate instantiations of the environment. On the other hand, the A3C technique tends to suffer when faced with more complex tasks. It takes long delays between actions and relevant reward signals, i.e., known as Partially Observable Environments.

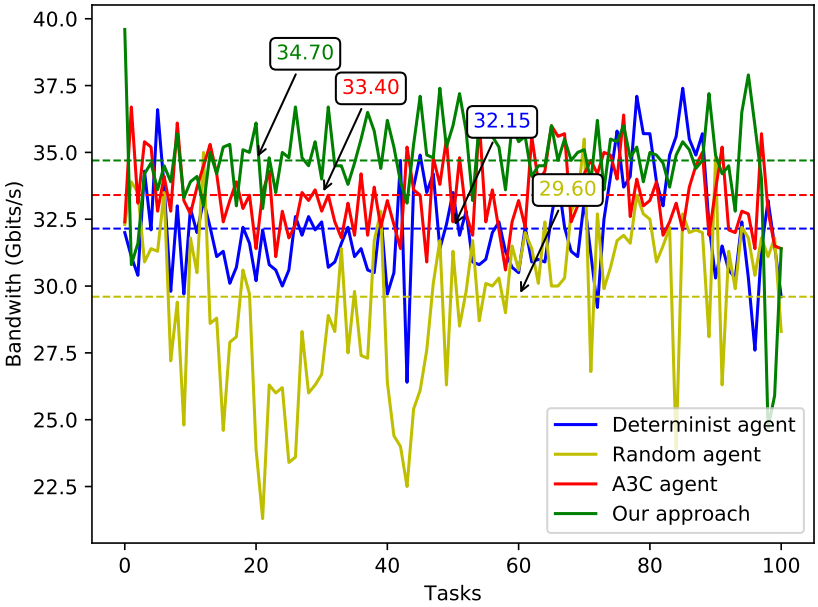

Figure 4.9: Evaluating The bandwidth utilization during task Scheduling

4.5 Conclusion

This chapter describes our approach to developing and implementing task assignment and scheduling mechanisms for SDN-enabled IoT networks using Deep Reinforcement Learning. The first contribution concerns the development of an empirical model and formulating a task assignment and scheduling problem that minimizes network latency while ensuring energy efficiency. The second contribution aims to develop different DRL algorithms, including deterministic placement algorithm, random, and A3C strategies, to show their effectiveness in offloading IoT tasks and reducing the amount of data carried on the IoT network edge, improving the latency, energy consumption, and the network overhead. Finally, we showed that our intelligent approach outperforms baseline algorithms in real-time selecting optimal allocation decision policies for task assignments and scheduling. Furthermore, we have demonstrated that our approach performed both local and global optimization, ensuring lower-latency communication and increased energy efficiency.

Although these solutions improve our answer to the problem of this chapter, they, however, suffer from some other limitations regarding the security and privacy of data exchanged across distributed IoT networks, which will be addressed in the next chapter.

Chapter 5

Enhancing Decentralization of IoT Network with Blockchain and Deep Reinforcement Learning

5.1 Introduction

In this chapter, we provide a deep learning technique that integrates SDN and blockchain to accomplish job scheduling and offload, increase the response rate of IoT services, and try to execute dynamic resource management. First, we introduce in Section 5.2 an overview of the architectural design of our solution approach. First, we introduce in Section 5.2 a novel architecture that offers a trustworthy IoT nodes placement with respect. Then, in Section 5.2.2 we formulate a task assignment and multi-objective Markov chain optimization approach to model the transition of trust features in Blockchain-enabled nodes. Next, we study in Section 5.3 different types of models that can address our task scheduling problem in our proposed platform through Deep Reinforcement Learning algorithms including Deep Q-Learning (DQN), Double DQN (DDQN), Dueling DQN and Dueling DDQN, as well as the Asynchronous Actor-Critic Agent (A3C) algorithm. We also show the feasibility of blockchain technology to construct a trust-enabled task offloading scheme for SDN-enabled IoT networks. After that, we evaluate in Section 5.4 our proposed architecture in terms of low-latency communication overhead and low-energy footprint. Finally, we conclude this chapter in Section 5.5.

5.2 Overview of the Architectural Design

We used Fog Computing technologies based on blockchain and SDN to have fast, secure, and reliable communication in an IoT environment. Specifically, our work helps to efficiently distribute tasks to fog nodes to serve application requests from IoT

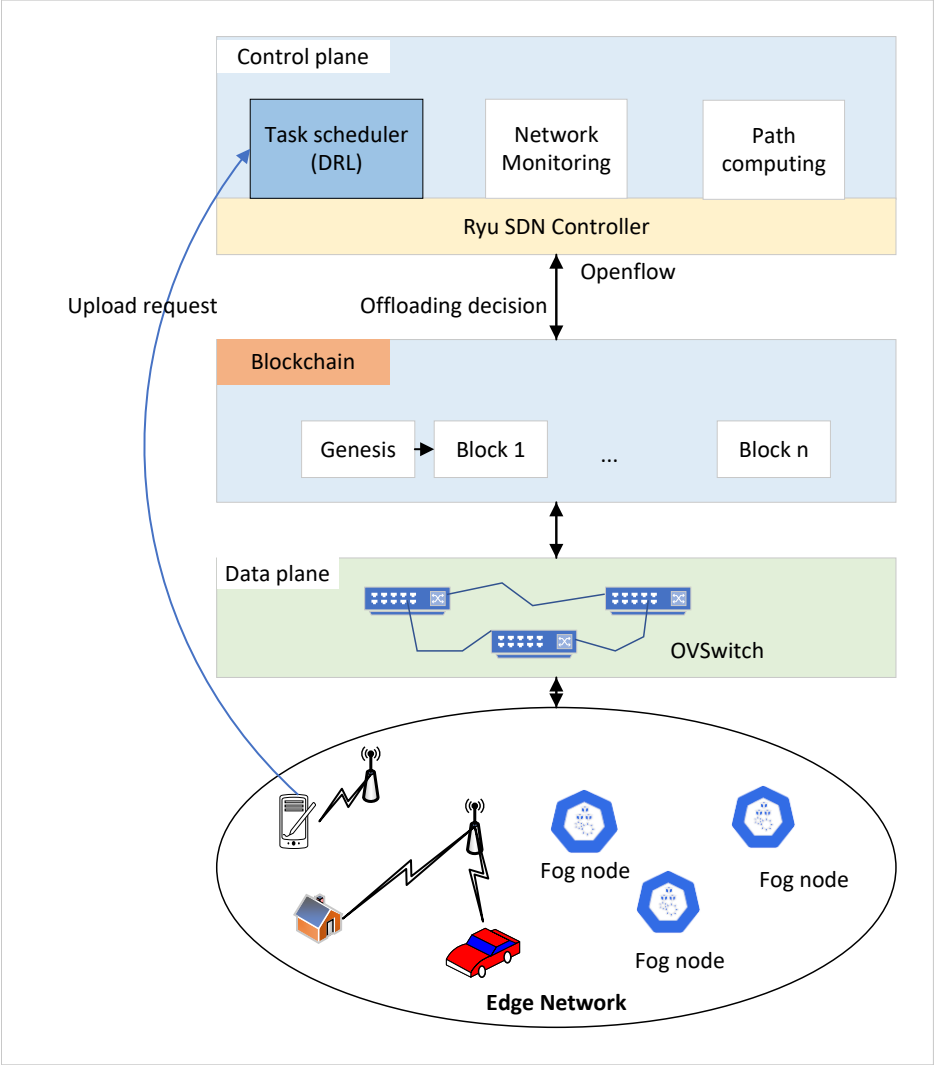

Figure 5.1: Blockchain-based SDN for Task Scheduling in IoT Network

devices with very low latency and conserve the overall system powered-battery.

5.2.1 System Architecture

Figure 5.1 shows a high-level architecture of our approach, which involves the following layers.

1. **Edge Network**: This layer represents the perception of the IoT environment and contains the IoT sensors and devices responsible for detecting the data in real-time and transmitting it to the next sublayer. The IoT devices send their requests to perform their tasks to the controllers at the fog nodes.

2. **Data plane**: The fog nodes are represented by this layer, which incorporates a variety of network equipment and devices such as SDN switches and routers. The IoT devices can be provided service by fog nodes to compute some tasks. SDN-based switches are in charge of sending data and receiving OpenFlow rules

by the SDN controller.

3. **Blockchain**: This layer ensures the protection of all types of communications between IoT stakeholders, such as switch and SDN controller interactions. Blockchain nodes allow for creating an encrypted record of all transactions that is impossible to alter and can be shared with the whole network devices.

4. **Control plane**: This layer maintains a logical and high-level view of the network topology provided to different services for managing the network. This is important for efficient offloading tasks assigned to available resources at fog nodes.

5.2.2 Problem statement

This section presents different aspects of our system from a modeling point of view. First, we explain task scheduling using deep reinforcement learning (DRL) in an IoT environment.

5.2.2.1 Task scheduling: Deep Reinforcement Learning based SDN Controller

In fog computing, task scheduling is a service that allows efficient distribution of the arriving tasks coming from mobile IoT applications to process in the fog nodes. The objective is to maximize efficiency and the capacity of task processing requests coming from mobile IoT applications while guaranteeing minimal latency with energy-saving. To address the task scheduling and offloading problem, we combine fog networking-based Blockchain with the SDN architecture for flexible and centralized operations, where the control plane provides decision-making according to a better strategy. In addition, the data plane performs the forwarding of tasks. Hence, we added service at the Northbound plane to connect with the SDN controller level to find and select the best scheduling decision policy and offload a task. The Task scheduler service consists of a queue containing task processing requests from mobile IoT applications and a scheduler decision-maker based on deep reinforcement learning.

An approach can be formulated to perform the task scheduling problem, where the SDN controller can be picked as the agent to make a series of decisions using deep reinforcement learning (DRL) [176]. Specifically, DRL is concerned with learning to solve a problem by trial and error. A Markov Decision Process (MDP) based model, motivated by recent progress in implementing DRL techniques, can be used to formalize our problem concerning task scheduling and offloading.

5.2.2.2 Secure Communication Based on Blockchain

The IoT is sometimes seen as an opening for cyber-attacks. This can sometimes be the case because you are connecting equipment that was not previously available and was not originally designed with these cyber threats in mind. Security issues are summarized when communicating due to a breach of security policies. To ensure secure communication and avoid the anonymity of the data transmitted, Blockchain protocols must be integrated with the different layers of the SDN-IoT architecture, using the encryption function used to create and validate blocks. These blocks contain the transactions that will be added to the Blockchain ledger that can be distributed on the network. To ensure P2P communication between stakeholders, an Ethereum decentralized exchange protocol allows users to create smart contracts. These smart contracts are based on a computer protocol to verify or enforce a mutual contract. They are deployed and publicly available on a Blockchain.

To secure communications in the network IoT, we use Blockchain smart contracts. A smart contract is a transaction verification application that is submitted to the Blockchain network and executed dynamically [177]. He acts as an agent for two IoT devices, including the fog node and SDN controller. The SDN controller, in our case, will store data on the Blockchain network using a smart contract, and the fog nodes (SDN Switches) will use the smart contract to collect data stored on the Blockchain network. A smart contract is a software-defined protocol that digitally enforces the settlement of a contract between multiple participating IoT nodes within the Blockchain network that will perform traceable and immutable transactions.

5.2.2.3 Statement of the Scheme

It is assumed that:

- The SDN controller is safe
- All other IoT devices might have been hacked.

The SDN controller receives a set of A tasks requested by the mobile IoT applications, assigned to the corresponding fog nodes to process them. The tasks are represented by:

$$\mathcal{A} = \{A_1, A_2, \ldots, A_A\} \tag{5.1}$$

And the fog nodes denoted by:

$$\mathcal{F} = \{f_1, f_2, \ldots, f_N\} \tag{5.2}$$

The SDN controller communicates with the IoT network stakeholders from the Blockchain (BC) system, where the OpenFlow data will be transferred as transactions in a block

in a time slot. These blocks must be verified and validated by a single fog node already selected by the BC (BCO) through a special consensus.

There are T time slots during the period when a period is a time between an invalidated block issued by the controller, and he is replied with a validated block. The time instant denoted by t, when $t \in \{1, 2, 3, \ldots, T\}$

Let consider $\gamma^f(t)$ represent the real trust features of fog node f; when $f \in \mathcal{F}$ and we consider the set of available trust features of fog node f is $\Phi = \{\Phi_0, \Phi_1, \ldots, \Phi_{P-1}\}$

We can use a Markov chain to model the transition of trust features in BC nodes. $\Rightarrow$ The transition probability of $\gamma^f(t)$ from one state V_s to another state W_s be $KV_sW_s(t)$; where

$$KV_sW_s(t) = Pr(\gamma^f(t+1) = W_s | \gamma^f(t) = V_s) \quad \text{where } V_s \text{ and } W_s \in \Phi \tag{5.3}$$

The message denoted m communicated between the SDN controller and another fog node may be verifying signatures, generating a message authentication code (MAC), or verifying MAC.

