Automated Pricing-based Provisioning of SDN/NFV Services in Distributed Multi-access Edge Computing using Cooperative Multi-Agent Deep Reinforcement Learning

Jean Jimmy Julien [1,*], Sirapop Nuannimnoi[1], and Ching-Yao Huang[2]
[1]EECS International Graduate Program, National Yang-Ming Chiao Tung University, Hsinshu City, Taiwan
[2]Electronics Engineering, National Yang-Ming Chiao Tung University, Hsinshu City, Taiwan
jimfirstone.ee10@nycu.edu.tw, sirapop.ee07@nycu.edu.tw, cyhuang@nctu.edu.tw
*corresponding author

Abstract—The disruption caused by Software Defined Network and Network Function Virtualization (SDN/NFV) technologies will have many impacts on the telecom network. Specifically, the network architecture based on ETSI MANO comprising Virtual Infrastructure Manager (VIM), Virtual Network Function Manager (VNFM), and NFV Orchestrator (NFVO) will significantly change how we operate and manage the telecom network. These impacts on the network architecture will be made gradually. Thus, the migration from non-virtualized networks to all virtualized networks will happen step by step. By introducing new actors into the telecom ecosystem, NFV-MANO will bring in new business models. It is envisaged that these new actors/models will promote competition hence the demand for more flexible charging models with real-time charging. In this research, we will address the architectural realization of the SDN/NFV charging model under the business model of MANO (MANagement and Orchestration) when the service provider (SP) uses the Network Function Virtualization Infrastructure (NFVI) from the NFVIaaS Provider in a distributed multi-access edge computing (MEC) environment. To optimize the Quality of Service (QoS) of MEC resource allocation for incoming services and maximize the overall operating profit of the service provider adopting our charging model, we also propose a new cooperative multi-agent actor-critic based deep reinforcement learning (MADRL) method trained with proximal policy optimization algorithm, namely Coop MAPPO. The results of our experiments showcase the superiority of the Coop-MAPPO multi-agent system over alternative decision-making approaches, with its potential for enhancing operational efficiency and profitability while minimizing failure rates.

Keywords- Multi-Agent Deep Reinforcement Learning; Distributed Edge Computing; Software Defined Network; Network Function Virtualization; Service Provisioning; Charging Factors; Charging Models

1. INTRODUCTION

The emergence of Software Defined Networks (SDN) and Network Function Virtualization (NFV) technologies marks the onset of a revolutionary phase in traditional networking paradigms. They create vast opportunities for telecommunications operators to reshape their networks and clouds infrastructures into cost-effective, elastically scaling environments. These changes will greatly affect how an operator designs, develops, manages, delivers, and charges its products and services. In order to adapt to these disruptive changes caused by SDN/NFV, telecom operators will need to change the way they used to charge and bill their customers for their products and services. The virtualization of Software Defined Networking (SDN) and Network Function Virtualization (NFV) is set to introduce several new actors in the ecosystem, including VNF vendors/VNF marketplaces, VNF as a Service (VNFaaS) provider, Network Function Virtualization Infrastructure as a Service (NFVIaaS) provider [1]. These actors can generate three disctint business models: Service Providers (SP) using the NFVI from NFVIaaS, SPs/VNFaaS provider offering VNF as a service to xVNOs (Virtual Network Operators), and Communication Service Providers (CSPs) leasing the VNF S/W(software) from VNF Providers. By reshaping the conventional networking paradigms, new revenue streams and new charging factors/models based on different assumptions under a desired business model can be introduced into the telecommunications ecosystem. Consequently, this evolution stands to challenge the conventional pricing and charging model due to its high flexibility. For example, CSPs can lease VNF software from VNF providers, and be charged by considering the type of VNF and its number of instances. In this business model, the charging model will be based on the maximum number of VNF requests, number of requests handled by VNFs, etc. In a previous study [2, 3], we identified and categorized new charging factors for SDN/NFV by linking them to MANO (Management and Orchestration) events and defined new charging models based on these charging factors to enable pricing of SDN/NFV products and services.

Multi-Edge Computing is a valuable approach to improving the performance and scalability of NFV services by offloading requests to the edge of the network. It refers to a further extension of MEC, where computing resources are deployed at multiple edge locations, distributed across the network [4 - 7]. This allows for even greater flexibility and

2767-6684/23/$31.00 ©2023 IEEE
DOI 10.1109/DSA59317.2023.00027

scalability in offloading NFV requests to the edge. MEC provides several benefits for NFV request offloading. Firstly, by offloading requests to the edge of the network, NFV services can be delivered with lower latency, improving the user experience. Secondly, MEC allows for greater scalability of NFV services, by distributing computing resources across multiple edge locations. Next, offloading requests to the edge of the network can reduce network congestion and improve overall network performance. Last but not least, by deploying computing resources closer to the end-users, MEC can provide enhanced security for NFV services.

Reinforcement learning (RL) [8] is a group of Artificial Intelligence (AI) techniques that focus on developing an AI agent or multiple AI agents capable of taking actions in an environment to maximize a notion of cumulative reward programmed by AI practitioners. With the advancement of deep learning (DL) [9], deep RL (DRL) has emerged as an effective set of AI algorithms to tackle several real-world problems in many areas, including even communication technology development, optimization, and automation in MEC networks [10 – 14]. However, despite the success of DRL in multiple aforementioned scenarios, as the environments in real-world scenarios grow bigger, one well-trained AI agent may not be enough to efficiently handle the given objectives. This is where the concept of multi-agent RL (MARL) [15 -18] comes into play. In a MARL system, there are at least two AI agents inside the shared environment either working together or competing against each other to achieve the shared objectives. MARL allows the incorporation of all the different angles of the environment and makes it easier to expand as each AI agent can be either trained independently or collaboratively.

In this research article, we will address the charging architecture of SDN/NFV in a distributed multi-edge access (MEC) network. We also propose a multi-agent system based on deep reinforcement learning for intelligent NFVIaaS resource provisioning and charging. The goal is to develop Edge-AI agents that can minimize execution latency and operation costs, and maximize overall profit for the host service provider. The main contributions of this paper can be summarized as follows.

- We address the MANO-based NFVIaaS offline charging architecture in the MEC network for the business model when a service provider leverages NFVI resources from an NFVIaaS provider. NFVIaaS offline charging is implemented by utilizing a charging data collector (CDC) located at the NFVI layer of each edge node to detect chargeable MANO events that correspond to charging factors. The CDC then sends the charging information to the Charging Data Function (CDF) simulator situated within the Virtual Infrastructure Manager (VIM) layer of each edge node. The role of the CDF simulator is to generate conclusive charging data records (CDRs) based on the received information.
- To automatically maximize the operational profit and minimize the service failures of the proposed MEC NFVIaaS, we design and develop a new multi-agent Edge-AI algorithm specifically for SDN / NFV service provisioning in MEC networks based on cooperative multi-agent actor-critic methods with proximal policy optimization, namely Coop MAPPO. The model trained with this algorithm can be deployed on each edge node decentrally and does not require that NFV requests are well-modeled. Our experiment results show that Coop MAPPO obtains higher rewards as compared to baseline algorithms.

