Received 30 December 2023, accepted 2 February 2024, date of publication 13 February 2024, date of current version 8 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3365635

RESEARCH ARTICLE

DQN Approach for Adaptive Self-Healing of VNFs in Cloud-Native Network

ARUNKUMAR ARULAPPAN[1], (Member, IEEE), ANIKET MAHANTI[2], (Senior Member, IEEE), KALPDRUM PASSI[3], (Senior Member, IEEE), THIRUVENKADAM SRINIVASAN[4], (Senior Member, IEEE), RANESH NAHA[5], (Member, IEEE), AND GUNASEKARAN RAJA[6], (Senior Member, IEEE)

[1]School of Computer Science Engineering and Information Systems, VIT University, Vellore 632014, India
[2]School of Computer Science, The University of Auckland, Auckland 1010, New Zealand
[3]School of Engineering and Computer Science, Laurentian University, Sudbury, ON P3E 2C6, Canada
[4]School of Electrical Engineering, VIT University, Vellore 632014, India
[5]Centre for Smart Analytics, Federation University Australia, Melbourne, VIC 3000, Australia
[6]NGNLab, Department of Computer Technology, Anna University, MIT Campus, Chennai 600044, India

Corresponding author: Kalpdrum Passi (kpassi@laurentian.ca)

The work of Kalpdrum Passi was supported by the School of Engineering and Computer Science, Laurentian University, Sudbury, Canada.

ABSTRACT The transformation from physical network function to Virtual Network Function (VNF) requires a fundamental design change in how applications and services are tested and assured in a hybrid virtual network. Once the VNFs are onboarded in a cloud network infrastructure, operators need to test VNFs in real-time at the time of instantiation automatically. This paper explicitly analyses the problem of adaptive self-healing of a Virtual Machine (VM) allocated by the VNF with the Deep Reinforcement Learning (DRL) approach. The DRL-based big data collection and analytics engine performs aggregation to probe and analyze data for troubleshooting and performance management. This engine helps to determine corrective actions (self-healing), such as scaling or migrating VNFs. Hence, we proposed a Deep Queue Learning (DQL) based Deep Queue Networks (DQN) mechanism for self-healing VNFs in the virtualized infrastructure manager. Virtual network probes of closed-loop orchestration perform the automation of the VNF and provide analytics for real-time, policy-driven orchestration in an open networking automation platform through the stochastic gradient descent method for VNF service assurance and network reliability. The proposed DQN/DDQN mechanism optimizes the price and lowers the cost by 18% for resource usage without disrupting the Quality of Service (QoS) provided by the VNF. The outcome of adaptive self-healing of the VNFs enhances the computational performance by 27% compared to other state-of-the-art algorithms.

INDEX TERMS Self-healing VNF, deep queue networks, operational automation, cloud-native deployment, ONAP, network intelligence.

I. INTRODUCTION

Software Defined Network (SDN) has engendered the virtualization of applications and networks, culminating in a cloud-native phase. Network Function Virtualisation (NFV) is being adopted in mobile networks with the deployment of 5G [1]. Each cellular network generation has led to the development of new business models [2]. Virtualized Infrastructure Manager (VIM) manages the entire lifecycle of the software and hardware, comprising the NFV Infrastructure (NFVI), and maintains a live inventory and allocation plan of both physical and virtual resources [3]. This approach follows the Management and Orchestration (MANO) function proposed by the European Telecommunication Standards Institute (ETSI). It results in simplifying service delivery and reduces cost with high-performance lifecycle management. Automation is the key to managing these complex networks and applications in various stages, such as Physical Network Function (PNF), Virtual Network Function (VNF), and Cloud-native Network Function (CNF)

The associate editor coordinating the review of this manuscript and approving it for publication was Farhana Jabeen.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

over a massive number of devices and different types of devices and services across many industries [4]. The cloud-native network is a software service that adheres to the design principle of cloud-native network functions without any hardware appliances attached to it. It includes CNF as the software component of a network function performed in a physical device and deployed on cloud-native data centres or cloud servers. The transformations in the underlying architecture and technologies lead to complexities in service lifecycle management [5]. Network Intelligence (NI) considers the embedding of Artificial Intelligence (AI) in future networks to fasten service delivery and operations, leverage Quality of Experience (QoE), and guarantee service availability, better agility, resiliency, faster customization, and security [6]. Network Intelligence technology allows the CSPs to capture the details of service and application-level VNF deployment in the network. The NI transforms how we optimize our network operations, significantly reducing operating costs. NI is envisioned to manage, pilot, and operate the forthcoming networking built upon SDN, NFV, and cloud [7].

Machine Learning (ML) for networking has enabled constant monitoring of a particular application of an ML tool that leads to optimizing the next-generation networks [8]. ML can exploit the hidden relationship between voluminous input data and complicated system outputs, especially for advanced techniques like deep learning. The other techniques, such as reinforcement learning, could further adapt the learning results and evolve automatically to the new environments [9]. The re-instantiating and benchmarking of complex services in automating the standard techniques are followed to deliver NFV solutions via ML techniques [10]. Predictive analytics powered by an AI engine enables forecasting results through leverage of data, sophisticated algorithms, advanced ML ability, and building on historical data [11]. AI algorithms are driven to monitor the present condition of equipment and help predict failure through data-driven techniques based on the analysis of preceding patterns [12]. These prediction and analysis techniques proactively fix issues with data centres, power lines, cell towers, and equipment present at the customer premises [13].

ML and AI can make edge networks more intelligent and show the way for next-generation networks. The AI can scale and operate the networks automatically in adopting new requirements in the model [14]. With this information model, a plug-and-play algorithm constitutes the changes to topology and route optimization as of the environmental changes. ML is a subset of AI that refers to collective data and pattern analysis, where the software system learns and adapts from continuous experiences over time.