Let consider $A_m = \{S_m, C_m\}$ denote a computation task related to message m, where S_m is the size of message m and C_m the required number of CPU cycle to complete this task.

We model the computation resources in fog node f for BC system as random variable Υ^f. The parameter Υ^f can be split into Y discrete intervals as $Y = \{Y_0, Y_1, ..., Y_{Y-1}\}$ and we denoted $\Upsilon^f(t)$ the computation resources at time slot t. We use a Markov chain to model the transition of computation state in fog nodes.
$\Rightarrow$ The transition probability of $\Upsilon^f(t)$ from one state a_s to another state b_s be $Za_sb_s(t)$, as

$$Za_sb_s(t) = Pr(\Upsilon^f(t+1) = b_s | \Upsilon^f(t) = a_s) \quad \text{and } a_s, b_s \in Y \tag{5.4}$$

The execution time of computation task A_m, can be denoted:

$$Ex_m = \frac{C_m}{\Upsilon^f(t)} \tag{5.5}$$

Thus, the computation rate is:

$$R^f(t) = a^f(t)\frac{S_m}{Ex_m} = a^f(t)\frac{S_m\Upsilon^f(t)}{C_m} \tag{5.6}$$

where $a^f(t)$ means when fog node allocated (equal 1) to execute a task related to message or not allocated (equal 0). At time slot t, only one fog node is assigned to the BC system.

According to a task assignment strategy described in the part 5.2.2.1, the SDN

controller must generate ($b \geq 1$) stream entries to transmit the packets. We specify P as the processing strategy and r is a data of flow rule to contain the decision to assign tasks to fog nodes. P and r must have the following relation:
$P = \{r_1, r_2, \ldots, r_b\}$; where $r = X_{ij}$; denotes the assignment of task A_i on fog node f_j computed in the following way:

$$X_{ij} = \begin{cases} 1, & \text{if } A_i \textit{ runs on } f_j \\ 0, & \text{otherwise} \end{cases} \tag{5.7}$$

Subsequently, the SDN controller sends to each switching device the flow rules, which contain the task assignment decisions using the blockchain network in the form of transactions that are represented by:
$H = \{h_1, h_2, \ldots, h_b\}$

To have a confidential communication between the SDN controller and the switches and that the flow rules will be secure, it is necessary that the content of P and itself in H.

An attacker can make his interventions mainly on the traffic by adding, deleting, and modifying flow entries. For example, suppose that g data for flow rules are attacked in P, so malicious flux data P' is made. The scenarios we can have been:

- Delete g flow entries: $P' = \{r_1, r_2, \ldots, r_{b-g}\}$
- Add g flow entries: $P' = \{r_1, r_2, \ldots, r_{b+g}\}$
- Modify g flow entries: $P' = \left\{r_1, r_2, \ldots, r'_{b-k+1}, \ldots, r'_b\right\}$

With our approach, the attacker cannot make modifications to the data flows and cannot inject flows into the switch flow tables without going through Blockchain, where the BCO will reject his interventions in the phase of verification described in the section below.

5.3 Proposed approach

In this section, we explain how to respond efficiently to requests for assigned tasks to fog nodes according to an efficient strategy while minimizing end-to-end delays and extending the battery life of our system by optimized power consumption. The task assignment decision and offloading will be communicated securely through the means of the blockchain, as illustrated in the figure 5.2.

5.3.1 Task scheduling

The IoT environment is very dynamic, making it challenging for the SDN controller to plan tasks properly. As a result, we employ the Deep Reinforcement Learning-

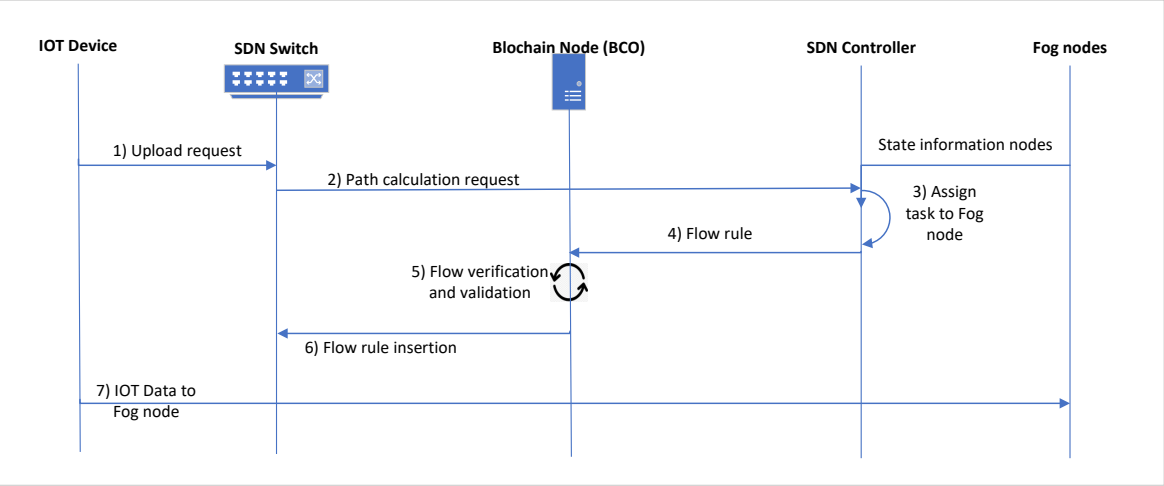

Figure 5.2: Assign task with secure communication

based technique for effective task placement in the relevant fog nodes to anticipate the best policy. Furthermore, the SDN controller uses a learning agent based on the Asynchronous Advantage Actor-Critic (A3C) algorithm [117] which maps the states of the environment to the actions that the agent can take to move from one state to another to maximize a reward at the overtime. Indeed, the SDN controller selects these actions during execution, even if an agent does not complete the knowledge of rewards and state transition functions.

In each state, the agent can predict both the value function $V(s)$ and the optimal policy function $\pi(s)$. The learning agent uses the value of the Value (Critical) function to update the optimal policy function (Actor), which means the probabilistic distribution of the action space. The learning agent determines the conditional probability $P(a|s;\theta)$, i.e., the parameterized probability that the agent chooses action a when it is in state s. The agent also learns how much better the rewards were than expected. This gives the agent new insight into the environment, and thus the learning process is better. The following expression gives the benefit metric:

$$A = Q(s,a) - V(s) \tag{5.8}$$

The learning of the task allocation policy is represented by algorithm 6 based on the Asynchronous Advantage Actor-Critic (A3C) model, which relies on multiple workers that are likely to explore different states and transitions in our environment. Each worker has its network parameters and a copy of our environment that makes it possible to provide an optimal decision to assign the tasks to the fog nodes. These workers interact with their respective environments asynchronously, learning with each interaction. A global network controls each worker. As each worker gains more knowledge, it contributes to the total knowledge of the global network. A global network provides each worker with more diverse training data. This configuration allows each worker

to gain knowledge from the experiences of another agent, allowing the overall "global network" to be better.

Algorithm 6: Asynchronous Advantage Actor-Critic Agent (A3C)

Output: Model trained with workers to assign task to node.

for $i \leftarrow 1$ ***to*** n **do** // n is the number of workers
  Wi.run() // Start worker thread
$step \leftarrow 1$ // ForEach worker Wi, initialize step counter
$T \leftarrow 0$ // Initialize episode counter
repeat
  $d\theta \leftarrow 0$; $d\theta_v \leftarrow 0$ // reset gradients
  $\theta' \leftarrow \theta$; $\theta'_v \leftarrow \theta_v$; $t \leftarrow step$ // Synchronize thread-specific parameters
  $s \leftarrow s_t$ // Initialize iteration
  // Get observation state
  while s *is not terminal and* $step - t < t_{max}$ **do**
    Simulate action a_t according to $\pi(a_t|s;\theta\ )$
    Receive reward r_t and next state s_{t+1}
    $step++$;
    $T++$;
  if s_t *is terminal state* **then**
    $R \leftarrow 0$
  else $R \leftarrow V(s_t, \theta'_v)$ // Bootstrap from last state

  for $i \leftarrow step - 1$ ***to*** t **do**
    $R \leftarrow r_i + \gamma R$
    $d\theta \leftarrow d\theta + \Delta_{\theta'} log(\pi(a_i|si;\theta')(R - V(s_i;\theta'_v)))$
    // Accumulate gradients
    $d\theta_v \leftarrow d\theta_v + \frac{\partial((R-V(s_i;\theta'_v))^2)}{\partial\theta'_v}$
  // Perform asynchronous update of θ and θ_v
  $\theta = \theta + d\theta$
  $\theta_v = \theta + d\theta_v$
until $T > T_{max}$

5.3.2 Secure communication

In our approach, we have designed a smart contract executed by the blockchain system to provide secure and confident communication in the IoT network based on SDN. We will use Ethereum blockchain technology [148] for intelligent contract management.

In figure 5.2, we show the process from demand to treat the task request by IoT device to offloading data while ensuring the verification and validation by blockchain node. Indeed, the SDN controller decides with his technique (DRL) to choose an optimal fog node corresponding to performing task requests. He generates a flow rule

OpenFlow to be inserted in the OpenFlow table of SDN Switch. Both interacting nodes, such as IoT device requests and fog nodes, must add their IDs within the packet obtained from the SDN controller.

After that, the SDN controller sends the packet containing data for the flow rule to storage in the blockchain. However, the blockchain node (BCO) receives the packet from the SDN controller. He will be verified by ID (hash address) with other nodes. The BCO is responsible for assigning all the nodes distributed in the network to Blockchain hash addresses. When a fake node attempts to access the network pretending to be real with a fake ID (hash address), it will not be authenticated on the recipient's side as it will not be stored on the distributed blockchain ledger. Hence, the data from the sender node will also be deleted the node will be blocked for further communications. On the other side, if the recorded Blockchain addresses verify the data obtained from a sender node, the fog node considers the data valid and deposited in the database.

For the validation of transactions, we use Proof-of-Authority (PoA) consensus in Ethereum Blockchain [178] where the mining is carried out faster because only the predefined Blockchain ledger authority has the privilege to mine the block and the rest of the nodes maintain data transparency.

5.3.3 BC-based consensus protocol

The SDN controller collects the OpenFlow data from the Switches. Then it generates the decisions necessary for optimally executing the task planning service.

These decisions will be considered as transactions will be validated in a block. The SDN controller communicates with the BC system to verify and validate its block and synchronizes it with the network's stakeholders.

This block verification and validation process follows the following steps:

1. The SDN controller sends the block to all nodes by the BC system. Only one node will be selected as an ordered node (BCO) responsible for processing the controller request. First, the BCO node verifies the MAC of the block. Then it is a signature when they are valid. It then verifies the MAC of all transactions and their signatures.
   We consider the following parameters:

   - α: generating one MAC
   - αc: verifying one MAC
   - β : verifying one signature
   - bs: the batch size of transactions issued by the SDN controller
   - v : the size of valid transactions

We ignore the cost of sending and the cost of receiving. The cost at BCO is calculated as follows: $(1 + bs/g) * (\alpha + \beta)$

2. The BCO node sends a message to all the other nodes of the BC. The latter checks the MAC of BCO and the signatures and the Mac of each transaction.
   - The cost at the BCO node is equal to $(N - 1)\alpha$
   - The cost at each node of BC without BCO node is: $\alpha + (bs/g)(\alpha + \beta)$
3. Subsequently, each BC node sends a message to all the other nodes from which each node will receive D messages. the BCO node must verify D MAC, and the other nodes generate $(N - 1)$ MAC and verify D MAC.
   - The cost at the BCO node is $D\alpha$
   - The cost at each node of BC without BCO node is $(N - 1 + D)\alpha$
4. After that, all nodes send a response message; hence, each node will receive D messages. The BCO node needs to generate $(N - 1)$ MAC and verify D MAC, and each other node generates $(N - 1)$ MAC and verifies D MAC, so it's the same cost at the BCO node and other nodes.
   The cost is : $(N - 1 + D)\alpha$
5. Finally, the SDN controller receives a D message from all nodes consisting of a validated block, and the ledger of the BC system will be updated by adding this new block. The BCO node and the other nodes need to generate (bs/g) MAC.
   - The cost at the controller is : $(bs/g)\alpha$
   - For one transaction, the cost at BCO node is :
     $I = (\frac{1}{g} + \frac{1}{bs})\beta + (\frac{1}{g} + \frac{1}{bs})\alpha + (\frac{2N+2D-2}{bs})\alpha$
     and the cost at the other node is :
     $J = (\frac{1}{g})\beta + (\frac{2}{g})\alpha + (\frac{2N+2D-2}{bs})\alpha$

The energy consumption defined as following :

$$EC = (I + J) \times P_{ie} \tag{5.9}$$

Where P_{ie} is the idle power

The objective function is to minimize the throughput and energy consumption formulated as follows:

- We assume as the computation speed of τ Hz for each module in the fog node,
- We consider the order node (BCO) is not fully trusted. He has Z trust to affect the system performance, and $Z \in [0, 1]$.

$$\Psi = \min[\frac{Z\tau}{I}\frac{Z\tau}{J}\frac{Z\tau}{EC}]$$

$$\Psi = \min[\frac{Z\tau}{(\frac{1}{g}+\frac{1}{bs})\beta+(\frac{1}{g}+\frac{1}{bs})\alpha+(\frac{2N+2D-2}{bs})\alpha} \frac{Z\tau}{(\frac{1}{g})\beta+(\frac{2}{g})\alpha+(\frac{2N+2D-2}{bs})\alpha} \frac{Z\tau}{[(\frac{2}{g}+\frac{1}{bs})\beta+(\frac{3}{g}+\frac{1}{bs})\alpha+(\frac{2N+2D-2}{bs})2\times\alpha]\times P_{ie}}] \quad (5.10)$$

5.3.4 Formulation Problem

We formulate this problem as a Markov decision process (MDP), defining the state action, action space, and the reward function.