The rest of this article is organized as follows. Section II discusses the background knowledge and reviews the related works in network function virtualization infrastructure, distributed edge computing, and multi-agent deep reinforcement learning. Section III defines the problem formulation. Section IV presents the proposed system architecture and the design of distributed Edge-AI. Section V shows an analysis of our simulation results. Finally, Section VI concludes this article and lists out possible future research directions.

2. Background Knowledge and Related Work

As discussed above, SDN/NFV will systematically influence network architecture. Migration from non-virtualized networks to all virtualized networks will be a gradual process, as outlined in [19]. In this section, we discuss the background knowledge and related works on SDN / NFV technologies, charging factors, charging models, and the applications of multi-agent reinforcement learning systems in resource provisioning.

2.1. Related Work

The influence of SDN/NFV on network architectures and management is introducing new pricing and business models to the telecommunications ecosystem. To illustrate, Naudts et al. [20] introduced an advanced dynamic pricing algorithm for required substrate resources. Their algorithm leveraged the infrastructure provider's revenue based on historical data, current infrastructure utilization levels, and competitor pricing. For another example, Chekired et al. [21], where they proposed a real-time dynamic pricing model within a cloud architecture designed for the charging and discharging services of electric vehicles and building energy management. This cloud computing architecture was built upon the foundations of SDN and NFV. However, it is important to note that none of these aforementioned SDN/NFV initiatives have explicitly defined charging models and architectures.

Some possible charging architectures have been developed in the cloud and M2M (Machine to Machine). To the best of our knowledge, cloud services can be categorized into three main

types: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [22]. Various pricing models and architectures can be applied to different types of cloud services based on their pricing factors. For example, Zhang et al. [23] presented an open standards-based framework for the integration of IMS (IP Multimedia Subsystem) and cloud computing. The charging system described in this document has initiated the charging process and sent offline or online charging requests. Furthermore, M2M communication introduces new charging models and architecture based on the weight and expected usage parameters of the four basic charging factors: data transfer, storage, connectivity and subscription [24 - 27]. In addition, Lin et al. [26] presented an M2M charging architecture design to collect information about defined charging factors in both the service and network layers.

2.2. ETSI Management and Orchestration (MANO)

NFV MANO constitutes an architectural framework designed to oversee the management and orchestration of virtualized network functions (VNFs) alongside other software components. It supports the management and orchestration of all resources in a cloud data center, including compute, network, storage, and virtual machine (VM). It enables flexible integration of new services and supports rapid scalability of network components. Illustrated in Figure 1, the ETSI MANO architecture is segmented into three primary functional divisions [28]: NFV orchestrator (NFVO), VNF manager (VNFM), and virtualized infrastructure manager (VIM). The goal of these three blocks is to deploy and connect functions and services throughout the network. Furthermore, MANO includes eight reference points, denoted as Ve-Vnfm-em, Nf-Vi, Ve-Vnfm-vnf, Or-Vi, Os-Ma-nfvo, Vn-Nf, Or-Vnfm, Vi-Vnfm [29-37], each of which manages the exchange of events among different functional blocks in MANO.

Within the MANO framework, events are categorized into two distinct types: Management and Control Events, and Usage and Data Events. The former is employed to initiate the creation or deletion of VNF instances and the dynamic scaling of such instances. Conversely, usage and data events furnish details concerning the utilization of NFV resources in terms of volume, duration, or a combination of both [38]. It is important to know which MANO events need to be measured to collect information about charging factors, as not all events are charging events. Only those related to the business model will be utilized for charging and billing purposes.

2.3. Charging Factors of SDN / NFV

Charging factors serve as the metrics employed by the operators to charge for services offered to their clients. In our prior research efforts, we have successfully identified and outlined a comprehensive set of charging factors for the new

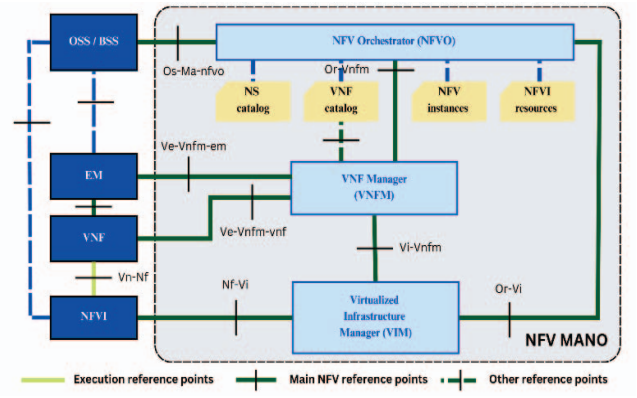


Figure. 1. MANO Architecture.

business model introduced by SDN/NFV [2]. In this section, we will provide a summary of the charging factors for the "Service Provider (SP) using NFVI from an NFVIaaS provider" business model. In this business model, the service provider (SP) will be charged for the use of NFVI resources. We identified seven such charging factors, including:

- Type of CPU used - A virtual CPU, commonly referred to as a virtual processor, represents a physical central processing unit (CPU) allocated to a virtual machine (VM). This charging factor refers to what type of CPU resource is offered to the service provider(SP). Illustrative examples encompass single-core CPUs, dual-core CPUs, quad-core CPUs, etc.
- CPU time usage - The CPU time signifies the duration of processor time utilized by a particular service. Corresponding charging will be applied to the service provider accordingly.
- Type of storage used - This charging pertains to the category of storage the service provider intends to utilize from NFVI. Different storage types are available, including primary storage and secondary storage.
- Storage usage - This charging factor quantifies the volume of storage allocated to the service provider. The service providers will be charged based on the size, the duration and the location of the storage it actively utilizes from the NFVI.
- Bandwidth - This charging factor signifies the upper limit of data transfer speed within a network or internet connection. A service provider will be charged according to the bandwidth provided to them.
- QoS Level - The Quality of Service (QoS) Level stands as a service assurance provided by the NFVIaaS provider to the service provider.
- Type of Accelerator used - This charging factor pertains to the specific category of accelerator resource extended to the service provider. Illustrative instances encompass hardware accelerators, graphics accelerators, cryptographic accelerators, web accelerators, and more.

2.4. Charging Models of SDN / NFV

Charging models serve as mechanisms that enable service providers to establish pricing for their offerings within the telecom network. In addition, charging models can help telecom customers find the right pricing plan for their specific network usage. Previously, we defined a list of charging models for each of the business models introduced by SDN/NFV [3]. This section will provide a concise overview of the charging models for the "Service provider(SP) using NFVI from an NFVIaaS provider" business model. These charging models are categorized based on the classification of charging factors delineated in our preceding research [2]. Users can be charged either based on volume and quantity or based on value. Where value-based charging factors are relevant, the customer will be charged in correspondence with the quality of services they receive. Conversely, in scenarios where volume-based charging factors come into play, customers will be charged for the amount of services they purchase. In this context, various charging models can be formulated by employing pertinent subsets of charging factors from both categories, as depicted in Table 1.