The enhancement of NFV has matured with the introduction of advanced orchestration models to a whole new paradigm. The Existing system addressed the self-healing problem with on-policy methods such as Proximal Policy Optimization (PPO) uses a new type of gradient method where policy uses a neural network to implement easily on the network. Advantage Actor-Critic (A2C) with Generalized Advantageous Estimation (GAE) uses a trajectory where the values are stored and executed in the environment in calculating the estimated advantageous function. The Trust Region Policy Optimization (TRPO) calculates the weighted probability of the current policy and formulates the optimization of that policy using Kullback–Leibler (KL) divergence measurement using action and reward with a Monte-Carlo trajectory. The proposed DRL algorithm chooses the DQN mechanism over other PPO, A2C GAE and TRPO mechanisms because of its scalability issues. The DQN is proposed using Stochastic Gradient Descent (SGD) to calculate the weight of the network. The learning objective uses a target network and an evaluating network. The algorithms were implemented to work on VMs with Apcera installed and were trained with data collected through Apceras API, and the simulation was carried out through a cloud cluster. We could see that DQN/DDQN outperforms PPO+MC, PPO+GAE, TRPO+MC, and TRPO + GAE compared to the applied agents. The proposed algorithm is designed with zero human intervention, where the service provider reads through the provisioning of data through Deep Queue Networks (DQN) that decide which public cloud is being used to serve the customer better. The self-healing operation restarts the VNF applications whenever it detects that the application has crashed or is down, increasing the overall availability of VNF. This leads to building a resilient and fault-tolerant application that can handle changes and perform well in emergencies. The existing solutions in comparison to proposed methodologies are discussed, and the objective of the proposed system is described below:

- The proposed system analyses the problem of self-healing VNF through the DRL mechanism that helps in decision-making based on the DQN prediction model.
- VNFs have become a powerful base operation to use the DRN technique in chaining operations where service providers can enable/disable services based on QoS.
- The proposed DQN algorithm decides on available instances on VMs, voice services to deploy, type of hardware resources to use, and rapid enabling of services.
- The DQN mechanism minimizes the resource usage provided by VNF without disrupting the QoS. Adaptive Self-healing of VNF results in faster deployment cycles and lower CapEx and OpEx.

The Open Networking Automation Platform (ONAP) is an organized open-source cloud networking project built with the objective of developing a greater orchestration and automation platform. The major goals of this paper are listed below:

- Provides a real-time operational environment based on AI/ML policy-driven orchestration and automation techniques.

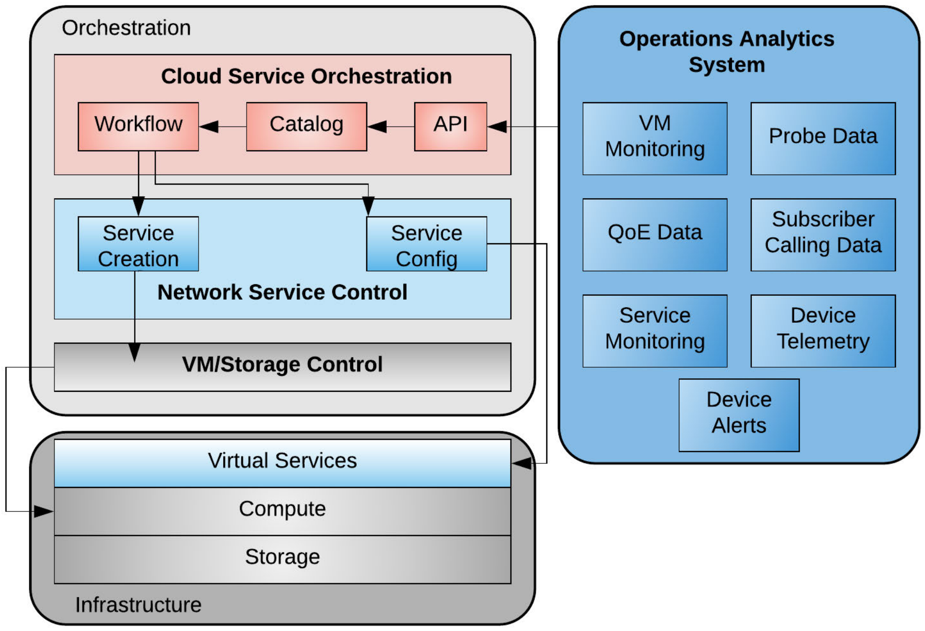


FIGURE 1. A network intelligent operations analytics system for NFV framework.

- The management of new services and their resources across the entire life cycle of the network
- ONAP addresses the industry problem of fragmentation as an initiative in taking the industry towards automation and convergence of enabling an ecosystem between open source and standards.
- Innovation through dis-aggregated services is deployed using VNF workloads in containers and virtual machines at the Edge that intensively push the envelope to 5G.

The major contributions of this paper are listed as follows:

- The six state-of-the-art Deep Reinforcement Learning (DRL) algorithms are examined with fundamental differences in their properties ranging from off-policy methods such as Deep Queue Networks (DQN) and Double Deep Queue Networks (DDQN) to on-policy methods such as Proximal Policy Optimization (PPO) Advantage Actor Critic (A2C).
- The different policies are compared to a baseline P-Controller in order to evaluate the performance with respect to simpler methods.
- The final policy applied by the agent shows considerable improvements over a simple control algorithm with respect to reward and performance with multiple experiments with varying loads and configurations tested.

The structure of the remaining paper is sectioned as follows: Section II describes the properties of DRL applied to VNFs in an NFV architecture for managing horizontal autoscaling. Section III deals with the system model and properties of applying DRL to the problem. Section IV describes how DRL solves the autoscaling problem by applying six state-of-the-art DRL agents to the proposed model. Section V presents the evaluation of results from various experiments and modeling. Lastly, section VI presents the conclusion and future work.