- **State space**: The learning agent must detect all the trust characteristics of all the nodes ($\gamma^N(t)$) as well as the computing capacities of all the fog nodes $\Upsilon^N(t)$. Therefore, the state space will be represented as follows:

$$S(t) = \begin{bmatrix} \gamma^1(t) & \gamma^2(t) & \dots & \gamma^N(t) \\ \Upsilon^1(t) & \Upsilon^2(t) & \dots & \Upsilon^N(t) \end{bmatrix} \quad (5.11)$$

  where $S(t)$ represent a state at time slot t

- **action space**: the agent needs to decide the selection of an ordered node (BCO) among the nodes of the BC system and the allocation of computation resources of the fog node.
  The action space denoted by: $A(t) = \left\{O^S(t), A^M(t)\right\}$.
  - $P^S(t) = [o^1(t), o^2(t), \dots, o^S(t)]$; 1 is order node (BCO) and 0 is otherwise
  - $A^M(t) = [a^1(t), a^2(t), \dots, a^M(t)]$; 1 the fog node allocated and 0 is otherwise

- **reward function**: To improve the throughput and the energy consumption of the consensus protocol for the BC system, we define the reward function as follows:

$$\Re = \lambda[\frac{Z\tau}{(\frac{1}{g}+\frac{1}{bs})\beta+(\frac{1}{g}+\frac{1}{bs})\alpha+(\frac{2N+2D-2}{bs})\alpha} + \frac{Z\tau}{(\frac{1}{g})\beta+(\frac{2}{g})\alpha+(\frac{2N+2D-2}{bs})\alpha} + \frac{Z\tau}{[(\frac{2}{g}+\frac{1}{bs})\beta+(\frac{3}{g}\frac{1}{bs})\alpha+(\frac{2N+2D-2}{bs})2\times\alpha]\times P_{ie}}] \quad (5.12)$$

  The parameter λ takes a positive sign whenever the assigned task is offloaded and executed by the fog node and a negative sign otherwise.

5.4 Performance Analysis

In this section, we present a simulation scenario to evaluate the performance of our proposed approach. First, we describe the tested setup, followed by simulation results and the corresponding discussions.

5.4.1 Testbed Setup

Mininet [173] as our embedded network emulators and Ethereum as a private Blockchain, as well as OpenFlow SDN, switches to create different IoT scenarios. Additionally, we extended Mininet to support the Open AI Gym toolkit [174] to carry out our deep reinforcement learning. Besides, we implemented an SDN northbound application based on the SDN Ryu controller [175] to perform global IoT traffic management, load balancing, global topology discovery, and monitoring. Furthermore, we developed our algorithms based on TensorFlow, an open-source Python-based artificial intelligence library for fast numerical computing. We also created an SDN-enabled blockchain interface using Web3.py, a Python library built for interacting with the Ethereum blockchain. We have developed over $1,000$ of IoT tasks simultaneously inside each IoT edge node for the experiments. Each IoT edge node generates over $10,000$ blockchain blocks over time. Each block hold over 10^6 IoT transactions. To evaluate the effectiveness of our approach against different DRL approaches, we first have evaluated different blockchain consensus algorithms, i.e., Proof of Work (PoW) and Practical Byzantine Fault Tolerance (PBFT), against our solution based on the Proof of Authority (PoA). Then, we developed different DRL approaches, i.e., Deep Q-Learning (DQN), Double DQN (DDQN), Dueling DQN, and Dueling DDQN, and evaluated them against our approach based on Asynchronous Actor-Critic Agent (A3C) Algorithm. The detailed simulation hyperparameters setting is given in Table 5.1.

Hyperparameter	Value
Learning rate	0.0001
Gamma	0.99
Update-freq	20
Num-Threads	5
Optimizer	RMSprop

Table 5.1: Hyperparameter values in simulation

5.4.2 Experimental Results and Discussion

5.4.2.1 Evaluation of Network Latency for different consensus algorithms

Various consensus algorithms have been studied to achieve reliability in IoT networks where multiple distributed unreliable nodes are involved in different communication scenarios. Although blockchain enables sharing IoT data safely and storing all kinds of log records on it, however, incurs a higher time delay in committing transactions. IoT devices need low latency when making permanent changes within the transactions due to their limited computation and constrained resources. Therefore, a consensus mechanism for IoT edge networks should also reduce latency to validate IoT blocks on the blockchain. The blockchain consensus time delay, also called blockchain network latency, is the time delay for the first confirmation of acceptance of a transaction by the network.

The evaluation of our solution approach for three types of consensus PoW, PBFT, and PoA are depicted in Figure 5.3 as a comparative histogram in terms of network latency. In addition, our approach is based on PoA and is compared against two other consensus mechanisms, i.e., Pow and PBFT.

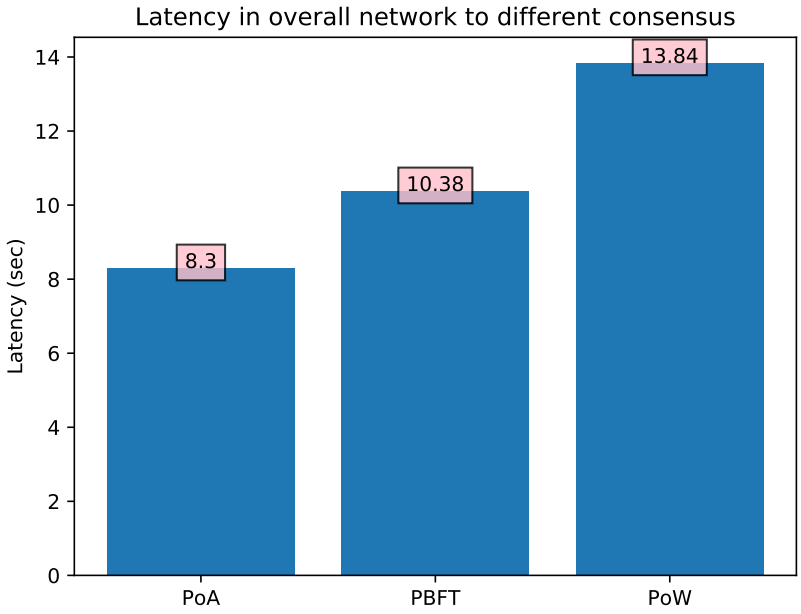

Figure 5.3: Latency in overall network to different consensus.

The PoW requires computational resources to mine the block versus the PoA and PBFT, which provide equal importance to all participating nodes in the peer-to-peer IoT network. This figure shows that the latency with PoA less latency is almost $8.3sec$ to mine a block versus the PoW and PBFT.

Our experiments show that the latency of PoW consensus achieves a higher latency

than other consensus algorithms. Indeed, PoW consumes a lot of computational resources to confirm a block. Similarly, although the practical Byzantine fault tolerance (PBFT) achieves better latency performance against the PoW because it gives equal importance to all the participating nodes of the peer-to-peer network, it performs a higher latency than our approach that implements the PoA consensus. Specifically, our approach achieves an average latency of 8.3 seconds for confirming an IoT transaction. In addition, the voting-based PBFT consensus achieves an average latency of 10.38 seconds to validate a transaction. The PoW consensus validates a new IoT transaction in an average time delay of 13.84 seconds. Therefore, our approach outperforms these approaches and archives better network latency.

5.4.2.2 Evaluating the Transactions Throughput

There are two subcategories of blockchain throughput: the read throughput and the transaction throughput. The former measures the rates at which data are read, i.e., the number of reading operations completed in a given period, formally expressed in several reads per second (RPS). However, read throughput is often not considered

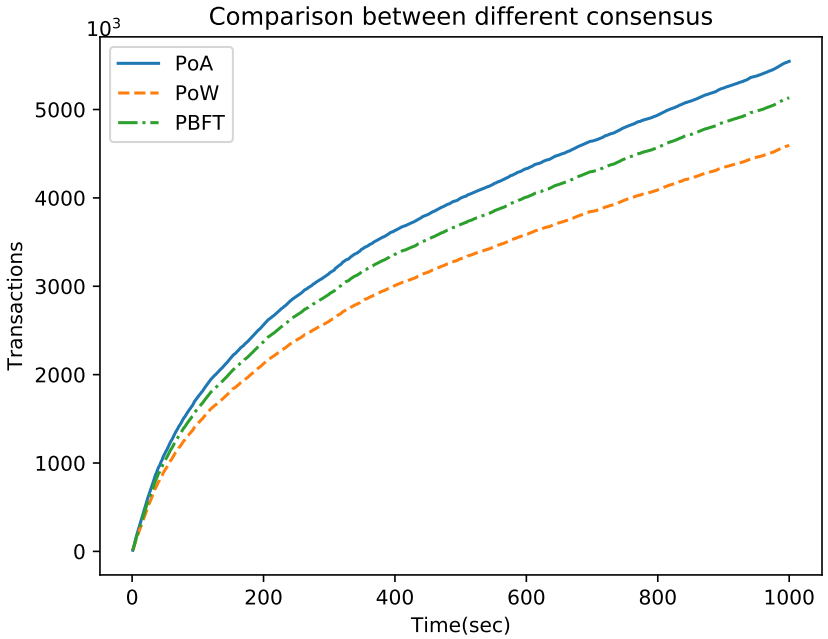

Figure 5.4: Transaction Rates (TX/sec) for Different Consensus Mechanisms

a critical metric to evaluate the quality of service in the blockchain network, since blockchain nodes typically achieve a significantly higher read and query efficiency. The letter is the transaction throughput, which measures how fast the blockchain can process the incoming IoT transactions, expressed in transactions per second (TPS). The blockchain network throughput is not measured at a single node, but it reflects the overall performance of the blockchain network across all nodes. Specifically, the

blockchain network throughput is the relationship of the total valid transactions per the whole time in seconds. Figure 5.4 shows the transaction rates for the PoW, PBFT, and our implementation of the PoA consensus algorithm. The transaction throughput increases linearly with the increase in block size. Our approach based on the PoA consensus achieves $6,000$ transactions compared against the PoW and the PBFT. The PoW achieves the lowest transaction throughput, i.e., an average of $4,000$ transactions per second, and the PBFT achieves an average throughput of $5,000$ transactions per second.

5.4.2.3 Valid blocks for SDN-BC system

The controller sends an invalidated block that contains transactions representing the data of OpenFlow rules. AFTER ATTAINING CONSENSUS, the BC system then delivers the relevant verified block to all fog nodes.

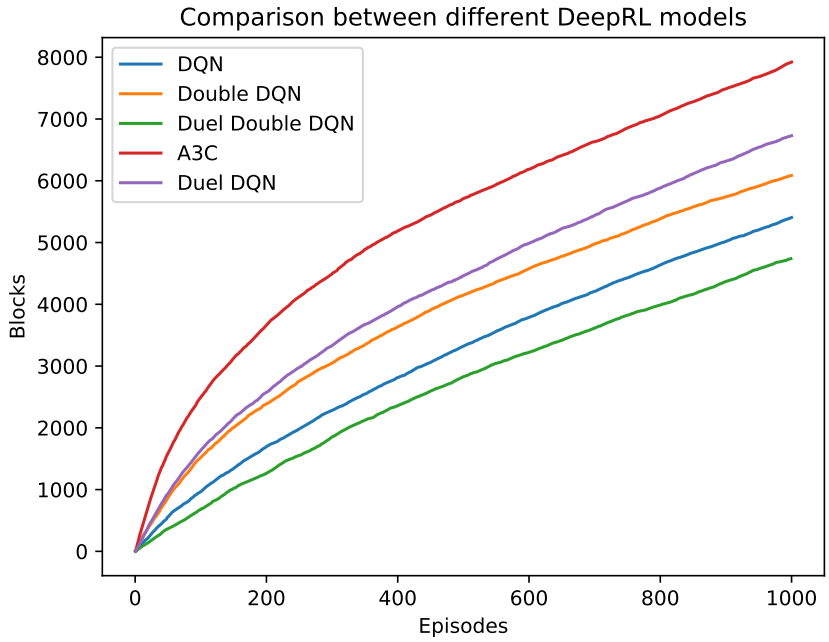

Figure 5.5: Comparison between different DRL models.