Table 1 illustrates several examples of volume-based and quantity-based charging models, such as " postpaid periodic billing based on maximum CPU usage limit " and " Prepaid billing based on maximum Storage usage limit ". The former involves charging users a fixed fee at regular intervals, like weekly or monthly, while the latter approach would immediately deduct the fee from a prepaid account in real-time, corresponding to the quantity of storage utilized.

It is crucial to note that the charging factors provided in Table 1 can be utilized to track resource consumption either in real-time or over specified time frames. Table 1 also presents some examples of charging models derived from value-based charging factors, such as " postpaid periodic billing based on maximum bandwidth and level of QoS supported", " postpaid periodic billing based on level of QoS and type of accelerator used". Importantly, it should be noted that all charging models relying on value-based charging factors within this particular business model are designed exclusively to accommodate postpaid models, as they do not operate on the premise of charging based on quantity and volume consumed."

Table 1. Charging models examples derived from NFVIaaS Charging Factors

Business Model	**Charging Factors Combination**	**Examples of Charging models**
Service provider(SP) using the NFVI from NFVIaaS Provider	• Type of CPU used • Amount of CPU used	✓ Postpaid monthly billing based on CPU usage limit ✓ Prepaid billing based on CPU usage limit
	• Type of storage used • Amount of storage used	✓ Postpaid periodic billing based on maximum Storage usage limit ✓ Prepaid billing based on Storage usage limit
	• Type of CPU • Type of storage • Amount of CPU usage • Amount of storage usage	✓ Postpaid periodic billing based on Storage usage limit ✓ Prepaid billing based on Storage usage limit
	• Bandwidth • QoS • Type of accelerator	✓ Postpaid Periodic billing based on maximum bandwidth, level of QoS and type of accelerator used.

2.5. Multi-Agent DRL

As introduced in the earlier section, reinforcement learning (RL) [8] is a class of machine learning methods concerned with how to train one or multiple intelligent agents to take actions in an environment with the goal to maximize a notion of cumulative reward. A DRL algorithm such as Deep Dueling Q-Networks [11] can be used to find an optimal slice request admission policy in just a few iterations of training. In scenarios similar to our use case, DRL also seems to provide promising results for service resource provisioning in NFV-enabled networks. Nouruzi et al. [12] formulated an optimization problem with the aim to minimize the cost of NFV-enabled network resource utilization and used a Deep Q Network (DQN) algorithm for resource allocation for function placement and dynamic routing by considering the available network resources as DQN states, which resulted in up to 14 % increase in the number of admitted network requests the average number and up to 20 % reduction in the network utilization cost. Li et al. [13] also studied and used a DRL algorithm for online service placement and request assignment problems in a MEC network where future request arrival is unpredictable. Their simulation results showed an improvement of almost 50% in performance compared to other baseline algorithms operating in the same environment.

Despite the success of the DRL algorithms in several communication applications, a large number of real-world problems in this field still cannot be fully solved by a single active RL agent that interacts with the environment. The direct solution to this challenge is to adopt the use of multi-agent systems. In a multi-agent reinforcement learning (MARL) setting [15 – 18], multiple agents are put together in a shared environment where their interests might be aligned or misaligned. MARL allows exploring all the different alignments and how they affect the agents' behavior. The first notable example of how MARL can be applied to a larger communication setting can be found in an article recently published by Chen et al. [16], in which they designed a multi-

agent deep RL-based elasticity control approach (DRLEC) to tackle the pivotal problem in maintaining the quality of service (QoS) and minimizing network cost in SDN/ NFV networks. They demonstrated that DRLEC would provide better performance than heuristics and single-agent DQN algorithms. Additionally, DRLEC could almost achieve the ideal performance which would only happen in case the dynamics of network traffic were in advance. In another example, Tam et al. [17] trained and used multi-agent deep Q-networks (MADQNs) to enforce a self-learning softwarization, optimize resource allocation policies, and advocate computation offloading decisions in an NFVI-MEC environment of a large-scale heterogeneous software-defined IoT cellular network. Similarly, since network traffic and computing demand have been changing dramatically due to diverse types of network services, such as high-quality video delivery and operating system (OS) updates, Suzuki et al. [18] proposed a dynamic virtual network (VN) allocation method based on cooperative multi-agent deep reinforcement learning, namely Coop-MADRL. In a scenario where the variables in the shared environment are fluctuating, the proposed Coop-MADRL could still achieve generalized performance in maximizing the utilization efficiency of limited network resources. The key idea in their work is to utilize the concept of a multi-agent system to reduce the number of candidate actions per agent and can improve the performance for VN allocation since the input parameters, and required resources to train and operate each agent reduce greatly, in comparison to having one big AI agent to operate on bigger environment information.

3. Problem Formulation

As shown in Figure 2, we consider a multi-access edge node ecosystem, which consists of a set of edge nodes E= $\{e_1, e_2, \dots, \}$. Each edge node e_i is equipped with the NFVI resources to realize the business model of SDN/NFV where the "Service Providers (SP) making use of the NFVI resources from NFVIaaS Providers". Therefore, the service providers can be charged by the usage of the NFVI resources in our proposed ecosystem. Besides, each edge node e_i hosts a logical smart edge NFV orchestrator to manage the NFVIaaS and its resources. The goal of the service provider managing the whole ecosystem is to maximize its operational profit while maintaining low failure rates across the MEC-enabled NFVIaaS system. Each edge node e_i has limited computation and communication capacities. Each edge node has |R| resource types, and the resources type are the number of CPU, memory and bandwidth which are among the most frequently utilized parameters. For every charging factor i relevant to a charging model, a corresponding charging function f_i can be formulated according to its requisite parameters denoted as $p_{i,1}, p_{i,2}, p_{i,3}, \dots \dots p_{i,m}$. Presented below is the definition of a charging function for a charging

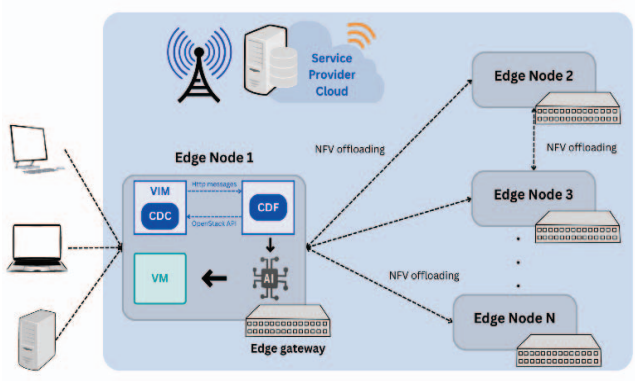


Figure 2. NFVIaaS System Overview

model based on CPU, memory and network resources along with their respective parameters:

1) Charging function for CPU based charging model:

$$f_{cpu}(CPU_{core}, CPU_{time}, CPU_{unit_{cost}}) = CPU_{core} \times CPU_{time} \times CPU_{unit_{cost}} \quad (1)$$

2) Charging function for memory based charging model:

$$f_{Mem}(Mem_{time}, Mem_{unit_{cost}}) = Mem_{time}\ x\ Mem_{unit_{cost}} \quad (2)$$

3) Charging function for network based charging model:

$$f_{BW}(BW_{time}, BW_{unit_{cost}}) = BW_{time}\ x\ BW_{Unit_{cost}} \quad (3)$$

Note that for each charging factor i, a comprehensive market research and commercial analysis is required to ascertain the weight of that factor, w_i, in the total pricing. Thus, the price of a charging model comprising n charging factors can be computed utilizing the charging function of each charging factor i and its corresponding weight w_i by the following overall pricing formula:

$$\sum_{n=i}^{n}(f_i \times w_i) \quad (4)$$

The profit maximization problem:

$$\max_{x \geq 0} \pi(Req)\ /\lambda(Req) \quad (5)$$

here π is our profit function and λ is our failure function. Letting R be the revenue function which is the total revenue acquire by the SP from the MEC node and C the cost function. The failure function can be defined as:

$$\lambda(Req) = f\_num/num(Req) \quad (6)$$

$$\pi(Req) = R(Req) - C(Req) \quad (7)$$

The profit function can be defined as the total price of the resources deducted by the total cost that the service provider will pay for the NFVI resources, which can be given as followed:

$$R = P_1^{CPU} + P_2^{Mem} + P_3^{BW} \quad (8)$$

$$C_t = C^{cpu}(t) + C^{mem}(t) + C^{BW}(t) \quad (9)$$

We assume that the expected values of the CPU, memory / storage, and network utilizations follow the uniform distributions under the range given in Table 3. Moreover, for simplicity, we also list the main symbols and mathematical notations along with their definitions adopted in this paper in Table 2.

4. Proposed Multi-Agent NFVIaaS Platform

Figure 2 illustrates the architecture of the proposed intelligent NFVI resource provisioning system and how each AI agent provisions the system and communicates with one another.

4.1. Data Collection and Simulations

In our prior research [2], we identified how to gather information about the charging factors from related MANO events. To illustrate, consider the following three (3) events from the NF-VI reference point: Allocate VM with an indication of compute resource, Update VM resources allocation, and Terminate VM, we can get the information about the "CPU type" charging factor. On the other hand, for the following three MANO events in the NF-VI reference point: Create a connection between VMs, configure connection between VMs and Remove connection between VMs, we can acquire the requisite information for the 'Bandwidth' charging factor. As depicted in Figure 2, for the NFVIaaS charging architecture, we use of a charging data collector (CDC) situated within the NFVI layer of each edge node to detect chargeable events such as "allocate VM with indication of compute resource", "update VM resources allocation", and "terminate VM", which correspond to the charging factors. Subsequently, the CDC transmits this pertinent charging information to the Charging Data Function (CDF) simulator positioned within the Virtual Infrastructure Manager (VIM) layer to generate final charging data records (CDRs).

Table 2. Table of symbols

Notations	Description
E	The number of edges nodes.
CPU_{core}	The number of CPU cores available on each node.
CPU_{time}	The amount of CPU time utilized on each edge node (e_i)
$CPU_{unit_{cost}}$	CPU unit time price.
Mem_{time}	how much memory time is used on each edge node (e_i) .
$Mem_{unit_{cost}}$	Memory unit time price.
BW_{time}	how much bandwidth time is used on each edge node (e_i).
$BW_{Unit_{cost}}$	Bandwidth usage unit price.
$\pi(Req)$	The profit earned by the service provider.
$R(Req)$	Total price of the NFVI resources.
$C(Req)$	Total cost paid by the service provider for the NFVI resources.
f_num_t	Number of failures at time step t.
f_{init}	Initial number of failures before the simulation begins.
$\tau_{\|NFV\|}$	Remaining time required to complete a NFV request.
η	Importance weight between operational profit and failures. The default value is 0.5.
P_t	Revenue of an edge node at time t.

4.2. Design of Multi-Agent DRL Algorithm

In this subsection, we first provide the description of our Edge-AI architecture design.

Table 3. Settings of Parameters for Simulations

Parameters	Values / Ranges
Number of edge nodes	5
Time per time step / episode (in ms)	100
Number of NFV requests per episode	50
Number of episodes in epoch	200, 50
Number of epochs	10
Storage capacity (in GB)	4 – 1024
Memory capacity (in GB)	4 – 16
CPU cores	[1, 2, 4] : single-, dual-, and quad-core
CPU clock speeds (GHz)	Single-core: 2.5 Dual-core: 3.5 Quad-core: 5.0
CPU cycle capacity (millions of instructions)	3,000 – 10,000 (Scaled: 3 - 10)
Bandwidth capacity (in Gbps)	0.1 – 10
Required CPU cycles (millions of instructions)	100 – 5,000 (Scaled: 0.1 - 5)
Required memory (in GB)	0.1 – 2
Required storage (in GB)	0.1 – 5
Required bandwidth (in Gbps)	0.1 – 1
Required completion time (in ms)	100 – 1,000
CPU unit cost (in $x10^{-4}$ USD / second)	0.5 – 1.0
Memory unit cost (in $x10^{-4}$ USD / MB second)	0.4 – 0.9
Bandwidth unit cost (in $x10^{-4}$ USD / MB)	0.4 – 0.9
Discounts (%)	0.01 – 0.3

Then, we present a cooperative multi-agent proximal policy optimization (Coop MAPPO) algorithm for solving the defined multi-agent RL problem. We begin by providing the detail of Markov Decision Processes (MDP) in our work as follows:

- State Space

We define the state space *S* as:

$$S \subseteq R^{E \times O}$$

where E and O denote the number of edge nodes participating in this NFVIaaS system and number of observed state features and parameters, respectively. A state at time step t can be given by:

$$s_t = [U_t, F_t, L_t, C_t, Req_t] \quad (10)$$

It summarizes the environment information of the entire MEC platform at time step *t*. Since we assume that all agents in this system always communicate with their peers whenever they are online, every node has the same state vector. $U_t = [u_{cpu,1}, u_{mem,1}, u_{bw,1}, u_{cpu,2}, u_{mem,2}, u_{bw,2}, \ldots, u_{cpu,E}, u_{mem,E}, u_{bw,E}]$ is the utilization rate vector of each edge node at time step *t*. Features inside this vector contain the current utilization levels of CPU, memory and bandwidth of every participating edge node. $F_t = [f_1, f_2, f_3, \ldots, f_E]$ is the vector of operational failure rates of each participating nodes at time step *t*. These failure rates are randomly generated based on Weibull distributions with the shape value equal to 2 and scale parameter equal to 1. We assume that every participating node has received 1,000 requests before the beginning of this simulation (f_{init}). Therefore, we update the failure rates with:

$$f_{t,E} = \left[f\_num_t + (f_{t-1,E} \times f_{init})\right] \Big/ (|NFV|_t + f_{init}) \quad (11)$$