II. RELATED WORKS

Network virtualization is in place because of a massive influx of devices coming with IoT and 5G applications [15]. Hence, there is tremendous pressure on next-generation infrastructure. Today's network largely comprises purpose-built infrastructure, with each device containing its own management software [16]. The network of tomorrow will be deployed using NFV and SDN. Instead of a separate router, VPN, and firewall on three different hardware pieces, you can run all three on the same Intel architecture-based infrastructure with a network intelligent operations analytics system as represented in Fig. 1. When you add softwaredefined networking, you add a degree of intelligence and flexibility to your network provider that can greatly reduce operating costs.

Traditional infrastructure and new NFV-based infrastructure will need to coexist in the network for a number of years to come. For all this to work, NFV service assurance must be integrated with the service orchestrator responsible for managing the VNF lifecycle. Service assurance analytics is a key input to the orchestrator, driving remedial changes to services/VNFs. To know what open source has already done to transform the operating system (Linux), the virtualization Infrastructure (OpenStack), and big data (Hadoop) started to work in a collaborative manner rather than relying on proprietary solutions [17].

A. OBSERVABILITY BRINGS CLARITY TO CLOUD-NATIVE NETWORK

In microservices, the observability in the cloud-native network has become very important. Zero intervention automation and containerization are important concepts, but microservices bring transparency and assurance to

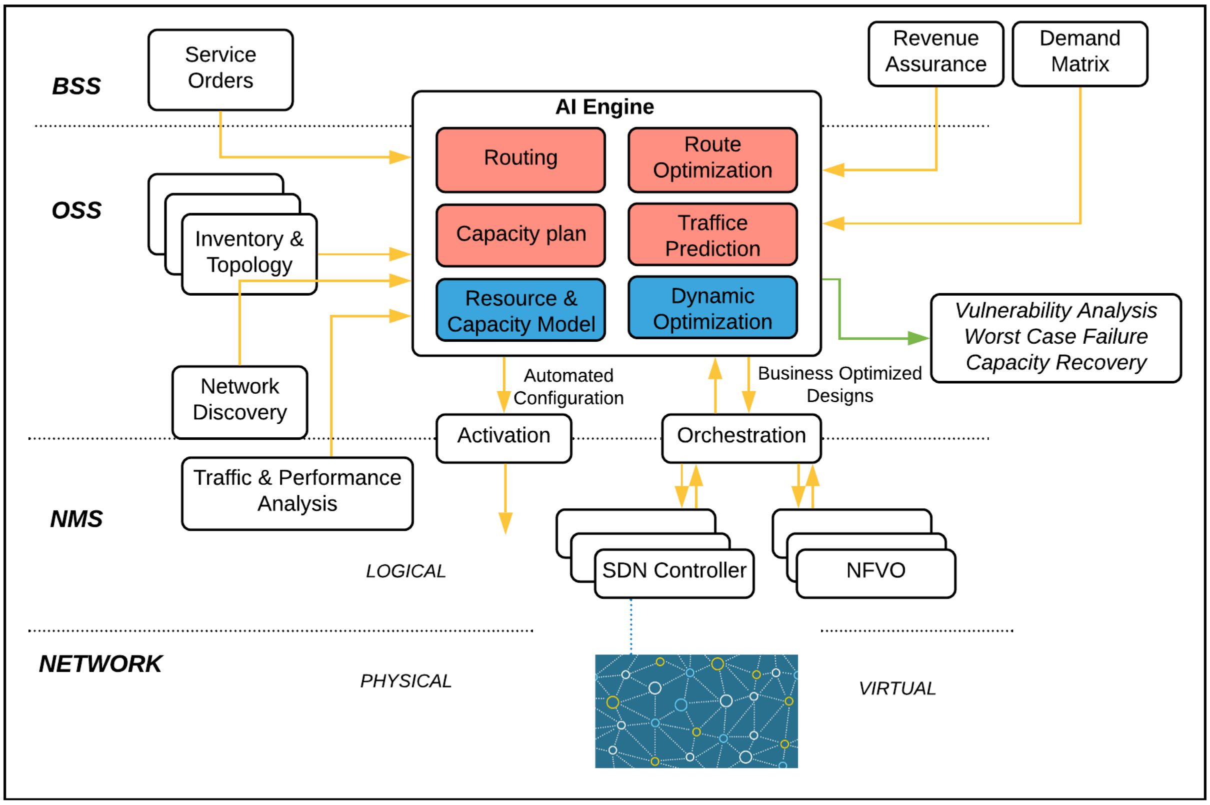


FIGURE 2. **AI Engine for next-generation cloud-native networks.**

network performance evaluation. In the telecom sector, the vendor-specific solution for service assurance observes the network traffic by pulling the data from the fibre optic connections via physical taps [18]. The observability of cloud-native networks has various properties, as listed below: How can we put a physical tap on a VM? How do we monitor the microservices when deployed in thousands on a single VM at a particular time? Whatever happens in physical infrastructure cannot be virtually possible in the cloud-native network [19]. The OpenSource cloud community has built robust ecosystem tools to provide service assurance compared to Fault Management, Configuration, Accounting, Performance, and Security (FCAPS) functionalities in the traditional physical telco cloud.

B. NETWORK AUTOMATION AND OPTIMIZATION

Due to the rapid increase in number of devices connected to the network, the communication networks have become complicated and hard to manage. The deployment of the latest technology like SD-WAN and its services like NFV and SDN has an enormous increase in complexity [20]. The advancement of automation techniques in network operation is leveraged by allowing network operators to use AI and ML technologies. Collecting network and device data usually predicts and pre-empts the possible issues in the network and applies fixes to optimize the network's reliability [21]. The service request on the customer portal holds detailed activities such as requests, complaints, interactions, and cross-channel portals. Quantitative and qualitative data are analyzed using various AI, ML, and deep learning techniques [22]. It also uncovers various trends and issues in performance (i.e.) based on the device, location, and time zone.