All fog nodes then learn the payloads in each transaction, allowing them to recognize events and OpenFlow orders from the SDN controller. Finally, these steps are carried out in the consensus phase, when a fog node can synchronize network-wide views.

Figure 5.5 shows a comparative study of valid blocks concerning training episodes for different Deep Reinforcement Learning models in the SDN-BC System, where each point is the average number of valid blocks per episode. Each DRL agent runs in AdamOptimizer [179] with a learning rate of $1e^{-4}$. This figure shows that training agents for job scheduling in an SDN-BC System allow almost $8,000$ blocks to be verified

and validated by an A3C agent using five workers, where each worker is assigned to a fog node. However, figure 5.6 presents the valid transactions in our SDN-BC System

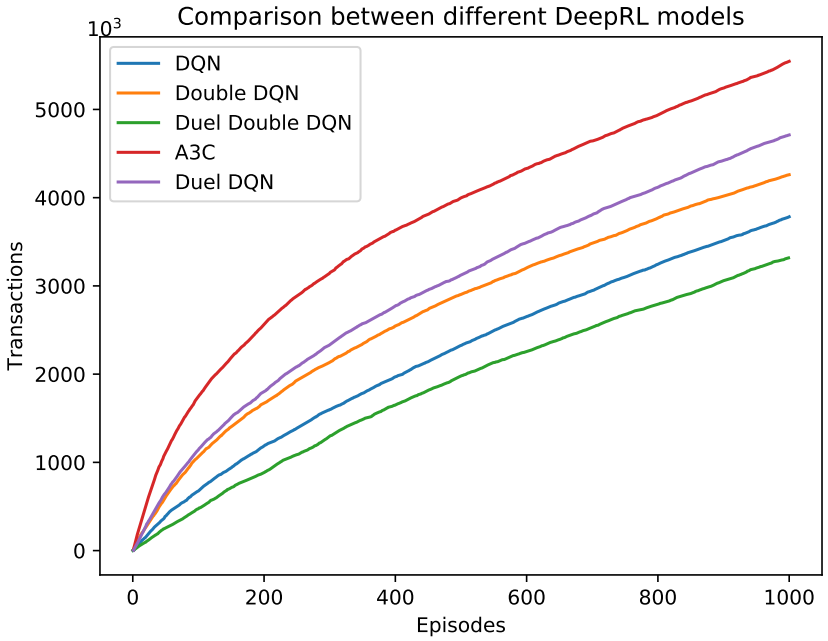

Figure 5.6: Comparison between different DRL models.

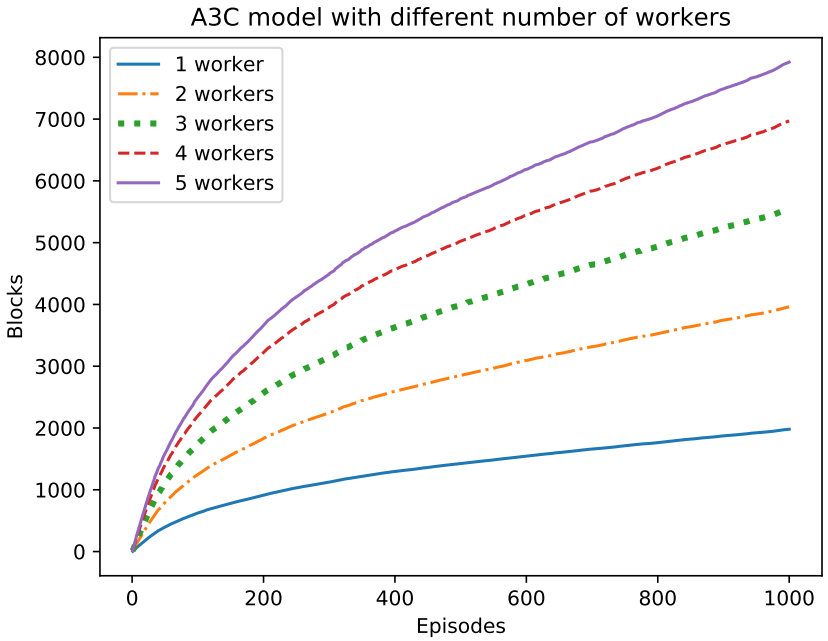

Figure 5.7: A3C model with different number of workers.

for different Deep Reinforcement Learning models where the A3C agent performs to obtain $8000 * 10^3$ transactions.

The A3C approach uses multiple agents that independently learn a policy from their environment and then collaborate with other agents to create global knowledge to choose the best decision. It is proven in the previous figures that the A3C model is the best compared to the other models.

Afterward, we studied the impact of the number of workers used asynchronously to learn an A3C agent to schedule tasks at different nodes. The number of workers is increased up to 5, as shown in figure 5.7. We observed that our A3C agent, with five workers, where each worker is assigned for each fog node to valid blocks, is the best to have a large number of blocks against others.

5.4.2.4 Throughput for SDN-BC system

The SDN controller must judge view modifications, access selection, and computing resource allocation to optimize performance, such as throughput. As previously stated,

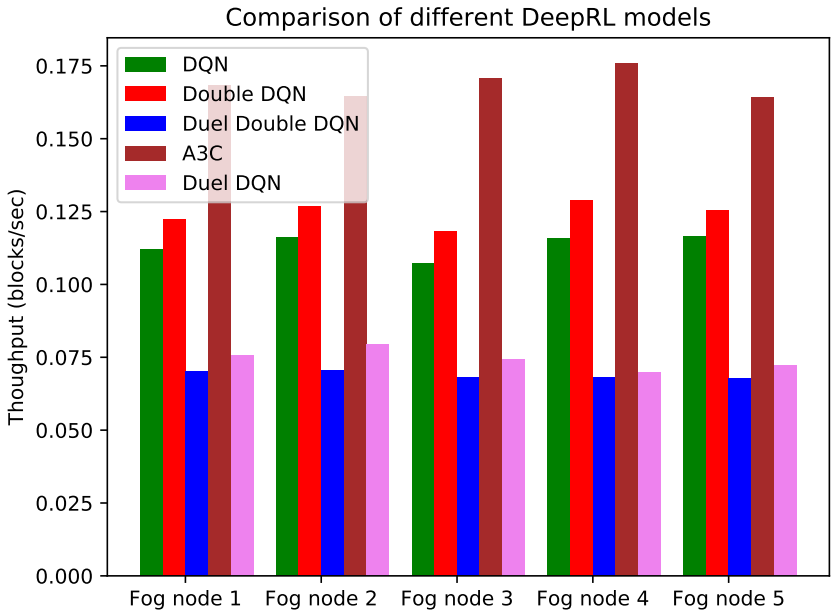

Figure 5.8: Throughput for different Deep Reinforcement Learning models.

we structure this joint problem as a Markov decision process in this part by defining state space, action space, and reward function. At time slot t, the learning agent must perceive state s(t) and make joint judgments regarding view updates, access selection, and computing resource allocation. As a result, the learning agent must know all fog node trust characteristics and edge computing node computational capabilities.

In this part, we study the throughput of our SDN-BC system during the learning of different DRL agents of $1,000$ training with $1,000$ tasks. Figure 5.8 presents the throughput in the different fog nodes for different Deep Reinforcement Learning mod-

els, calculated by mining blocks per second. As we can see from this figure, the A3C agent better out the other models where it can obtain 0.175 blocks/second.

5.4.2.5 Mining blocks time for SDN-BC system

Satoshi Nakamoto assured us that the complexity of mathematical issues would rise with time; mining one block takes 10 minutes and, in addition, the time your transaction is finished. Therefore, the time it takes a miner to mine a block in a proof of work method depends entirely on the mathematical problem's difficulty. However, Proof of Work requires computational resources at a critical time to mine the block, hence necessitating the investigation of alternative consensuses such as PBFT and PoA.

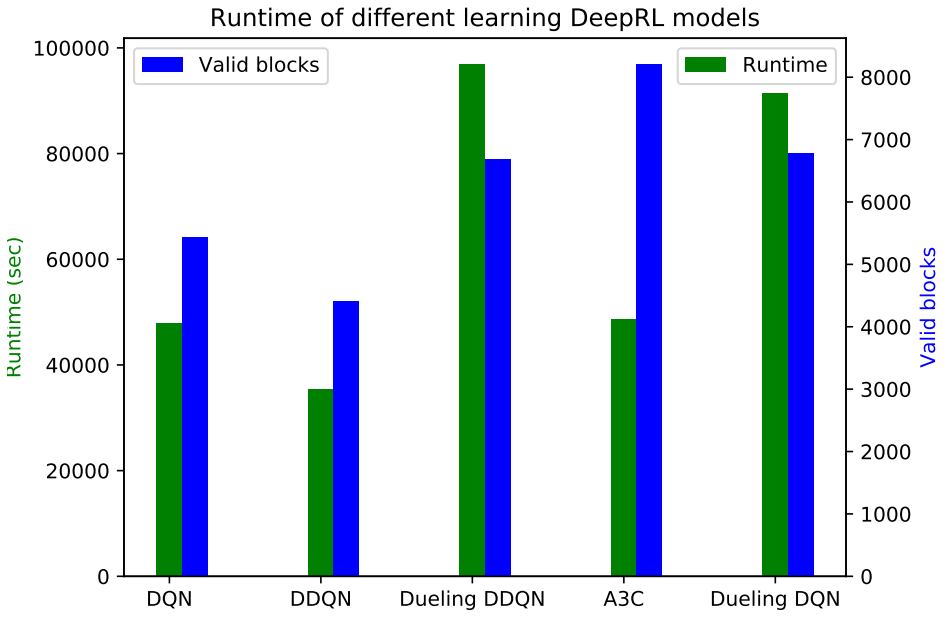

Figure 5.9: Runtime of different learning DRL models.

In our SDN-BC system, we studied the ability to process many block validations for the different DRL models in a minimum of time.

Figure 5.9 shows the runtime for learning to schedule tasks with different DRL models to mine blocks. It is a trade-off to make valid blocks with the runtime. We can see that the A3C agent is the better model to make almost $8,000$ blocks during nearly $50,000$ sec compared to other models.

In addition, figure 5.10 shows a comparative study between different DRL models about the time valid blocks during the phase of learning. It turned out that the A3C model is the best approach. The latter had an average of 8.3 seconds to verify and validate a block and associate it with the Blockchain than the other models.

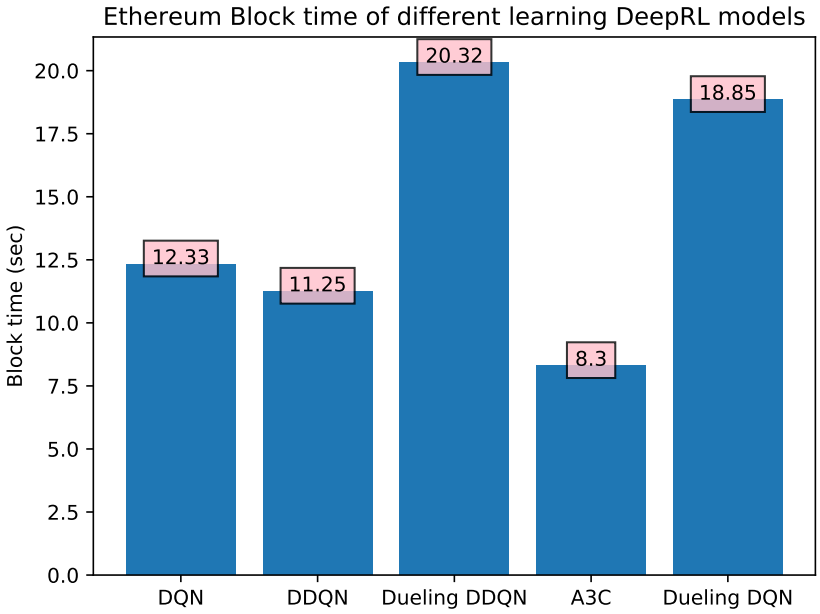

Figure 5.10: Block time of different learning DRL models.

5.4.2.6 Energy-Efficiency

One of the essential aspects of monitoring and optimizing the Blockchain-enabled SDN-IoT architecture is energy usage. Our solution beats many DRL models that employ the PoW or PBFT consensus methods, which do not account for the limits of IoT devices, which introduce overhead in energy usage. Using the PoA consensus to mine a block, we can efficiently decide and send packets through the A3C model for optimum scheduling tasks. Our strategy has been demonstrated to be effective in Figure 5.11.

We consider that batteries power all the nodes of our SDN-BC network, and we want to reduce the overall energy consumption of our system. Therefore, we trained an SDN controller agent per $1,000$ episodes that should be able to schedule $1,000$ tasks at low power fog nodes. Figure 5.11 illustrates the energy consumption of different DRL models that were mining blocks in our SDN-BC system. We can see that the A3C model allows keeping the total energy of our system against the other models.

Furthermore, to study the impact of PoW, PBFT, and PoA consensus on the energy efficiency of our system that we can see in the figure 5.12 shows the PoA consensus is better than the other consensus.