To decide whether the task being successfully accepted or fail at each time step or not, we apply the following rule after the acceptance decisions have been made:

$$\begin{cases} req_{NFV,time} - \tau_{NFV} \geq 0 & success \\ req_{NFV,time} - \tau_{NFV} < 0 & fail \end{cases}$$

Furthermore, to decide whether the task being offloaded to other nodes will fail at each time step or not, we first convert the required resources to the available resources on the target nodes, e.g., if the original CPU type requirement is single-core but the selected node has a dual-core CPU, then the time it takes to complete the request should be shortened. To convert the required resource from the original node to the resource unit of the target node, we rely on the following equation:

$$CPU_{cycles} = CPU_{time} * speed(CPU_{core})$$

$$CPU_{time_{new}} = CPU_{cycles} \Big/ speed(CPU_{core_{new}})$$

We can determine whether the request will be satisfied by the selected node or not based on the utilization levels and the required resource levels. Thus, we apply the following rules:

$$\begin{cases} req_{NFV,utilization} < 1 - u_{res,E} & success \\ req_{NFV,utilization} \geq 1 - u_{res,E} & fail \end{cases}$$

$L_t = [l_1, l_2, l_3, \ldots, l_E]$ is the vector of maximum time required for all remaining tasks of each participating nodes at time step *t*. The maximum time required for each edge node to complete all remaining task can be given by:

$$l_t = \max(0, \tau_1, \tau_2, \ldots, \tau_{|NFV|}) \quad (12)$$

C_t or $C(t)$ is the vector of costs, as defined earlier in Section 3, of all participating nodes at time step *t*. $Req_t = [req_{1,cpu}, req_{1,mem}, req_{1,bw}, req_{1,storage}, req_{1,time}, req_{2,cpu}, req_{2,mem}, req_{2,bw}, req_{2,storage}, req_{2,time}, \ldots, req_{|NFV|,mem}, req_{|NFV|,bw}, req_{|NFV|,storage}, req_{|NFV|,time}]$ is the vector containing information regarding the required resources of all NFV requests at time step *t*.

- Action space

Each agent has to decide whether to accept the incoming NFV service request or push it to other interconnected edge nodes in the network. Therefore, we define the action space $A = \{a_1, a_2, a_3, \ldots, a_E\}$, where an action a_t can be an integer within the range $[0, E)$ and E is the number of participating edge nodes.

- Reward function

In this article, the optimization goal needs to consider both operational profit maximization and failure minimization. Therefore, the intermediate reward function of each agent at time t can be formulated as

$$r_t = \eta \sum(\mathrm{P}_t - C_t) \Big/ (1 - \eta) f\_num_t \quad (13)$$

where η is an importance factor. If the value is high, the agent will prioritize operational profit. On the other hand, if the value is low, the agent will prioritize the overall failure rates of task offloading. In this study, we weight both equally, thus we set the default value of this parameter to 0.5.

- Learning Algorithm

Deep RL algorithms can be divided into two most common types: a) value-based DRL and b) policy-based DRL algorithms [39]. Value-based Deep RL algorithms, such as DQN, design value networks to estimate Q-values and then improve their policies based on the learned value functions. However, value-based DRL algorithms cannot solve large-

scale and continuous action space problems such as ours. Due to the nature of greedy action selection within value-based DRL, these algorithms cannot handle stochastic policy problems. Policy-based DRL algorithms, on the other hand, directly calculate the policies to maximize the discounted rewards. The most common policy-based DRL is the policy gradient (PG) algorithm. Although the PG algorithms have strong versatility and a more stable training process, they have their inherent weaknesses due to large variance of trajectories, low sample utilization, and general attraction towards local optima.

To compensate for the weak points of both RL algorithm types, researchers have introduced actor-critic (AC) architecture into the field of Deep RL [40 – 42]. In the classic AC architecture, there are two neural networks working together: one is a critic network and the other is an actor network. The critic network, also known as the value network in some literature, is used to estimate the value function while the actor network, also known as the policy network in some literature, is used to optimize the policy according to the estimated value function. Since AC is a combination of both RL algorithm types, AC has the advantages of both algorithms (best of both worlds). The value function from the critic network makes the policy update with lower trajectory variance. At the same time, the policy from the actor network makes it possible to handle continuous action problems and improve the versatility. Proposed and developed in 2017 by OpenAI [43], the proximal policy optimization (PPO) algorithm is one of the most recent advancements in the field of DRL that provides an improvement on Actor-Critic (AC) architecture. Unlike classical AC, PPO ensures that the updated policy is not too different from the old policy to ensure low variance in training. This leads to smoother training and the assurance that the AI agent may not go down an unrecoverable path of taking senseless actions. PPO defines the probability ratio between the new policy and the old policy as

$$r(\theta) = \frac{\pi_\theta(a|s)}{\pi_\theta^{old}(a|s)} \tag{14}$$

Now the objective function of PPO can be written as

$$F^{Clip}(\theta) = E\left[\begin{matrix}\min(r(\theta)\, \hat{A}_{\theta_{old}}(s,a), \\ clip(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\theta_{old}}(s,a))\end{matrix}\right] \tag{15}$$

In the above equation, the function clip truncates the policy ratio between the range [1-ϵ, 1+ϵ]. The objective function of PPO takes the minimum value between the original value and the clipped value. Because of this surrogate objective, PPO allows the usage of multiple epochs of gradient ascent without stepping too far from the old policy. Therefore, in this work, we decide to use PPO to train multiple distributed AI agents to solve the cooperative MARL problem.

5. Evaluation

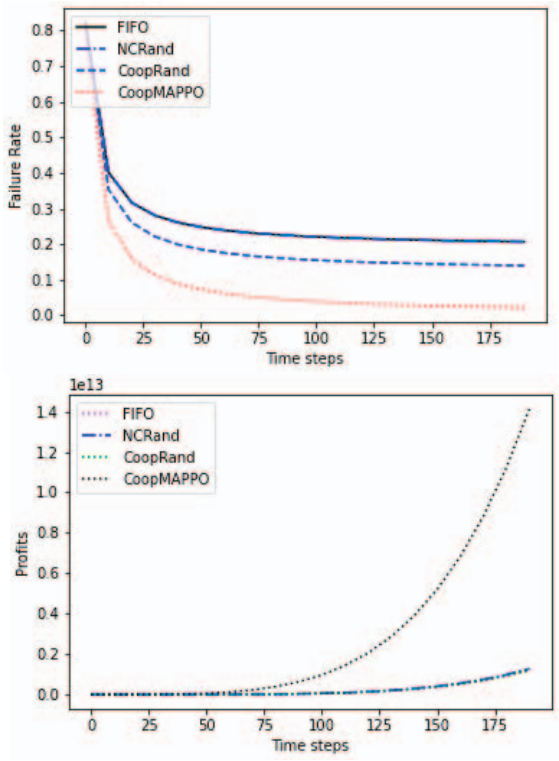
In this section, we focus on evaluating our proposed multi-agent system and comparing it with some other well-known multi-agent methods. We begin by describing the simulation settings of incoming service requests to our proposed NFVIaaS platform. Next, we describe the training process of the multi-agent Edge-AI system. Finally, we evaluate its performance in terms of operational costs and profit, and average response time, which are the key metrics to satisfy both the users' and the service providers' needs.