C. 5G WILL BE A TURNING POINT FOR NFV

The ETSI Zero-touch network and service management (ZSM) aims to enable largely autonomous networks driven by high-level policies and rules. These networks are capable of self-configuration, self-monitoring, self-healing, and self-optimization without any human intervention in the future for automated execution of overall operational processes [23]. This requires a new horizontal and vertical end-to-end architecture framework designed for closed-loop automation and optimized for data-driven machine learning and artificial intelligence engine for future generation cloud-native deployment as illustrated in Fig. 2. The ZSM architecture allows for managing the operational data by separating it from the management applications. The efficient access to cross-domain data exposure (e.g., topology, telemetry data) could be leveraged by intelligent network and server capabilities (e.g., AI, ML for automation) [24]. This architectural design helps enable closed-loop automation (i.e.) service assurance and process fulfillment at the network and service-management levels in the VNF self-healing process. This indeed results in automated decisions bounded by various policies and rules using a self-optimization decision-making mechanism. The data has become the lifeblood of bringing automation across cross-domain services. Rapid access to real-time management data has become a key process to AI, ML, and closed-loop automation [25]. The rise in data persistence from cross-domain data services allows data to be stored separately from the application and shared amongst other consumers. The data includes various attributes such as performance, trace, configuration, assurance, topology, and inventory data. The ZSM architectural design is meant for closed-loop automation and is optimized with data-driven ML and AI techniques [11].

Closed-loop is a feedback-driven operation that helps in continuous adaptation and optimization of network resource utilization and fulfillment of automated service assurance. The analytics are bounded by various policies and rules in determining the operational conditions for which automation conditions are allowed. Based on the insights from various research literature, the auto-scaling problem in VNFs is analyzed. The proposed DQN mechanism uses traditional model-free Q-learning to learn about a scaling policy and implements an auto-scaling solution based on the available legacy system in VNF. These approaches demonstrate good results based on the traditional rule-based auto-scaling solutions. The experiment evaluations are hard to compare as they operate in different environments for implementing RL solutions and modeling the auto-scaling problem. It is solved using the DRL mechanism with VNFs generates the performance measurements as inputs and executes scaling operations (VNF self-healing) based on those existing measurements. This paper has addressed six DRL algorithms to an NFV system architectural model for the autoscaling problem in ONAP platform using closed-loop automation orchestration as illustrated in Fig. 3. Virtual Probe analytics chooses the DQN approach rather than RPROP because of its scalability issues. The DQN proposes an update to the network through stochastic gradient descent of smaller batches sampled from stored observations.

III. SYSTEM MODEL

A. PROBLEM DEFINITION

NFV removes the dependency between the network functions hardware and software using VNF. To achieve faster deployment cycles, we need lower Capital Expenditure (CapEx) and dynamic utilization of cloud resources to lower Operating Expenses (OpEx) [2]. Reactive action is needed to allocate more resources during an unprecedented load rise during the modeling of VNF during self-healing operations with closed-loop orchestration. To make these changes happen, a VNF has to be managed by some system with action to be executed. Meanwhile, for now, this management is carried out manually with the analytics of data being generated by the VNFs. The systematic procedures in a management platform have a large complexity that examines the possibility of using DRL in networks by initiating the analytics and action selection. The collective input dataset comprises an internal and external load of VNFs being deployed on the data centre for four days of data collection. The output comprises the average CPU load of 8 VNFs measured with respect to the packets sent and received from the live network. The computation is performed using a GitHub TensorForce based on Google Cloud SDK with a local installation of Python 3.5 on the Linux platform. Six state-of-the-art algorithms are compared, and computations are performed using this dataset.

Reinforcement Learning (RL) can take action toward an environment by studying self-learning and adaptable agents that progress to maximize the rewards resulting from the actions made. The actions a_t are taken by the agent in an environment that is based on the current state s_t, reward r_t, and policy π. The policy evaluation in the intermediate state s_{t+1}, reward r_{t+1} and its next action are carried out in a closed-loop, which turns the interaction between the agent-environment. In essence reinforcement learning is an optimization problem to maximize the reward over time. Wherein the optimization is based on the state's environment, reward influence, and actions performed in changing the states and the reward. The agent in an environment exploits the current knowledge and maximizes the rewards by taking greedy actions, pushing forward the acquisition of new knowledge by exploring actions. Lastly, the agent adapts to its policy with respect to the dynamics of state transition over time.

B. VNF MODELLING DURING SCALING (SELF-HEALING) OPERATIONS

In this section, a VNF is modeled with horizontal scaling operations. The states are generic in the nature of the environment. Translating the states to different counters with various kinds of VNFs is possible. The states are chosen, and the measurements are based on the three classes L^{ext}, L^{in}, p^{QoS}. The total of three observable values are divided into loads l, errors e and allocation u. The loads are considered either as internal L^{in}, or external L^{ext}, and the errors p^{QoS}. The observable values are represented in Table 1. To strengthen MDP properties, the model with values k=5 of one state is combined with the last observed values into one state. So, the delay in observation is measured three times for every time step by the RL agents that are being applied to the model. The state at time t is observed with values with k=5, which holds 15 dimensions in a state as given in eq. 1.

$$s_t = \begin{pmatrix} u_t^{vm} & u_{t-1}^{vm} & u_{t-k}^{vm} \\ l_t^{tr} & l_{t-1}^{tr} & l_{t-k}^{tr} \\ l_t^{pr} & l_{t-1}^{pr} & l_{t-k}^{pr} \end{pmatrix} \tag{1}$$

The scaling action possible to perform on the model, and those actions can be executed on the model via an external entity such as an RL agent,

$$a_t \in \{1, 0, -1\} \forall t \tag{2}$$

The values where a_t=1 represent a scale-out function and $a_t = -1$ represent a scale-in function. Whenever a scaling