5.5 Conclusion

In this chapter, we have presented our contribution to developing a novel Blockchain-based that uses advanced deep learning mechanisms to support energy-aware task

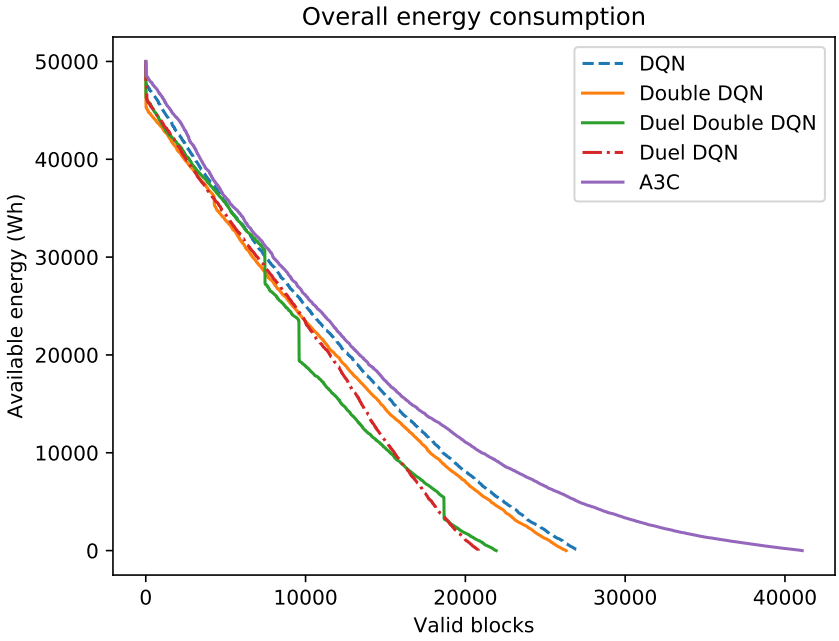

Figure 5.11: Overall energy consumption.

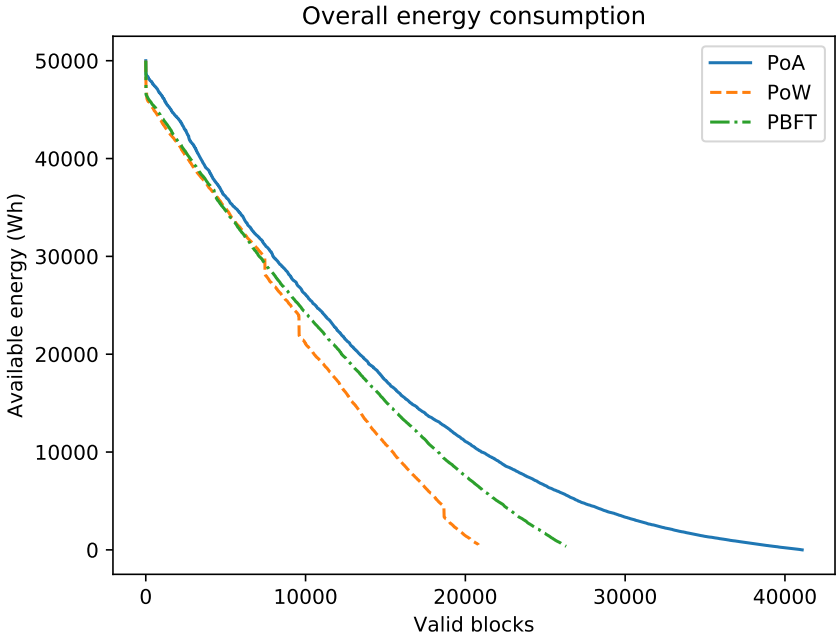

Figure 5.12: Overall energy consumption for different consensus.

scheduling and offloading in an SDN-enabled IoT network. The first part has been devoted to the design of our architecture and the proposed task offloading and scheduling approach, which considers battery-powered IoT devices. First, we introduced a three-layer architecture that included the edge network layer, data plane, blockchain layer, and SDN control plane layer. Then, we have demonstrated that blockchain technology

may be used to build a trustless task offloading scheme for SDN-enabled IoT networks. It delivers improved dependability and low latency while providing faster throughput and reduced network overhead using a Deep Reinforcement Learning algorithm. Next, we have developed a scheduling policy based on the Asynchronous Actor-Critic Agent (A3C) algorithm in symbiosis with an election-based consensus mechanism to validate IoT transactions, which provides high performance and fault tolerance while ensuring low latency and achieving energy efficiency. We proposed a task offloading scheme that incorporates a cycle-accurate energy consumption model, load-balanced, and energy-efficient design of IoT-powered edge devices to reduce energy consumption. Finally, the results of the experiments confirm that our approach that Combining A3C policy and PoA blockchain consensus offers 50% better energy efficiency than PoW and PBFT baseline consensus algorithms while ensuring lower network overhead and low latency.

Chapter 6

Conclusion and Perspectives

6.1 Conclusion

The main challenging issues addressed in this thesis concern the design of scalable, energy-efficient, low-latency, and lower network overhead IoT network edge architecture for smart and intelligent environments, such as smart buildings, smart cities, etc. Our goal was to reduce the energy impact of computing and communication infrastructures, while improving service quality, which is critical for lowering environmental effects and ensuring the infrastructure's long-term viability while transitioning to renewable energy sources. The energy effect of fog and the management of computing and communication resources in these infrastructures is a relatively new issue that has received little research.

These preliminary studies provide insight into the viability of fog solutions from an energy standpoint to enable IoT applications. IoT applications generate a great deal of complexity due to their heterogeneity (communication protocols, calculation requests, IoT objects, and hardware resources), distributed modeling into interdependent services, and various Quality of Service requirements. In addition, most IoT applications have high QoS requirements, mainly expressed as response time requirements, and necessitate near-instantaneous processing times.

To address this plethora of challenges, the contributions made by our research work have been presented in several stages and shown as follows:

1. In the first contribution, we proposed a comprehensive architectural design to empower SDN-enabled Context-Aware IoT systems and networks to create a flexible, agile, and reconfigurable framework to improve energy efficiency in smart buildings and enable automated building operations and control to implement an intelligent energy management system in smart micro-grids. First, we investigated many factors that influence energy consumption in campus buildings. Next, we created a context-awareness model that allows the most accurate prediction of users' actions based on their daily energy consumption profiles in

educational and residential facilities. Then, for designing and delivering customized IoT services on-demand, we launched an IoT service chaining solution. Finally, we included a machine learning engine that aids the context reasoning framework in inferring context judgments and feeding back modifications to the SDN controller for automated traffic steering and policy insertion.

2. In the second contribution, we demonstrated the possibility of establishing a job assignment and scheduling algorithms for SDN-enabled IoT networks by using Deep Reinforcement Learning. We devised a job assignment and scheduling problem that reduces network latency while maintaining energy economy. Our method beats deterministic placement algorithms, random placement algorithms, and A3C strategies in determining optimum allocation decision rules for task assignments and scheduling in real-time. Additionally, our method is optimized locally and globally, resulting in decreased communication latency and higher energy efficiency.

3. Finally, in the third contribution, we demonstrate the viability of using blockchain technology to build a trustless task offloading scheme for SDN-enabled IoT networks. Our solution approach delivers improved dependability and low latency while providing faster throughput and reduced network overhead using a Deep Reinforcement Learning algorithm. Furthermore, we proposed a three-layer architecture that included the edge network layer, data plane, blockchain layer, and SDN control plane layer in our approach. The suggested task offloading and scheduling approach considers battery-powered IoT devices, incorporating a cycle-accurate energy consumption model, load-balanced, and energy-efficient design of IoT-powered edge devices to reduce energy consumption.

6.2 Perspectives

The various works carried out in this thesis open the way to diverse research perspectives linked to the extension of the contributions mentioned above. These perspectives are in the short term. See immediate directly related to this work and which are in progress concerning the use of microservice architecture for highly deployable IoT and long-term perspectives that concern the interoperability and connectivity between heterogeneous blockchain platforms for secure IoT data sharing, as well as the integration and the adaptation of our work to other architectures.

The short-term perspectives' outlook is as follows:

- **Federated Learning for online task offloading and resource allocation:** This research work explained how the A3C approach and how the workers' agents

interact with their environments, collect experiences, receive rewards when exploring different IoT network states and send their model to the leading agent to perform global optimization. Then, the agent updates its model and synchronizes it with other distributed workers. However, these distributed agents do not share transitions but collect their own experiences. As a result, those transitions are often Not Independent and Identically Distributed (Non-IID) and are generally unbalanced. Federated Machine Learning (FedML) [180] is recently envisioned as a promising approach to solve the problem of the non-IID data in distributed massive IoT. Instead of sharing and disclosing the training dataset with the main worker agent or for A3C, the model hyperparameters are optimized collectively by substantial interconnected IoT devices that act as local learners. Additionally, FedML maintains privacy as data are not exported to third-party servers but remain inside the IoT devices. Thus, it enables aggregating and updating hyperparameters by leveraging the cooperation between massive IoT devices. Therefore, our future work will develop distributed intelligence using FedML to create an online task offloading and resource allocation with enormous IoT networks.

- **Cell-Based Microservices for Highly Deployable IoT:** This thesis pinpointed how blockchain verifier nodes could be deployed to the virtualization layer and a controller service abstraction layer using Kubernetes microservices through an Infrastructure as Code (IaC) model. At the same time, these microservices offer a global security approach chained as a single service to mitigate attacks. However, it is associated with the complexities of the distributed system and a higher chance of failure during communication. Cell-Based Microservices [181] are recently envisioned as a promising approach to building independently deployable components that can be loaded from multiple network locations and repositories at runtime. Furthermore, unifying and harmonizing these components will enable horizontal and vertical scalability and deployment independence. Thus, multiple federated IoT application components can run independently and be redeployed without restarting IoT applications or interrupting IoT services. Therefore, our future work will develop distributed Cell-Based microservices to ensure deployment independence and promise flexibility and autonomy. The sought-after goal is to guarantee modularity and cooperation among distributed federated IoT networks.

In the long term, the main prospects envisaged concern:

- **Enforcing IoT data integrity and privacy with homomorphic encryption:** In this research work, blockchain helped create distributed and immutable audit trail reviews of IoT transactions. Smart contracts are often open to the

public and perform only computations over plaintext data. They are made available on the Internet so that users can simply access them by selecting the correct URL. However, IoT data are analyzed over edge devices, which are usually more complex and sensitive to storage in the public blockchain (e.g., healthcare data). Security in Blockchain could be enhanced with Homomorphic encryption (HE) [182] to enable computation on encrypted data without leaking any information about the underlying data. HE performs calculations on encrypted data without decrypting them, and data are analyzed without breaching personal privacy. Therefore, our future work will integrate Blockchain transactions (i.e., local model parameters) with homomorphic encryption to secure data with high privacy in a decentralized model that enables high-performance data distribution and scalable communication between edge nodes, even in federated models to provide better privacy-preservation.

- **Interoperability and Connectivity between heterogeneous blockchains:** The architecture described in this work highlights the pivotal role of blockchain in enforcing the security and the efficiency of SDN-enabled IoT networks towards its integration in a large-scale massive and resilient 5G ecosystem. We argue that with the Industry 4.0 rocket, the considerable investment being made in blockchain platforms (e.g., Hyperledger, Ethereum, R3 Corda, Tezos, EOSIO, etc.), along with various collaborating companies– each focusing on a specific platform that fulfills their particular commercial tasks—makes it impossible to impose a single blockchain solution for multiple companies. It would be more exciting to keep these participants having their data and management system, while creating bridges between blockchains to ensure interoperability among them. We believe that the development of Oracle blockchain [183] to format messages for communication, not only between heterogeneous blockchain platforms but also with the existing enterprise management system, can offer interoperability between organizations and networks. For example, transactional activities on one blockchain can be used as inputs to trigger the execution of another processing on a different blockchain. Therefore, our future work looms to consider the development of blockchain Oracle to offer bridges for cross-ledgers interoperability, enable a wide range of cross-communication functions, and support diverse security and data structuring capabilities to avoid corrupt, malicious, or inaccurate data.

Chapter 7

Author Biography

Publications

Journals

1. **Bassem Sellami**, Akram Hakiri, Sadok Ben Yahia, Pascal Berthou. *Energy-aware task scheduling and offloading using deep reinforcement learning in SDN-enabled IoT network* [205], Computer Networks, Volume 210, 2022, 108957, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2022.108957

2. **Bassem Sellami**, Akram Hakiri, and Sadok Ben Yahia. "*Deep Reinforcement Learning for Energy-Aware Task Offloading in Join SDN-Blockchain 5G massive IoT Edge Network*" [206]. Elsevier Journal of Future Generation Computer Systems (August, 1st, 2022). https://doi.org/10.1016/j.future.2022.07.024

Conferences

1. **B. Sellami**, A. Hakiri, S. Ben Yahia and P. Berthou, "*Deep Reinforcement Learning for Energy-Efficient Task Scheduling in SDN-based IoT Network*" [207], 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), 2020, pp. 1-4, https://doi.org/10.1109/NCA51143.2020.9306739.