5.1. Incoming Service Request Simulations

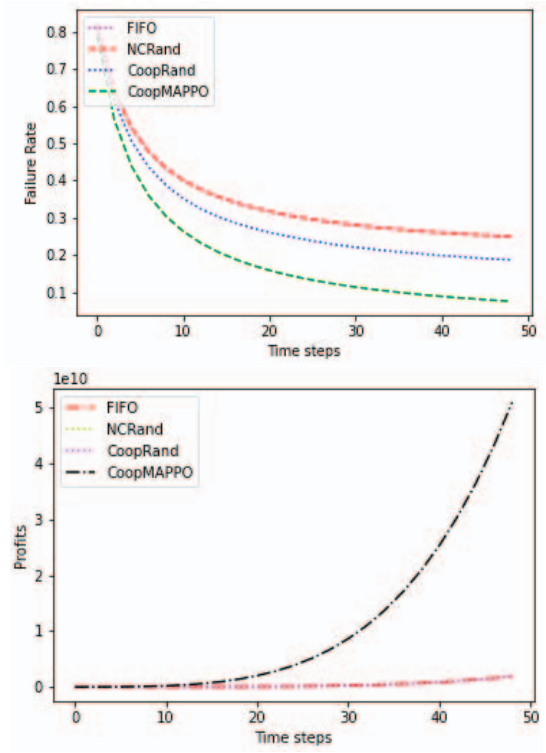
To make sure that our agents can effectively offload the NFV requests across the proposed ecosystem, we train our agents in parallel on multiple episodes or time steps of randomly generated incoming service requests based on uniform distributions and given parameter ranges as shown in Table 3. For fair comparison among the proposed multi-agent RL system and other baseline approaches, we first generate 50 requests for each time step for all edge nodes and save them to CSV files. To make sure that each agent is exposed to enough data, we generate 200 data points for each edge node. These data points are generated using pseudo-random number generators with different seed numbers, ranging from 1 to 10 based on the node IDs. Once each agent finishes all their decisions in each episode or time step, the intermediate reward will be calculated based on their operational profit and service completion failures. For simplicity, we have all agents consider their own received NFV requests all at once first before considering all other NFV requests offloaded from the other nodes. Moreover, this allows for the agents to have a better opportunity to coordinate their actions and allocate resources more efficiently. To test our agents, we simulate extra 100 data points following the same configurations.

For each time step or an episode, the agent repeats the following operations for NFV placement:

1) Each agent reads the information of every NFV service request, its own resource utilization levels, and last known states of other nodes according to (10).
2) For each NFV request Req_t, the agent makes the decision whether to accept the task or to offload to the other connected nodes.
3) If the agent accepts the task, the node will perform computation on such task while the agent moves on to make the offloading decision of other tasks in the queue based on the updated states.
4) Whenever an edge node receives a task offloaded from another node, it will add this task to their incoming task queue for decision making and replies with its updated state.
5) After an edge node makes the decision for all their own requests, offloading agent will receive an intermediate reward.



(a) Training period (50 requests, 200 time steps)



(b) Testing period (50 requests, 50 time steps)

Figure 3. Simulation results (a) training period (b) testing period

6) After the decisions of all requests in the current time step have been made, we move on to the next time step. Since we consider each time step equal to 100 milliseconds, any requests that require less than 100 milliseconds of computation will be assumed completed. If the tasks are incomplete, they will occupy the resources. Utilization levels will be updated before the agent starts considering new tasks in the new time step.

5.2. Evaluation Results and Analysis

For performance evaluation of our proposed intelligent NFVIaaS resource provisioning, we focus on two different groups of metrics: a) operational costs and profits, and b) service completion successes and failures. We compare our proposed multi-agent DRL method with two random decision makers based on multinomial distribution.

- Cooperative random off-loaders (Coop-Rand): random decisions generated under the multinomial distribution. Each decision made represents node ID to offload the request to.
- Non-cooperative random off-loaders (Non-Coop-Rand): random decisions generated under the binomial distribution. Each decision made is whether or not to run each task.
- FIFO: a decision agent that accepts requests sequentially until the node reaches capacity.

As mentioned in Table 3, we train all agents simultaneously on datasets with 50 NFV requests sending each participating nodes in each episode, 200 episodes in one epoch, and 10 epochs. 50 episodes of additional simulated data points are used for evaluating our proposed algorithm against the aforementioned baselines. The results from our experiments are as follows.

1) Operational Profits

Based on the simulation results presented in Figure 3.a and 3.b, comparing the performance of the proposed multi-agent system (Coop MAPPO) with baseline algorithms for NFV request offloading, our findings demonstrate that Coop MAPPO outperforms other algorithms in terms of maximizing cumulative profit over time.

A detailed breakdown of the cumulative profit generated by each node for both the proposed multi-agent system and the baseline algorithms is provided in Table 4. The results reveal that the MEC NFVIaaS system implemented with Coop MAPPO achieves a superior level of cumulative profit

compared to MEC NFVIaaS systems employing alternative baseline algorithms. Furthermore, our observation indicates that NFVIaaS systems that enable edge nodes to offload tasks to connected nodes exhibit significantly higher profit levels when compared to systems that solely rely on local task execution decisions. These findings emphasize the substantial financial advantages associated with allowing edge nodes to distribute their tasks among interconnected nodes.

Our experimental results strongly support the use of the proposed multi-agent system, Coop MAPPO, in optimizing cumulative profit over time. Additionally, they highlight the economic benefits gained by NFVIaaS systems that incorporate task offloading capabilities for edge nodes.

2) Task Completions

In addition to its promising financial advantages, our proposed Coop MAPPO exhibits a notable capability in maintaining low failure rates. This is achieved by training the agents to intelligently select optimal nodes for task.

Table 4. Profit after operating in test period (in $1,000)

Algorithm	Node 1	Node 2	Node 3	Node 4	Node 5
FIFO	26.54	38.25	49.45	46.63	22.57
NCRand	26.46	37.84	48.83	44.69	26.17
CoopRand	103.95	8157.87	4188.11	3179.89	3096.37
CoopMAPPO	97.16	8368.20	4990.41	4849.96	3381.78

Table 5. Failure rate after operating in test period

Algorithm	Node 1	Node 2	Node 3	Node 4	Node 5
FIFO	0.23	0.31	0.26	0.24	0.21
NCRand	0.22	0.31	0.26	0.24	0.20
CoopRand	0.01	0.15	0.16	0.18	0.19
CoopMAPPO	0.05	0.13	0.08	0.07	0.03

offloading, rather than accepting tasks indiscriminately when their utilization limits are reached. Consequently, the MEC NFVIaaS system that integrates Coop MAPPO surpasses other baseline algorithms in completing a greater number of NFV requests.