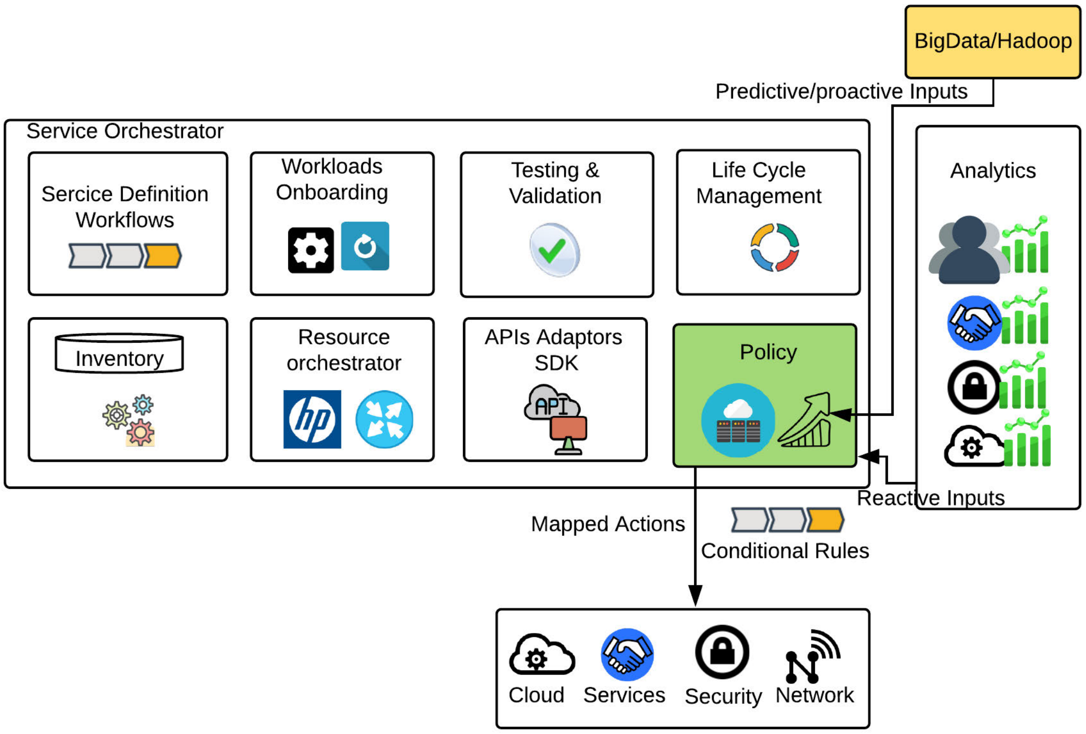


FIGURE 3. Closed-loop orchestration in ONAP for VNF Self-healing process.

action with $a_t \neq 0$ model is performed, the model is set to a scaling state, where the distributed loads and their work have been reflected in state-transition dynamics, which is described in eq. 2. The present state and its timeframe n_{req}, with 3 and 5-time steps, is randomized by leading from the real system. In reality, a VNF configuration and its scaling vary based on the allocation of the number of VMs by the VNF. In the case of dealing with the model that is configurable, with respect to modeled scaling using the parameters N^{vm}_{min} and N^{vm}_{max}. Therefore, the result of those scaling actions becomes limited when N^{vm}_{min} and N^{vm}_{max}.

C. DYNAMICS OF STATE-TRANSITION IN DISCRETE MODEL

The state transition determines in detail the internal calculations and samplings that are used in the state transitions traversed in time with the discrete model [10]. Based on a state transition, the overview of an implemented model and its internal dependencies is accustomed at DQN/DDQN for the self-healing of VNFs.

1) CALCULATION OF U^{VM}_T

The VM utilization rate is calculated through a normalized measurement with respect to the minimum and maximum number of VMs.

$$u^{vm}_t = \frac{N^{vm}_t - N^{vm}_{min}}{N^{vm}_{max} - N^{vm}_{min}} \tag{3}$$

2) SAMPLING OF PROCESS LOAD L^{PR}_T

The whole sample load processing model is driven by generating external traffic with l^{pr}_t. Based on the generated traffic load, the current process load is calculated in two ways based on the scaling state. In a real VNF, the scaling action generates additional load on the VM by means of internal processes in re-distributing data and leads to the preparation of a new VM configuration. With this, the process load is calculated as given below in eq. 4.

$$l^{pr}_t = \begin{cases} l^{tr}_t + U(-\eta_{pr}, \eta_{pr} + \eta_{scale}), & \textit{if scaling state} \\ l^{tr}_t + U(-\eta_{pr}, \eta_{pr}), & \textit{else} \end{cases} \tag{4}$$

Random variable U(a,b) is drawn between (a, b) from a uniform distribution, where $\eta_{pr}, \eta_{scale}>0$

3) SAMPLING LOAD OF L^{CPU}_T, L^{RAM}_T

The CPU and RAM load reflects the traffic load and how it gets processed and distributed between VMs. The initial values of CPU and RAM have been inspired from the real system l^{cpu}_0, l^{ram}_0 and varied with the number of processes and their noise [48]. The re-distribution also holds the load between VMs and scaling in/out processes. The share update rule, where $i \in \{cpu, ram\}$ is given in eq. 5.

$$l^t_i = g_{sc}(l^i_t)+\alpha_i(l^{pr}_t - l^{pr}_{t-1})+\beta_i(l^{tr}_t - l^{tr}_{t-1}) + \eta_i U(-1, 1) \tag{5}$$

$$g_{sc}(l^i_t) = \begin{cases} l^i_{t-m}\frac{N^{vm}_{old}}{N^{vm}_{new}}(\frac{1}{n} + U(-0.1, 0.1)), & \textit{if scaling state} \\ l^i_{t-1} \end{cases} \tag{6}$$

TABLE 1. List of Symbols.