2. A. Hakiri, **B. Sellami**, S. Ben Yahia and P. Berthou, "*A SDN-based IoT Architecture Framework for Efficient Energy Management in Smart Buildings*" [208], 2020 Global Information Infrastructure and Networking Symposium (GIIS), 2020, pp. 1-6, https://doi.org/10.1109/GIIS50753.2020.9248495.

3. A. Hakiri, **B. Sellami**, S. Ben Yahia and P. Berthou, "*A Blockchain Architecture for SDN-enabled Tamper-Resistant IoT Networks*" [209], 2020 Global

Information Infrastructure and Networking Symposium (GIIS), 2020, pp. 1-4, https://doi.org/10.1109/GIIS50753.2020.9248492.

4. A. Hakiri, **B. Sellami**, P. Patil, P. Berthou and A. Gokhale, "*Managing Wireless Fog Networks using Software-Defined Networking*" [210], 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), 2017, pp. 1149-1156, https://doi.org/10.1109/AICCSA.2017.9.

Workshop

1. **B. Sellami**, Abdelaziz El Fazziki, Mohamed Taha Bennani, and S. Ben Yahia. *Smart Irrigation Management System.* 2022 16th International Conference on Signal Image Technology & Internet based Systems. Dijon, France (Hybrid meeting, remote or presence) – October 19-21, 2022.

2. Hakiri, A., **Sallemi, B.**, Ghandour, F., Ben Yahia, S. *Secure, Context-Aware and QoS-Enabled SDN Architecture to Improve Energy Efficiency in IoT-Based Smart Buildings* [211]. In: Jemili, I., Mosbah, M. (eds) Distributed Computing for Emerging Smart Networks. DiCES-N 2020. Communications in Computer and Information Science, vol 1348. Springer, Cham. https://doi.org/10.1007/978-3-030-65810-6_4

Peer Review Recognition

Peer-reviewed International Journal Papers

1. International Journal of General Systems: IJGS (Print ISSN: 0308-1079 Online ISSN: 1563-5104)

2. Concurrency Computation Practice and Experience Journal (Print ISSN: 15320626, Online ISSN: 15320634)

Peer-reviewed International Conferences and Program Committee

1. 1st BAIC Workshop on Blockchain and AI for Community, which will be held in conjunction with 14th ACM Web Science Conference, 26 – 29, June 2022.

2. 26th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES 2022), 7-9 September 2022.

3. The 33rd International Conference on Database and Expert Systems Applications (DEXA 2022), 22 - 24 August 2022.

Participation in Research Projects

Current Projects

- **IPERCITIES** : Plateforme d'expérimentation IoT Pour l'Education et la Recherche pour la Création de smart cITIES en Tunisie. Programme d'Encouragement des Jeunes Chercheurs (PEJC). Tunisian Ministry of Higher Education. Jan 2019 - Dec 2021. ISSAT Mateur
- **H2020. NGI Atlantic Project**. HyPer-5G: Hyper Performance Digital Twins for Holistic Management of Resilient 5G Edge Networks.
- **Project SIGIRO** : Un système intelligent pour la gouvernance par une gestion intégrée des ressources en eau : Cas de la région de Marrakech-Safi. FST-Maroc.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[3] Ericsson, "Iot connections outlook," 2021.

[4] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, 2012.

[6] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46–59, 2015.

[7] C. Dsouza, G.-J. Ahn, and M. Taguinod, "Policy-driven security management for fog computing: Preliminary framework and a case study," in *Proceedings of the 2014 IEEE 15th international conference on information reuse and integration (IEEE IRI 2014)*, pp. 16–23, IEEE, 2014.

[8] C. C. Byers and P. Wetterwald, "Fog computing distributing data and intelligence for resiliency and scale necessary for iot: The internet of things (ubiquity symposium)," *Ubiquity*, vol. 2015, no. November, pp. 1–12, 2015.

[9] H. Hejazi, H. Rajab, T. Cinkler, and L. Lengyel, "Survey of platforms for massive iot," in *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*, pp. 1–8, IEEE, 2018.

[10] 6gworld, "Sustainability in new and emerging technologies," 2021.

[11] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," *Computer Networks*, vol. 75, pp. 453–471, 2014.

[12] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, "A cloud–mec collaborative task offloading scheme with service orchestration," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5792–5805, 2019.

[13] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[14] B. Yi, X. Wang, K. Li, S. k. Das, and M. Huang, "A comprehensive survey of network function virtualization," *Computer Networks*, vol. 133, pp. 212–262, 2018.

[15] P. Amangele, M. J. Reed, M. Al-Naday, N. Thomos, and M. Nowak, "Hierarchical machine learning for iot anomaly detection in sdn," in *2019 International Conference on Information Technologies (InfoTech)*, pp. 1–4, IEEE, 2019.

[16] A. Abdou, P. C. Van Oorschot, and T. Wan, "Comparative analysis of control plane security of sdn and conventional networks," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3542–3559, 2018.

[17] X. Wang, X. Li, S. Pack, Z. Han, and V. C. Leung, "Stcs: Spatial-temporal collaborative sampling in flow-aware software defined networks," *ieee journal on selected areas in communications*, vol. 38, no. 6, pp. 999–1013, 2020.

[18] A. Rahman, M. J. Islam, Z. Rahman, M. M. Reza, A. Anwar, M. P. Mahmud, M. K. Nasir, and R. M. Noor, "Distb-condo: Distributed blockchain-based iot-sdn model for smart condominium," *IEEE Access*, vol. 8, pp. 209594–209609, 2020.

[19] N. Kshetri, "Can blockchain strengthen the internet of things?," *IT professional*, vol. 19, no. 4, pp. 68–72, 2017.

[20] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *Ieee Access*, vol. 4, pp. 2292–2303, 2016.

[21] M. Conoscenti, A. Vetro, and J. C. De Martin, "Blockchain for the internet of things: A systematic literature review," in *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pp. 1–6, IEEE, 2016.

[22] X. Zhang and X. Chen, "Data security sharing and storage based on a consortium blockchain in a vehicular ad-hoc network," *IEEE Access*, vol. 7, pp. 58241–58254, 2019.

[23] J. Gao, K. O.-B. O. Agyekum, E. B. Sifah, K. N. Acheampong, Q. Xia, X. Du, M. Guizani, and H. Xia, "A blockchain-sdn-enabled internet of vehicles environment for fog computing and 5g networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4278–4291, 2019.

[24] G. C. Polyzos and N. Fotiou, "Blockchain-assisted information distribution for the internet of things," in *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, pp. 75–78, IEEE, 2017.

[25] F. Buccafurri, G. Lax, S. Nicolazzo, and A. Nocera, "Overcoming limits of blockchain for iot applications," in *Proceedings of the 12th international conference on availability, reliability and security*, pp. 1–6, 2017.

[26] L. Jiang, L. Da Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu, "An iot-oriented data storage framework in cloud computing platform," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1443–1451, 2014.

[27] J. E. Ibarra-Esquer, F. F. González-Navarro, B. L. Flores-Rios, L. Burtseva, and M. A. Astorga-Vargas, "Tracking the evolution of the internet of things concept across different application domains," *Sensors*, vol. 17, no. 6, p. 1379, 2017.

[28] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, pp. 20–26, IEEE, 2016.

[29] K. Bilal, O. Khalid, A. Erbad, and S. U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Computer Networks*, vol. 130, pp. 94–120, 2018.

[30] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: multiaccess edge computing for 5g and internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6722–6747, 2020.

[31] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–36, 2019.

[32] M. Etsi, "Mobile-edge computing," *Introductory Technical White Paper*, 2014.

[33] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: Challenges and solutions," *Journal of cleaner production*, vol. 140, pp. 1454–1464, 2017.

[34] C. Vallati, A. Virdis, E. Mingozzi, and G. Stea, "Mobile-edge computing come home connecting things in future smart homes using lte device-to-device communications," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 77–83, 2016.

[35] M. Liyanage, P. Porambage, A. Y. Ding, and A. Kalla, "Driving forces for multi-access edge computing (mec) iot integration in 5g," *ICT Express*, vol. 7, no. 2, pp. 127–137, 2021.

[36] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of mec in the internet of things," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 84–91, 2016.

[37] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, 2017.

[38] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[39] O. Zakaria, J. Britt, and H. Forood, "Internet of things (iot) automotive device, system, and method," jul 2017. US Patent 9,717,012.

[40] W. Balid, H. Tafish, and H. H. Refai, "Intelligent vehicle counting and classification sensor for real-time traffic surveillance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1784–1794, 2017.

[41] S. K. Datta, J. Haerri, C. Bonnet, and R. F. Da Costa, "Vehicles as connected resources: Opportunities and challenges for the future," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 26–35, 2017.

[42] H. Sun, Z. Zhang, R. Q. Hu, and Y. Qian, "Challenges and enabling technologies in 5g wearable communications," *arXiv preprint arXiv*, vol. 1708, 2017.

[43] M. H. Yaghmaee, A. Leon-Garcia, and M. Moghaddassian, "On the performance of distributed and cloud-based demand response in smart grid," *IEEE transactions on smart grid*, vol. 9, no. 5, pp. 5403–5417, 2017.

[44] M. Satyanarayanan, "Edge computing.," *Computer*, vol. 50, no. 10, pp. 36–38, 2017.

[45] A. L. Virk, M. A. Noor, S. Fiaz, S. Hussain, H. A. Hussain, M. Rehman, M. Ahsan, and W. Ma, "Smart farming: an overview," *Smart Village Technology*, pp. 191–201, 2020.

[46] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for internet of things realization," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2961–2991, 2018.

[47] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[48] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[49] D. Mondal, "The internet of thing (iot) and industrial automation: a future perspective," *World Journal of Modelling and Simulation*, vol. 15, no. 2, pp. 140–149, 2019.

[50] W. Steiner and S. Poledna, "Fog computing as enabler for the industrial internet of things," *e & i Elektrotechnik und Informationstechnik*, vol. 133, no. 7, pp. 310–314, 2016.

[51] Y. Chen, Y. Zhang, S. Maharjan, M. Alam, and T. Wu, "Deep learning for secure mobile edge computing in cyber-physical transportation systems," *IEEE Network*, vol. 33, no. 4, pp. 36–41, 2019.

[52] D. Milojicic, "The edge-to-cloud continuum," *Computer*, vol. 53, no. 11, pp. 16–25, 2020.

[53] V. Millnert, J. Eker, and E. Bini, "End-to-end deadlines over dynamic topologies," in *31st Euromicro Conference on Real-Time Systems (ECRTS 2019)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[54] J. Guo, Z. Song, Y. Cui, Z. Liu, and Y. Ji, "Energy-efficient resource allocation for multi-user mobile edge computing," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–7, IEEE, 2017.

[55] J.-W. Kao, Y.-Y. Shih, A.-C. Pang, and Y.-C. Lin, "Radio resource allocation for d2d-assisted full-duplex cellular networks," in *2015 Seventh International Conference on Ubiquitous and Future Networks*, pp. 721–726, IEEE, 2015.

[56] Z. Ding, Y.-C. Tian, M. Tang, Y. Li, Y.-G. Wang, and C. Zhou, "Profile-guided three-phase virtual resource management for energy efficiency of data centers," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 3, pp. 2460–2468, 2019.

[57] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3056–3069, 2017.

[58] H. Dai, Y. Huang, Y. Xu, C. Li, B. Wang, and L. Yang, "Energy-efficient resource allocation for energy harvesting-based device-to-device communication," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 509–524, 2018.

[59] S. Sundar and B. Liang, "Offloading dependent tasks with communication delay and deadline constraint," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 37–45, IEEE, 2018.

[60] F. Wang, Y. Li, Z. Wang, and Z. Yang, "Social-community-aware resource allocation for d2d communications underlaying cellular networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 5, pp. 3628–3640, 2015.

[61] B. Yin, Y. Cheng, L. X. Cai, and X. Cao, "Online sla-aware multi-resource allocation for deadline sensitive jobs in edge-clouds," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2017.

[62] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power iot edge devices," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 7–12, IEEE, 2016.

[63] G. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, and J. Root, "Tactical cloudlets: Moving cloud computing to the edge," in *2014 IEEE military communications conference*, pp. 1440–1446, IEEE, 2014.

[64] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2017.

[65] P. Patil, A. Hakiri, and A. Gokhale, "Cyber foraging and offloading framework for internet of things," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, pp. 359–368, IEEE, 2016.

[66] C.-W. Tsai, C.-F. Lai, H.-C. Chao, and A. V. Vasilakos, "Big data analytics: a survey," *Journal of Big data*, vol. 2, no. 1, pp. 1–32, 2015.

[67] N. A. Angel, D. Ravindran, P. D. R. Vincent, K. Srinivasan, and Y.-C. Hu, "Recent advances in evolving computing paradigms: Cloud, edge, and fog technologies," *Sensors*, vol. 22, no. 1, p. 196, 2021.