To complement these observations, the detailed data presented in Table 5 outlines the failure rates experienced by each node in both the proposed multi-agent system and the other baseline algorithms. It is noticeable that our Coop MAPPO demonstrates lower failure rates when compared to the failure rates observed in the other baseline algorithm's MEC NFVIaaS systems. This finding underscores the superior reliability and robustness of Coop MAPPO in mitigating failures. The application of Coop MAPPO contributes to a significant reduction in failure rates and enhances the ability of the system to effectively handle a larger volume of NFV requests compared to alternative baseline algorithms. This outcome emphasizes the practical advantages of adopting our proposed multi-agent system in real-world scenarios.

In addition to these two benefits of our proposed system, Coop MAPPO demonstrates strong scalability and adaptability characteristics. Its distributed nature, ability to handle a large volume of NFV requests, and dynamic decision-making process enable the system to effectively scale and adapt to various scenarios and scales. The agents can be duplicated, re-deployed on new edge nodes, and fine-tuned to fit with new edge nodes environment based on their own NFV request and utilization data. The agents can also be clustered into several cluster of multi-access edge ecosystems in order to control the size of input data of each individual agent. By leveraging the advantages of multi-agent systems and reinforcement learning, Coop MAPPO provides a practical solution for NFV offloading in real-world deployments. The system can be deployed in various NFVIaaS environments, ranging from small-scale deployments with a limited number of nodes to large-scale deployments with a significant number of interconnected resources.

6. Conclusion And Future Work

The advent of SDN/NFV virtualization introduces new dynamics and stakeholders in the telecommunications ecosystem, including VNF Vendors/VNF Marketplaces, VNFaaS Providers, and NFVIaaS Providers. These entities foster competition, leading to the need for more flexible pricing models such as prepaid and postpaid charging. In our previous research, we have explored and defined charging factors specific to SDN and NFV, enabling telecom operators to devise effective charging models for SDN/NFV accounting management.

In this paper, we have implemented an SDN/NFV charging architecture within a distributed multi-edge access (MEC) network. Additionally, we propose a multi-agent system based on deep reinforcement learning, aimed at intelligent resource provisioning and charging for NFVIaaS. The objective of our research is to develop Edge-AI agents capable of minimizing failure rates in NFV request executions, reducing operational costs, and maximizing overall profit for the MEC NFVIaaS platform. After conducting experiments with 100 episodes of 50 NFV requests from each edge node, our proposed cooperative multi-agent system based on PPO, namely Coop-MAPPO, outperform three other baseline decision makers in terms of average failure rates and operational profits.

For future work, we envision several directions that build upon the scalability and adaptability of our proposed system. These directions encompass the exploration of advanced pricing models for VNF as a Service (VNFaaS) and the practical implementation of charging models for SDN/NFV. VNF scheduling and resource allocation techniques can also be integrated with our Coop MAPPO multi-agent system to

further enhance scalability and adaptability. To expand the intelligence of Edge-AI agents within the proposed multi-agent system, novel algorithms and methodologies can be investigated. These algorithms and methodologies will empower agents to make more informed decisions in dynamic and heterogeneous environments, thereby enhancing the system's adaptability to varying network conditions and resource availability. Pursuing these directions will contribute to the advancement of SDN/NFV charging models, resource allocation strategies, and decision-making techniques, which could lead to more efficient and effective network and service management.

REFERENCES

[1] TM Forum GB989, "Impact of SDN/NFV on Charging and Billing", R15.5.1, March 2016.

[2] J. J. Julien, F. J. Lin, C. -H. Yu and W. -H. Hu, "Charging Factors for Enabling SDN/NFV Accounting Management," 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), Matsue, Japan, 2019, pp. 1-4, doi: 10.23919/APNOMS.2019.8893145.

[3] J. J. Julien, F. J. Lin, C. -H. Yu and W. -H. Hu, "Charging Models for MANO-based 5G Core Networks," 2020 International Computer Symposium (ICS), Tainan, Taiwan, 2020, pp. 19-24, doi: 10.1109/ICS51289.2020.00015.

[4] N. Kiran, X. Liu, S. Wang and C. Yin, "VNF Placement and Resource Allocation in SDN/NFV-Enabled MEC Networks," 2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Seoul, Korea (South), 2020, pp. 1-6, doi: 10.1109/WCNCW48565.2020.9124910.

[5] L. A. Haibeh, M. C. E. Yagoub and A. Jarray, "Agile Design and Dimensioning of MEC-NFV Infrastructure to Support Heterogeneous Service Chains," GLOBECOM 2022 - 2022 IEEE Global Communications Conference, Rio de Janeiro, Brazil, 2022, pp. 3670-3675, doi: 10.1109/GLOBECOM48099.2022.10001555.

[6] M. Wang, B. Cheng, W. Feng and J. Chen, "An Efficient Service Function Chain Placement Algorithm in a MEC-NFV Environment," 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9013235.

[7] B. Wu, J. Zeng, L. Ge, S. Shao, Y. Tang and X. Su, "Resource Allocation Optimization in the NFV-Enabled MEC Network Based on Game Theory," ICC 2019 - 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 2019, pp. 1-7, doi: 10.1109/ICC.2019.8761912.

[8] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," in IEEE Transactions on Neural Networks, vol. 9, no. 5, pp. 1054-1054, Sept. 1998, doi: 10.1109/TNN.1998.712192.

[9] X. Du, Y. Cai, S. Wang and L. Zhang, "Overview of deep learning," 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), 2016, pp. 159-164, doi: 10.1109/YAC.2016.7804882.

[10] J. Kaur, M. A. Khan, M. Iftikhar, M. Imran and Q. Emad Ul Haq, "Machine Learning Techniques for 5G and Beyond," in IEEE Access, vol. 9, pp. 23472-23488, 2021, doi: 10.1109/ACCESS.2021.3051557.

[11] W. Cheng, X. Liu, X. Wang and G. Nie, "Task Offloading and Resource Allocation for Industrial Internet of Things: A Double-Dueling Deep Q-Network Approach," in IEEE Access, vol. 10, pp. 103111-103120, 2022, doi: 10.1109/ACCESS.2022.3210248.

[12] A. Nouruzi, A. Zakeri, M. R. Javan, N. Mokari, R. Hussain and S. M. A. Kazmi, "Online Service Provisioning in NFV-Enabled Networks Using Deep Reinforcement Learning," in IEEE Transactions on Network and Service Management, vol. 19, no. 3, pp. 3276-3289, Sept. 2022, doi: 10.1109/TNSM.2022.3159670.

[13] Y. Li, W. Liang and J. Li, "Profit Driven Service Provisioning in Edge Computing via Deep Reinforcement Learning," in IEEE Transactions on Network and Service Management, vol. 19, no. 3, pp. 3006-3019, Sept. 2022, doi: 10.1109/TNSM.2022.3159744.