Symbol	Description
L^{ext}	External load of the VNF
L^{int}	Internal load of the VNF
p^{QoS}	QoS degradation
u_t^{vm}	VM utilization normalized at time 't'
l_t^{pr}	process load normalized at time 't'
l_t^{tr}	traffic load normalized at time 't'
l_t^{cpu}	Average CPU load of the VMs utilized at time 't'
l_t^{ram}	Average RAM load of the VMs utilized at time 't'
e_t^{hw}	Magnitude of hardware error at time 't'
e_t^{lat}	Magnitude of latency error at time 't'
N_t^{vm}	VM utilization scaled in at time 't'
N_{max}^{vm}	Maximum number of VMs
N_{min}^{vm}	Minimum number of VMs
η_{pr}	Noise level on process load
η_{scale}	Added process noise due to scaling
$g_{sc}(l_t^i)$	Added/removed load between VMs
k^{lat}	Gain on latency error
P_e^{lat}	Probability of latency error on bad state
P_e^{hw}	Probability of hardware error in bad state
$Q(s, a; \theta)$	Q-function approximation using a neural network with weights θ
$L^{DQN}(s, a; \theta)$	Loss function to minimize θ when training a DQN agent
$\pi(a\|s, \theta)$	Representation of policy probability distribution of action 'a' in state 's' given neural network weights θ
$L^{TRPO-A}(\theta)$	Loss function to maximize θ when training a TRPO agent with Monte Carlo trajectories
$L^{PPO-A}(\theta)$	Loss function to maximize θ when training a PPO agent with Monte Carlo trajectories
$L^{TRPO-AC}(\theta)$	Loss function to maximize θ when training an Advantage Actor-Critic agent with TRPO and GAE
$L^{PPO-AC}(\theta)$	Loss function to maximize θ when training an Advantage Actor-Critic agent with PPO and GAE
$\theta_{\pi_{old}}$	Future discounted reward (Q) given actions sampled from old policy
$KL(\pi)$	Kullback-Leibler divergence measurement between current and old policy distribution
$\hat{G}_t$	Future discounted reward (Q) calculated from Monte Carlo trajectory
$V(s_t; w)$	Critic as representation of value functions using a neural network with weights w
$\hat{A}_t^{GAE(\lambda)}$	Estimation of advantage function using GAE with eligibility tracy λ
δ_t	Temporal difference (TD) error at time 't'

Here $g_{sc}(l_t^i)$ is the load add/remove due to re-distribution when the scaling action is active, m defines the number of time steps taken in a system in scaling state since the last scaling action $a_t \neq 0$. Further calculations are performed in a real system based on the CPU and RAM measurements mentioned in eq. 6.

$$l_i^t = \begin{cases} 1, & if\ l_i^t > 1 \\ l_{min}^i, & if\ l_i^t < l_{min}^i \\ l_i^t, & else \end{cases} \tag{7}$$

4) ERROR SAMPLING

The agent resource optimization is carried out to avoid the negative impact the model provides. The measurement of a negative impact is said to be hidden and sampled with probability P, wherein in real VNF, there seems to be a high-risk factor [6]. The errors are defined based on the reasoning

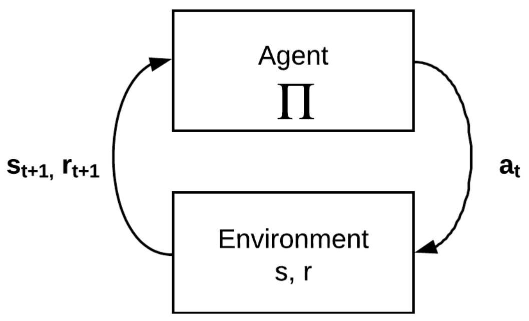


FIGURE 4. A systematic reinforcement learning schema for self-learning agents.

as listed below in eq. 7 and eq. 8.

$$e_t^{lat} = \begin{cases} k^{lat}(l_t^{tr} - u_t^{vm}), & if\ l_t^{tr} \geq u_t^{vm}\ and\ U(0,1) > (1 - P_e^{lat}) \\ 0, & else \end{cases} \tag{8}$$

$$e_t^{hw} = l_t^{pr}(e^{cpu} + e^{ram}) \tag{9}$$

where,

$$e^{cpu} = \begin{cases} \frac{1}{2}, & l_t^{cpu} > 0.9\ and\ U(0,1) > (1 - P_e^{hw}) \\ 0, & else \end{cases}$$

$$e^{ram} = \begin{cases} \frac{1}{2}, & l_t^{ram} > 0.95\ and\ U(0,1) > (1 - P_e^{hw}) \\ 0, & else \end{cases}$$

IV. PROPOSED SYSTEM

The autoscaling problem is solved using the DRL mechanism with VNFs generating the performance measurements as inputs and executing scaling operations (VNF self-healing) based on those existing measurements as represented in Fig. 4. This paper has addressed six DRL algorithms to an NFV system architectural model for the autoscaling problem in ONAP platform using closed-loop automation orchestration. Virtual Probe analytics chooses the DQN approach rather than RPROP because of its scalability issues. The DQN proposes an update to the network through stochastic gradient descent of smaller batches sampled from stored observations. This paper addresses changes in comparison with the previously developed NFQ:

1) Using Stochastic Gradient Descent (SGD) as an update method for calculating the weight of the network.
2) Collective sampling of transitions using random mini-batches for updates; the method is called experience replay.
3) Two networks are used during learning (i.e.) a target network Q(s,t;$\hat{\theta}$) and an evaluating network Q(s,t;θ)

The two network learning approaches advance in strengthening the learning stability, where the first one is meant for updating weights, and the latter is meant for evaluating the value as shown in eq. 9. In every C time step, it held constant in between two networks (i.e.) target network Q(s,t;$\hat{\theta}$) and