[68] E. Badidi, Z. Mahrez, and E. Sabir, "Fog computing for smart cities' big data management and analytics: A review," *Future Internet*, vol. 12, no. 11, p. 190, 2020.

[69] C. Mwase, Y. Jin, T. Westerlund, H. Tenhunen, and Z. Zou, "Communication-efficient distributed ai strategies for the iot edge," *Future Generation Computer Systems*, 2022.

[70] F. Samie, L. Bauer, and J. Henkel, "From cloud down to things: An overview of machine learning in internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4921–4934, 2019.

[71] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM transactions on networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[72] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[73] C. Moratelli, S. Johann, M. Neves, and F. Hessel, "Embedded virtualization for the design of secure iot applications," in *2016 International Symposium on Rapid System Prototyping (RSP)*, pp. 1–5, IEEE, 2016.

[74] H. Li, X. Xu, J. Ren, and Y. Dong, "Acrn: A big little hypervisor for iot development," in *Proceedings of the 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE 2019, (New York, NY, USA), p. 31–44, Association for Computing Machinery, 2019.

[75] Y. Li and H. Takada, "isotee: A hypervisor middleware for iot-enabled resource-constrained reliable systems," *IEEE Access*, vol. 10, pp. 8566–8576, 2022.

[76] A. Celesti, D. Mulfari, M. Fazio, M. Villari, and A. Puliafito, "Exploring container virtualization in iot clouds," in *2016 IEEE international conference on Smart Computing (SMARTCOMP)*, pp. 1–6, IEEE, 2016.

[77] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[78] K. Sood, S. Yu, and Y. Xiang, "Software-defined wireless networking opportunities and challenges for internet-of-things: A review," *IEEE Internet of Things Journal*, vol. 3, no. 4, pp. 453–463, 2015.

[79] W. Braun and M. Menth, "Software-defined networking using openflow: Protocols, applications and architectural design choices," *Future Internet*, vol. 6, no. 2, pp. 302–336, 2014.

[80] M.-K. Shin, K.-H. Nam, and H.-J. Kim, "Software-defined networking (sdn): A reference architecture and open apis," in *2012 International Conference on ICT Convergence (ICTC)*, pp. 360–361, IEEE, 2012.

[81] S. Schmid and J. Suomela, "Exploiting locality in distributed sdn control," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 121–126, 2013.

[82] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, p. 69–74, mar 2008.

[83] J. Kundrat, J. Vojtěch, P. Škoda, R. Vohnout, J. Radil, and O. Havliš, "Yang/netconf roadm: Evolving open dwdm toward sdn applications," *Journal of Lightwave Technology*, vol. 36, no. 15, pp. 3105–3114, 2018.

[84] A. A. Khan, M. Abolhasan, W. Ni, J. Lipman, and A. Jamalipour, "An end-to-end (e2e) network slicing framework for 5g vehicular ad-hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 7, pp. 7103–7112, 2021.

[85] G. ETSI, "Network functions virtualisation (nfv): Architectural framework," *ETsI Gs NFV*, vol. 2, no. 2, p. V1, 2013.

[86] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," *Computer Networks*, vol. 75, pp. 453–471, 2014.

[87] B. Chatras, U. S. T. Kwong, and N. Bihannic, "Nfv enabling network slicing for 5g," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, pp. 219–225, IEEE, 2017.

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

[89] J. V. Ramrao and A. Jain, "Dynamic 5g network slicing," *International Journal*, vol. 10, no. 2, 2021.

[90] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research directions in

network service chaining," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, pp. 1–7, 2013.

[91] N. ETSI *et al.*, "Network functions virtualisation (nfv); management and orchestration," *NFV-MAN*, vol. 1, p. v0, 2014.

[92] A. J. Gonzalez, G. Nencioni, A. Kamisiński, B. E. Helvik, and P. E. Heegaard, "Dependability of the nfv orchestrator: State of the art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3307–3329, 2018.

[93] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques, and open issues," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1206–1243, 2018.

[94] I. Sarrigiannis, E. Kartsakli, K. Ramantas, A. Antonopoulos, and C. Verikoukis, "Application and network vnf migration in a mec-enabled 5g architecture," in *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 1–6, IEEE, 2018.

[95] B. Martini, F. Paganelli, A. Mohammed, M. Gharbaoui, A. Sgambelluri, and P. Castoldi, "Sdn controller for context-aware data delivery in dynamic service chaining," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pp. 1–5, IEEE, 2015.

[96] A. H. Shamsan and A. R. Faridi, "Network softwarization for iot: A survey," in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 1163–1168, IEEE, 2019.

[97] A. Molina Zarca, D. Garcia-Carrillo, J. Bernal Bernabe, J. Ortiz, R. Marin-Perez, and A. Skarmeta, "Enabling virtual aaa management in sdn-based iot networks," *Sensors*, vol. 19, no. 2, p. 295, 2019.

[98] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[99] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for iot big data and streaming analytics: A survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.

[100] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," *Advances in neural information processing systems*, vol. 29, 2016.

[101] G. E. Hinton, "A practical guide to training restricted boltzmann machines," in *Neural networks: Tricks of the trade*, pp. 599–619, Springer, 2012.

[102] N. Zhang, S. Ding, J. Zhang, and Y. Xue, "An overview on restricted boltzmann machines," *Neurocomputing*, vol. 275, pp. 1186–1199, 2018.

[103] P. Sharma, S. Jain, S. Gupta, and V. Chamola, "Role of machine learning and deep learning in securing 5g-driven industrial iot applications," *Ad Hoc Networks*, vol. 123, p. 102685, 2021.

[104] A. Elsaeidy, K. S. Munasinghe, D. Sharma, and A. Jamalipour, "Intrusion detection in smart cities using restricted boltzmann machines," *Journal of Network and Computer Applications*, vol. 135, pp. 76–83, 2019.

[105] S. Hwang, J. Jeong, and Y. Kang, "Svm-rbm based predictive maintenance scheme for iot-enabled smart factory," in *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, pp. 162–167, 2018.

[106] X. Yang, Z. Wu, and Q. Zhang, "Bluetooth indoor localization with gaussian–bernoulli restricted boltzmann machine plus liquid state machine," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–8, 2022.

[107] S. J. Desai, M. Shoaib, and A. Raychowdhury, "An ultra-low power, "always-on" camera front-end for posture detection in body worn cameras using restricted boltzman machines," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 1, no. 4, pp. 187–194, 2015.

[108] H.-G. Kim, S.-H. Han, and H.-J. Choi, "Discriminative restricted boltzmann machine for emergency detection on healthcare robot," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 407–409, 2017.

[109] M. Tschannen, O. Bachem, and M. Lucic, "Recent advances in autoencoder-based representation learning," *arXiv preprint arXiv:1812.05069*, 2018.

[110] R. M. Cichy and D. Kaiser, "Deep neural networks as scientific models," *Trends in cognitive sciences*, vol. 23, no. 4, pp. 305–317, 2019.

[111] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*, pp. 421–436, Springer, 2012.

[112] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, "Backpropagation and the brain," *Nature Reviews Neuroscience*, vol. 21, no. 6, pp. 335–346, 2020.

[113] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014.

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

[115] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep q-learning," in *Learning for Dynamics and Control*, pp. 486–489, PMLR, 2020.

[116] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*, pp. 1995–2003, PMLR, 2016.

[117] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, PMLR, 2016.

[118] F. Cicirelli, A. Guerrieri, G. Spezzano, and A. Vinci, "An edge-based platform for dynamic smart city applications," *Future Generation Computer Systems*, vol. 76, pp. 106–118, 2017.

[119] S. Bhattacharya, S. R. K. Somayaji, T. R. Gadekallu, M. Alazab, and P. K. R. Maddikunta, "A review on deep learning for future smart cities," *Internet Technology Letters*, vol. 5, no. 1, p. e187, 2022.

[120] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[121] Y. Liu, C. Yang, L. Jiang, S. Xie, and Y. Zhang, "Intelligent edge computing for iot-based energy management in smart cities," *IEEE network*, vol. 33, no. 2, pp. 111–117, 2019.

[122] K.-D. Thoben, S. Wiesner, and T. Wuest, ""industrie 4.0" and smart manufacturing-a review of research issues and application examples," *International journal of automation technology*, vol. 11, no. 1, pp. 4–16, 2017.

[123] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *Journal of Manufacturing Systems*, vol. 48, pp. 144–156, 2018. Special Issue on Smart Manufacturing.

[124] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

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

[126] A. Nordrum, "Wall street occupies the blockchain - financial firms plan to move trillions in assets to blockchains in 2018," *IEEE Spectrum*, vol. 54, pp. 40–45, oct 2017.

[127] A. Nordrum, "Govern by blockchain dubai wants one platform to rule them all, while illinois will try anything," *IEEE Spectrum*, vol. 54, pp. 54–55, oct 2017.

[128] V. Y. Kemmoe, W. Stone, J. Kim, D. Kim, and J. Son, "Recent advances in smart contracts: A technical overview and state of the art," *IEEE Access*, vol. 8, pp. 117782–117801, 2020.

[129] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE international congress on big data (BigData congress)*, pp. 557–564, Ieee, 2017.

[130] P. Tasca and C. J. Tessone, "Taxonomy of blockchain technologies. principles of identification and classification," *arXiv preprint arXiv:1708.04872*, 2017.

[131] E. Nyaletey, R. M. Parizi, Q. Zhang, and K.-K. R. Choo, "Blockipfs - blockchain-enabled interplanetary file system for forensic and trusted data traceability," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 18–25, 2019.

[132] R. Kumar and R. Tripathi, "Chapter 2 - blockchain-based framework for data storage in peer-to-peer scheme using interplanetary file system," in *Handbook of Research on Blockchain Technology* (S. Krishnan, V. E. Balas, E. G. Julie, Y. H. Robinson, S. Balaji, and R. Kumar, eds.), pp. 35–59, Academic Press, 2020.

[133] X. Tao, M. Das, Y. Liu, and J. C. Cheng, "Distributed common data environment using blockchain and interplanetary file system for secure bim-based collaborative design," *Automation in Construction*, vol. 130, p. 103851, 2021.

[134] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019.

[135] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.

[136] M. Bartoletti and L. Pompianu, "An Empirical Analysis Of Smart Contracts: Platforms, Applications, And Design Patterns," in *Financial Cryptography Workshops*, 2017.

[137] K. Christidis and M. Devetsikiotis, "Blockchains And Smart Contracts For The Internet Of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[138] S. Raval, *Decentralized applications: harnessing Bitcoin's blockchain technology.* " O'Reilly Media, Inc.", 2016.

[139] N. V. Keizer, F. Yang, I. Psaras, and G. Pavlou, "The case for ai based web3 reputation systems," in *2021 IFIP Networking Conference (IFIP Networking)*, pp. 1–2, 2021.

[140] Z. Liu, Y. Xiang, J. Shi, P. Gao, H. Wang, X. Xiao, B. Wen, Q. Li, and Y.-C. Hu, "Make web3.0 connected," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2021.

[141] A. Hafid, A. S. Hafid, and M. Samih, "Scaling blockchains: A comprehensive survey," *IEEE Access*, vol. 8, pp. 125244–125262, 2020.

[142] M. Tahir, M. H. Habaebi, M. Dabbagh, A. Mughees, A. Ahad, and K. I. Ahmed, "A review on application of blockchain in 5g and beyond networks: Taxonomy, field-trials, challenges and opportunities," *IEEE Access*, vol. 8, pp. 115876–115904, 2020.

[143] E. Bellini, Y. Iraqi, and E. Damiani, "Blockchain-based distributed trust and reputation management systems: A survey," *IEEE Access*, vol. 8, pp. 21127–21151, 2020.

[144] S. Khan, M. B. Amin, A. T. Azar, and S. Aslam, "Towards interoperable blockchains: A survey on the role of smart contracts in blockchain interoperability," *IEEE Access*, vol. 9, pp. 116672–116691, 2021.

[145] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *IEEE Access*, vol. 7, pp. 117134–117151, 2019.

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

[147] I.-C. Lin and T.-C. Liao, "A survey of blockchain security issues and challenges.," *Int. J. Netw. Secur.*, vol. 19, no. 5, pp. 653–659, 2017.

[148] D. D.Vujicic and S.Randic, "Blockchain technology, bitcoin, and ethereum: A brief overview," in *2018 17th international symposium infoteh-jahorina (infoteh)*, pp. 1–6, IEEE, 2018.

[149] J. Sun, J. Yan, and K. Z. Zhang, "Blockchain-based sharing services: What blockchain technology can contribute to smart cities," *Financial Innovation*, vol. 2, no. 1, pp. 1–9, 2016.

[150] B.-K. Zheng, L.-H. Zhu, M. Shen, F. Gao, C. Zhang, Y.-D. Li, and J. Yang, "Scalable and privacy-preserving data sharing based on blockchain," *Journal of Computer Science and Technology*, vol. 33, no. 3, pp. 557–567, 2018.