[14] H. Sami, H. Otrok, J. Bentahar and A. Mourad, "AI-Based Resource Provisioning of IoE Services in 6G: A Deep Reinforcement Learning Approach," in IEEE Transactions on Network and Service Management, vol. 18, no. 3, pp. 3527-3540, Sept. 2021, doi: 10.1109/TNSM.2021.3066625.

[15] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey", Artificial Intelligence Review, vol. 55, no. 2, pp. 895–943, Feb. 2022.

[16] J. Chen, J. Chen and H. Zhang, "DRLEC: Multi-agent DRL based Elasticity Control for VNF Migration in SDN/NFV Networks," 2021 26th IEEE Asia-Pacific Conference on Communications (APCC), 2021, pp. 89-93, doi: 10.1109/APCC49754.2021.9609866

[17] P. Tam, S. Math, A. Lee, and S. Kim, "Multi-Agent Deep Q-Networks for Efficient Edge Federated Learning Communications in Software-Defined IoT", Computers, Materials & Continua, vol. 71, no. 2. 2022.

[18] A. Suzuki, R. Kawahara and S. Harada, "Cooperative Multi-Agent Deep Reinforcement Learning for Dynamic Virtual Network Allocation With Traffic Fluctuations," in IEEE Transactions on Network and Service Management, vol. 19, no. 3, pp. 1982-2000, Sept. 2022, doi: 10.1109/TNSM.2022.3149243.

[19] ONF, "Impact of SDN and NFV on OSS/BSS", ONF Solution Brief, March 1, 2016

[20] B. Naudts, M. Flores, R. Mijumbi, S. Verbrugge, J. Serrat and D. Colle, "A dynamic pricing algorithm for a network of virtual resources," 2016 IEEE NetSoft Conference and Workshops (NetSoft), Seoul, Korea (South), 2016, pp. 328-335, doi: 10.1109/NETSOFT.2016.7502429.

[21] D. A. Chekired, L. Khoukhi and H. T. Mouftah, "Decentralized Cloud-SDN Architecture in Smart Grid: A Dynamic Pricing Model," in IEEE Transactions on Industrial Informatics, vol. 14, no. 3, pp. 1220-1231, March 2018, doi: 10.1109/TII.2017.2742147.

[22] G. Laatikainen, A. Ojala, and O. Mazhelis, "Cloud Services Pricing Models", in Software Business. From Physical Products to Software Services and Solutions, 2013, pp. 117–129.

[23] W. Zhang, W. Lei, X. Chen, and S. Liu, "An open standards-based framework to integrate IMS and cloud computing", Services Transactions on Cloud Computing, vol. 2, pp. 28–40, 07 2014.

[24] F. J. Lin, W. Tsai, E. Cerritos, B. -T. Lin and W. -H. Hu, "Identification and analysis of charging factors in M2M communications," 2015 IEEE 2nd World Forum on

Internet of Things (WF-IoT), Milan, Italy, 2015, pp. 160-165, doi: 10.1109/WF-IoT.2015.7389045.

[25] F. J. Lin, W. Tsai, B. -T. Lin and W. -H. Hu, "Charging models for M2M communications," 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), Kanazawa, Japan, 2016, pp. 1-6, doi: 10.1109/APNOMS.2016.7737197.

[26] F. J. Lin, B. -Y. Chen, B. -T. Lin and W. -H. Hu, "Charging architecture for M2M communications," 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA, USA, 2016, pp. 123-128, doi: 10.1109/WF-IoT.2016.7845405.

[27] F. J. Lin, K. -L. Tsai, S. -Y. Song, W. -C. Hsu, Y. -T. La and W. -H. Hu, "A Tool for Defining Charging Models for M2M Communications," 2018 IEEE International Congress on Internet of Things (ICIOT), San Francisco, CA, USA, 2018, pp. 104-109, doi: 10.1109/ICIOT.2018.00021.

[28] R. Mijumbi, J. Serrat, J. -l. Gorricho, S. Latre, M. Charalambides and D. Lopez, "Management and orchestration challenges in network functions virtualization," in IEEE Communications Magazine, vol. 54, no. 1, pp. 98-105, January 2016, doi: 10.1109/MCOM.2016.7378433.

[29] TSI GS NFV-IFA 008 Ver. 2.4.1, "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Ve-Vnfm reference point - Interface and Information Model Specification", August 2017.

[30] ETSI GS NFV-IFA 013 Ver. 2.4.1, "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification", August 2017.

[31] ETSI GS NFV-INF 002 V1.2.1, "Network Functions Virtualisation (NFV); Architectural Framework", December 2014.

[32] ETSI GS NFV-INF 003 V1.1.1, "Network Functions Virtualisation (NFV); Infrastructure; Compute Domain", December 2014.

[33] ETSI GS NFV-INF 004 V1.1.1, "Network Functions Virtualisation (NFV); Infrastructure; Hypervisor Domain", January 2015.

[34] ETSI GS NFV-INF 005 V1.1.1, "Network Functions Virtualisation (NFV); Infrastructure; Network Domain", December 2014.

[35] ETSI GS NFV-IFA 007 Ver. 2.3.1, "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Or-Vnfm reference point - Interface and Information Model Specification", August 2017.

[36] ETSI GS NFV-IFA 005 Ver. 2.3.1, "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification", August 2017.

[37] ETSI GS NFV-IFA 006 Ver. 2.3.1, "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Vi-Vnfm reference point - Interface and Information Model Specification", August 2017.

[38] ETSI GR NFV-EVE 008 V3.1.1, "Network Function Virtualization Release 3; Charging; Report on Usage Metering and Charging Use Cases and Architectural Study", December 2017.

[39] X. Wang et al., "Deep Reinforcement Learning: A Survey," in IEEE Transactions on Neural Networks and Learning Systems, 2022, doi: 10.1109/TNNLS.2022.3207346.

[40] V. Konda and J. Tsitsiklis, "Actor-Critic Algorithms", in Advances in Neural Information Processing Systems, 1999, vol. 12.

[41] A. Shaghaghi, A. Zakeri, N. Mokari, M. R. Javan, M. Behdadfar and E. A. Jorswieck, "Proactive and AoI-Aware Failure Recovery for Stateful NFV-Enabled Zero-Touch 6G Networks: Model-Free DRL Approach," in IEEE Transactions on Network and Service Management, vol. 19, no. 1, pp. 437-451, March 2022, doi: 10.1109/TNSM.2021.3113054.

[42] J. J. Alves Esteves, A. Boubendir, F. Guillemin and P. Sens, "A Heuristically Assisted Deep Reinforcement Learning Approach for Network Slice Placement," in IEEE Transactions on Network and Service Management, vol. 19, no. 4, pp. 4794-4806, Dec. 2022, doi: 10.1109/TNSM.2021.3132103.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms", arXiv [cs.LG], 19-Jul-2017.