Algorithm 1 DQN/DDQN With Inspiration

```
1:  Initialize memory D
2:  Initialize Q-network with random weights $\theta$
3:  Initialize the target Q-network with weights $\theta'$=$\theta$
4:  for episode = [1,...,M] do
5:      Sample initial state s1 from the environment
6:      for t = [1, T] do
7:          //Execute action with $\epsilon$-greedy approach
8:          $a_t = \begin{cases} random\ action\ with\ probability\ \epsilon \\ arg\ max_a Q(s_t, a; \theta), & else \end{cases}$
9:          Execute $a_t$ on the environment and observe $r_t, s_{t+1}$
10:         Store transition $\{s_t, a_t, r_t, s_{t+1}\}$ in D
11:         //Update network with experience replay
12:         Sample minibatch B=$\left\{\left(s_T^{(1)}, a_T^{(1)}, r_T^{(1)}, s_{r+1}^{(1)}, \ldots, s_T^{(n)}, a_T^{(n)}, r_T^{(n)}, s_{r+1}^{(n)}\right)\right\}$
13:         $y^{(i)} = \begin{cases} r_T^{(i)}, & ifs_{r+1}^{(i)} = terminal \\ r_T^{(i)} + \gamma_a^{max} Q(s_{r+1}^i, a; \hat{\theta}), & elseifDQN \\ r_T^{(i)} + \gamma Q(s_{r+1}^i, arg\ max_a Q(s_{t+1}, a; \theta); \hat{\theta}), & elseifDDQN \end{cases}$
14:         e=$(y^{(i)} \ldots . y^{(n)} - (\theta(s_T^{(1)}, a_T^{(1)}; \theta) \ldots . Q(s_T^{(n)}, a_T^{(n),\theta}))$
15:         $L(\theta) = e.e^r$
16:         Perform a minimizing step in $\theta$ on L($\theta$)(*)
17:         //Reset target network
18:         if $((episode - 1)T + t)\%C = 0$ then
19:             $\hat{\theta} = \theta$
20:         end if
21:     end for
22: end for
23: (*)= e.g., with Stochastic gradient descent or Adam optimization
```

Q(s,t;θ). The steps that occur in between synchronizing the network weights Q(s,t;θ) are updated with experience replay and *SGD*.

$$L^{DQN}(s_t, a_t; \theta) = \left(r_t + \gamma_a^{max} Q(S_{t+1}, a; \hat{\theta}) - (S_{t+1}, a; \hat{\theta})\right)^2 \tag{10}$$

A. TRUST REGION POLICY OPTIMIZATION

The ratio of a weighted probability is maximized with a constraint with respect to θ, between the current policy and change in that policy is formulated as an optimization problem as shown in eq. 10.

$$\max_\theta L_t^{TRPO-A}(\theta) = \hat{E}_t\left\{\frac{\pi(a_t|s_t, \theta)}{\pi(a_t|s_t, \theta_{old})}\theta_{\pi_{old}}(s_t, a_t)\right\}$$
$$= \frac{1}{T}\sum_{T=1}^{T}\frac{\pi(a_t|s_t, \theta)}{\pi(a_t|s_t, \theta_{old}})\hat{\theta}_{\pi_{old}}(s_t, a_t) \tag{11}$$

The Kullback-Leibler divergence measurement is one way of calculating the future discounted reward, $\hat{G}_t$ of approximation $Q_{\pi_{old}}(s_t, a_t)$ with a trajectory experienced in interaction with the environment, i.e., Monte Carlo trajectories.

B. DQN AND DDQN

The DQN algorithm is proposed with improvements fed with better learning capabilities for handling autoscaling of Self-healing of VNFs in a network. The influential improvements that are driven by the DQN algorithm are Double DQN (DDQN), experienced replay based on prioritization, and Dueling networks. The overview of an implemented model and its internal dependencies, by arrow representation, are given in Fig. 5. The DDQN has become the simplest method to implement as it requires only a minimal change to the loss function represented in eq. 11. The action values proved that DDQN reduces overestimations compared to standard DQN. The pseudocode with experience replay for DQN and DDQN is mentioned in algorithm 1.

$$L^{DDQN}(s_t, a_t; \theta) = \Bigg(r_t + \gamma Q\Big(s_{t+1}, arg \max_a Q(s_{t+1}, a; \theta)\hat{\theta} - Q(s_t, a_t; \theta)\Big)\Bigg)^2 \tag{12}$$

C. PROXIMAL POLICY OPTIMIZATION (PPO)

The PPO algorithm holds a new type of gradient method where policy uses a neural network for policy optimization based on TRPO logic, but it's easy to implement on

Algorithm 2 PPO With Monte Carlo Trajectories

```
1:  Initialize the $\pi$-network as Actor with weights $\theta$
2:  Initialize memory D
3:  for episode = [1,...,M] do
4:      Sample initial state s1 from the environment
5:      while $s_t \neq terminal$ do
6:          //Save transitions from T steps with the old policy
7:          for t = [1,...,T] do
8:              Sample $a_t$ from $\pi\ (a_t|s_t;\theta)$
9:              Execute $a_t$ and store $\{s_{t+1}, r_t\}$ in D
10:         end for
11:         // MC Estimate of future discounted reward
12:         for t = [1,...,T] do
13:             $\hat{G}_t = \sum_{i=0}^{T-t} \gamma^i r_{t+i}$
14:         end for
15:         //Update policy network
16:         $L^{\pi}(\theta') = \frac{1}{T}\sum_{T=1}^{T} \begin{cases} min\left(\frac{\pi(a_T|s_T;\theta')}{\pi(a_T|s_T;\theta)'}, 1+\epsilon\right)\hat{Q}_T, & if \hat{Q}_T > 0 \\ max\left(\frac{\pi(a_T|s_T;\theta')}{\pi(a_T|s_T;\theta')'}, 1-\epsilon\right)\hat{Q}_T, & else \end{cases}$
17:         Perform a maximizing step in $\theta$ on $L^{\pi}(\theta')$ (*)
18:     end while
19: end for
20: (*) = e.g., with Stochastic gradient ascent or Adam Optimization
```