[151] K. Chen, S. Zhang, Z. Li, Y. Zhang, Q. Deng, S. Ray, and Y. Jin, "Internet-of-things security and vulnerabilities: Taxonomy, challenges, and practice," *Journal of Hardware and Systems Security*, vol. 2, no. 2, pp. 97–110, 2018.

[152] H. R. Bokkisam, S. Singh, R. M. Acharya, and M. P. Selvan, "Blockchain-based peer-to-peer transactive energy system for community microgrid with demand response management," *CSEE Journal of Power and Energy Systems*, vol. 8, no. 1, pp. 198–211, 2022.

[153] D. Kundu, "Blockchain and trust in a smart city," *Environment and Urbanization ASIA*, vol. 10, no. 1, pp. 31–43, 2019.

[154] D. Bruneo, F. Longo, G. Merlino, A. Puliafito, and N. Kushwaha, "Integrating iot and cloud in a smart city context: the# smartme case study," *International Journal of Computer Applications in Technology*, vol. 57, no. 4, pp. 267–280, 2018.

[155] K. Wüst and A. Gervais, "Do you need a blockchain?," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pp. 45–54, IEEE, 2018.

[156] M. M. Aung and Y. S. Chang, "Traceability in a food supply chain: Safety and quality perspectives," *Food control*, vol. 39, pp. 172–184, 2014.

[157] A. Mittal, B. Sharma, and P. Ranjan, "Real estate management system based on blockchain," in *2020 IEEE 7th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, pp. 1–6, 2020.

[158] S. Latifi, Y. Zhang, and L.-C. Cheng, "Blockchain-based real estate market: One method for applying blockchain technology in commercial real estate market," in *2019 IEEE International Conference on Blockchain (Blockchain)*, pp. 528–535, 2019.

[159] I. Karamitsos, M. Papadaki, N. B. Al Barghuthi, *et al.*, "Design of the blockchain smart contract: A use case for real estate," *Journal of Information Security*, vol. 9, no. 03, p. 177, 2018.

[160] A. Tobin and D. Reed, "The inevitable rise of self-sovereign identity," *The Sovrin Foundation*, vol. 29, no. 2016, 2016.

[161] N. Naik and P. Jenkins, "uport open-source identity management system: An assessment of self-sovereign identity and user-centric data platform built on blockchain," in *2020 IEEE International Symposium on Systems Engineering (ISSE)*, pp. 1–7, 2020.

[162] R. Soltani, U. T. Nguyen, and A. An, "A new approach to client onboarding using self-sovereign identity and distributed ledger," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1129–1136, IEEE, 2018.

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

[164] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, 2014.

[165] E. F. Kfoury and D. J. Khoury, "Secure end-to-end volte based on ethereum blockchain," in *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, pp. 1–5, IEEE, 2018.

[166] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of consensus protocols on blockchain applications," in *2017 4th international conference on advanced computing and communication systems (ICACCS)*, pp. 1–5, IEEE, 2017.

[167] A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat, and S. Chatterjee, "Performance characterization of hyperledger fabric," in *2018 Crypto Valley conference on blockchain technology (CVCBT)*, pp. 65–74, IEEE, 2018.

[168] M. Hearn and R. G. Brown, "Corda: A distributed ledger," *Corda Technical White Paper*, vol. 2016, 2016.

[169] R. G. Brown, J. Carlyle, I. Grigg, and M. Hearn, "Corda: an introduction," *R3 CEV, August*, vol. 1, no. 15, p. 14, 2016.

[170] S. Rouhani and R. Deters, "Performance analysis of ethereum transactions in private blockchain," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 70–74, IEEE, 2017.

[171] M. Kaleem, A. Mavridou, and A. Laszka, "Vyper: A security comparison with solidity based on common vulnerabilities," in *2020 2nd Conference on Blockchain Research Applications for Innovative Networks and Services (BRAINS)*, pp. 107–111, 2020.

[172] O. N. F. Open Network Foundation, "OpenFlow Switch Specification," April 2013.

[173] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, (New York, NY, USA), p. 6, Association for Computing Machinery, 2010.

[174] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[175] R. project team, *RYU SDN FRAMEWORK*. FIXME, February 2014.

[176] J. Wang, C. Xu, Y. Huangfu, R. Li, Y. Ge, and J. Wang, "Deep reinforcement learning for scheduling in cellular networks," in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–6, IEEE, 2019.

[177] D. Fakhri and K. Mutijarsa, "Secure iot communication using blockchain technology," in *2018 International Symposium on Electronics and Smart Devices (ISESD)*, pp. 1–6, IEEE, 2018.

[178] S. Joshi, "Feasibility of proof of authority as a consensus protocol model," *arXiv preprint arXiv:2109.02480*, 2021.

[179] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project adam: Building an efficient and scalable deep learning training system," in *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pp. 571–582, 2014.

[180] L. Zang, X. Zhang, and B. Guo, "Federated deep reinforcement learning for online task offloading and resource allocation in wpc-mec networks," *IEEE Access*, vol. 10, pp. 9856–9867, 2022.

[181] "Cell-based architecture: A decentralized reference architecture for cloud-native applications," 2021.

[182] R. Shrestha and S. Kim, "Chapter ten - integration of iot with blockchain and homomorphic encryption: Challenging issues and opportunities," in *Role of Blockchain Technology in IoT Applications* (S. Kim, G. C. Deka, and P. Zhang, eds.), vol. 115 of *Advances in Computers*, pp. 293–331, Elsevier, 2019.

[183] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic, "Trustworthy blockchain oracles: Review, comparison, and open research challenges," *IEEE Access*, vol. 8, pp. 85675–85685, 2020.

[184] M. Kaushik, R. Sharma, S. A. Peious, M. Shahin, S. Ben Yahia, and D. Draheim, "A systematic assessment of numerical association rule mining methods," *SN Computer Science*, vol. 2, no. 5, p. 348, 2021.

[185] M. Kachroudi, G. Diallo, and S. Ben Yahia, "KEPLER at OAEI 2018," *OM@ ISWC*, vol. 2018, pp. 173–178, 2018.

[186] A. Mouakher and S. Ben Yahia, "On the efficient stability computation for the selection of interesting formal concepts," *Information Sciences*, vol. 472, pp. 15–34, 2019.

[187] A. Houari, W. Ayadi, and S. Ben Yahia, "A new FCA-based method for identifying biclusters in gene expression data," *International Journal of Machine Learning and Cybernetics*, vol. 9, pp. 1879–1893, 2018.

[188] S. Ben Chaabene, T. Yeferny, and S. Ben Yahia, "A roadside unit placement scheme for vehicular ad-hoc networks," in *Advanced Information Networking and Applications: Proceedings of the 33rd International Conference on Advanced Information Networking and Applications (AINA-2019) 33*, pp. 619–630, Springer, 2020.

[189] M. Kaushik, R. Sharma, S. A. Peious, M. Shahin, S. Ben Yahia, and D. Draheim, "On the potential of numerical association rule mining," in *Future Data and Security Engineering. Big Data, Security and Privacy, Smart City and Industry 4.0 Applications: 7th International Conference, FDSE 2020, Quy Nhon, Vietnam, November 25–27, 2020, Proceedings 7*, pp. 3–20, Springer, 2020.

[190] W. E. Djeddi, S. Ben Yahia, and E. M. Nguifo, "A novel computational approach for global alignment for multiple biological networks," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 15, no. 6, pp. 2060–2066, 2018.

[191] S. Abdellatif, M. A. Ben Hassine, S. Ben Yahia, and A. Bouzeghoub, "ARCID: A new approach to deal with imbalanced datasets classification," in *SOFSEM 2018: Theory and Practice of Computer Science: 44th International Conference*

on Current Trends in Theory and Practice of Computer Science, Krems, Austria, January 29-February 2, 2018, Proceedings 44, pp. 569–580, Springer, 2018.

[192] R. Sharma, M. Kaushik, S. A. Peious, S. Ben Yahia, and D. Draheim, "Expected vs. unexpected: Selecting right measures of interestingness," in *Big Data Analytics and Knowledge Discovery: 22nd International Conference, DaWaK 2020, Bratislava, Slovakia, September 14–17, 2020, Proceedings 22*, pp. 38–47, Springer, 2020.

[193] I. Osman, S. F. Pileggi, S. Ben Yahia, and G. Diallo, "An alignment-based implementation of a holistic ontology integration method," *MethodsX*, vol. 8, p. 101460, 2021.

[194] J. Zakraoui, S. Elloumi, J. M. Alja'am, and S. Ben Yahia, "Improving arabic text to image mapping using a robust machine learning technique," *IEEE access*, vol. 7, pp. 18772–18782, 2019.

[195] M. Shahin, S. Arakkal Peious, R. Sharma, M. Kaushik, S. Ben Yahia, S. A. Shah, and D. Draheim, "Big data analytics in association rule mining: A systematic literature review," in *2021 the 3rd International Conference on Big Data Engineering and Technology (BDET)*, pp. 40–49, 2021.

[196] R. Bhattarai, I. Pappel, H. Vainsalu, S. Ben Yahia, and D. Draheim, "The impact of the single digital gateway regulation from the citizens' perspective," *Procedia Computer Science*, vol. 164, pp. 159–167, 2019.

[197] S. Allani, R. Chbeir, T. Yeferny, and S. Ben Yahia, "Smart directional data aggregation in VANETs," in *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pp. 63–70, IEEE, 2018.

[198] I. B. Sassi, S. Ben Yahia, and I. Liiv, "MORec: At the crossroads of context-aware and multi-criteria decision making for online music recommendation," *Expert Systems with Applications*, vol. 183, p. 115375, 2021.

[199] B. Abidi, S. Ben Yahia, and C. Perera, "Hybrid microaggregation for privacy preserving data mining," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 23–38, 2020.

[200] S. Suran, V. Pattanaik, S. Ben Yahia, and D. Draheim, "Exploratory analysis of collective intelligence projects developed within the eu-horizon 2020 framework," in *Computational Collective Intelligence: 11th International Conference, ICCCI 2019, Hendaye, France, September 4–6, 2019, Proceedings, Part II 11*, pp. 285–296, Springer, 2019.

[201] S. Ben Chaabene, T. Yeferny, and S. Ben Yahia, "A roadside unit deployment framework for enhancing transportation services in maghrebian cities," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 1, p. e5611, 2021.

[202] S. Arakkal Peious, R. Sharma, M. Kaushik, S. A. Shah, and S. Ben Yahia, "Grand reports: A tool for generalizing association rule mining to numeric target values," in *Big Data Analytics and Knowledge Discovery: 22nd International Conference, DaWaK 2020, Bratislava, Slovakia, September 14–17, 2020, Proceedings 22*, pp. 28–37, Springer, 2020.

[203] T. Yeferny, S. Hamad, and S. Ben Yahia, "Query learning-based scheme for pertinent resource lookup in mobile P2P networks," *IEEE Access*, vol. 7, pp. 49059–49068, 2019.

[204] A. Houari, W. Ayadi, and S. Ben Yahia, "NBF: An FCA-based algorithm to identify negative correlation biclusters of dna microarray data," in *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pp. 1003–1010, IEEE, 2018.

[205] B. Sellami, A. Hakiri, S. B. Yahia, and P. Berthou, "Energy-aware task scheduling and offloading using deep reinforcement learning in sdn-enabled iot network," *Computer Networks*, vol. 210, p. 108957, 2022.

[206] B. Sellami, A. Hakiri, and S. B. Yahia, "Deep reinforcement learning for energy-aware task offloading in join sdn-blockchain 5g massive iot edge network," *Future Generation Computer Systems*, vol. 137, pp. 363–379, 2022.

[207] B. Sellami, A. Hakiri, S. B. Yahia, and P. Berthou, "Deep reinforcement learning for energy-efficient task scheduling in sdn-based iot network," in *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, pp. 1–4, IEEE, 2020.

[208] A. Hakiri, B. Sellami, S. B. Yahia, and P. Berthou, "A sdn-based iot architecture framework for efficient energy management in smart buildings," in *2020 Global Information Infrastructure and Networking Symposium (GIIS)*, pp. 1–6, IEEE, 2020.

[209] A. Hakiri, B. Sellami, S. B. Yahia, and P. Berthou, "A blockchain architecture for sdn-enabled tamper-resistant iot networks," in *2020 Global Information Infrastructure and Networking Symposium (GIIS)*, pp. 1–4, IEEE, 2020.

[210] A. Hakiri, B. Sellami, P. Patil, P. Berthou, and A. Gokhale, "Managing wireless fog networks using software-defined networking," in *2017 ieee/acs 14th interna-*

tional conference on computer systems and applications (aiccsa), pp. 1149–1156, IEEE, 2017.

[211] A. Hakiri, B. Sallemi, F. Ghandour, and S. Ben Yahia, "Secure, context-aware and qos-enabled sdn architecture to improve energy efficiency in iot-based smart buildings," in *Distributed Computing for Emerging Smart Networks: Second International Workshop, DiCES-N 2020, Bizerte, Tunisia, December 18, 2020, Proceedings 2*, pp. 55–74, Springer, 2020.