the network. Using KL- KL-divergence measurement, PPO optimizes the ratio between different policies to attenuate the ratio difference, as mentioned in eq. 12. Stable update and implementation are performed with PPO in algorithm 2, resulting in faster convergence. AI-driven Closed-loop network architecture is a feedback-driven operation that helps in continuous adaptation and optimization of network resource utilization and fulfillment of automated service assurance. The analytics are bounded by various policies and rules in determining the operational conditions for which automation conditions are allowed. Based on the insights from various research literature, the auto-scaling problem in VNFs is analyzed. The proposed DQN mechanism uses traditional model-free Q-learning to learn about a scaling policy and implements an auto-scaling solution based on the available legacy system in VNF. These approaches demonstrate good results based on the traditional rule-based auto-scaling solutions. The experiment evaluations are hard to compare as they operate in different environments for implementing RL solutions and modeling the auto-scaling problem. (13)–(15), as shown at the bottom of the page.

D. ADVANTAGE ACTOR-CRITIC (A2C) WITH GAE

The actor-critic method approximates the advantage function as Generalized Advantageous Estimation (GAE), as mentioned in algorithm 3.

The trajectory T is stored and executed in the environment in calculating the estimated advantage function as

$$\hat{A}_t^{GAE(\lambda)} = \sum_{T=0}^{T} (\lambda_\gamma)^T \delta_{t+T} \tag{16}$$

where,

$$\delta_t = r_t + \gamma V(s_{t+1}; w) - V(s_t; w)$$

$$L_t^{PPO-A}(\theta) = \hat{E}_t \begin{cases} min\left(\frac{\pi(a_T|s_T;\theta')}{\pi(a_T|s_T;\theta)'}, 1+\epsilon\right) Q_{\pi_{old}}(s_t, a_t), & if Q_{\pi_{old}}(s_t, a_t) > 0 \\ max\left(\frac{\pi(a_T|s_T;\theta')}{\pi(a_t|s_t;\theta')'}, 1-\epsilon\right) Q_{\pi_{old}}(s_t, a_t), & else \end{cases} \tag{13}$$

$$= \frac{1}{T}\sum_{T=1}^{T} \begin{cases} min\left(\frac{\pi(a_T|s_T;\theta')}{\pi(a_T|s_T;\theta)'}, 1+\epsilon\right) Q_{\pi_{old}}(s_t, a_t), & if Q_{\pi_{old}}(s_t, a_t) > 0 \\ max\left(\frac{\pi(a_T|s_T;\theta')}{\pi(a_T|s_T;\theta')'}, 1-\epsilon\right) \hat{Q}_{\pi_{old}}(s_T, a_T), & else \end{cases} \tag{14}$$

$$s.t\ \hat{E}_t\{KL(\pi(.|s_t, \theta), \pi(.|s_t, \theta_{old})\} \leq \alpha \tag{15}$$

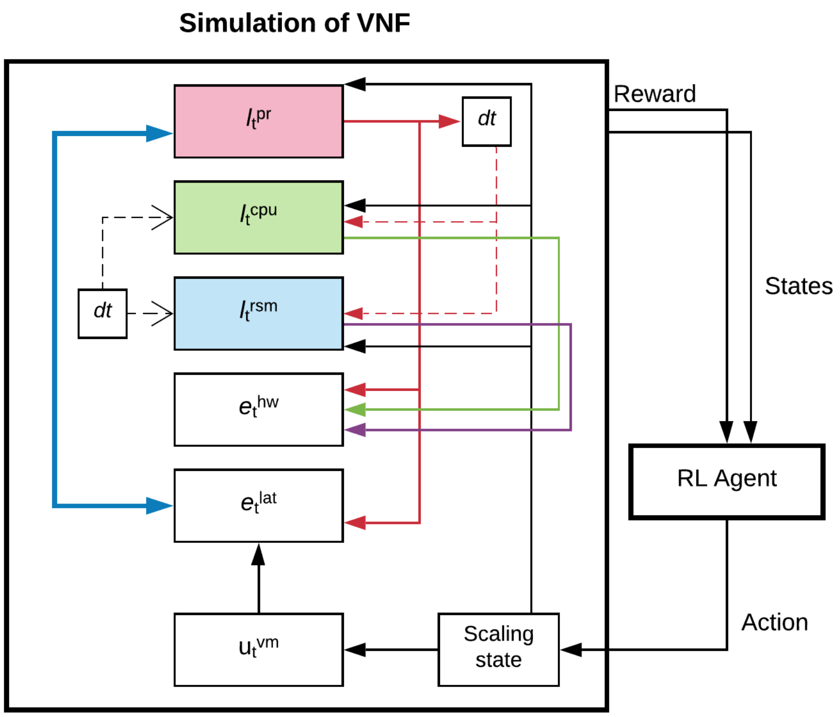


FIGURE 5. State transition model driven by an external load represents real VNF traffic.

Here $V(.)$ is represented using a neural network as a value function. The implementation of both PPO and TRPO with GAE estimates by replacing $Q_{\pi_{old}}(s_t, a_t)$ with

$$A_{\pi_{old}}(s_t, a_t) \approx A_{\pi}^{G\hat{A}E(\lambda)}((s_t, a_t) \tag{17}$$

Advantage function methods of A2C and optimization problems for TRPO and PPO changes are mentioned in eq. 13 and eq. 14. (18)–(20), as shown at the bottom of the next page.

V. PERFORMANCE EVALUATION

The evaluation section helps analyze the task and implements a discrete-time model for VNFs to capture the characteristics of the scaling operation. It also examines the robustness and generalization during the learning process of six state-of-the-art DRL algorithms. The auto-scaling or self-healing of VNF agents is based on workload management on virtual machines and happens in an automatic manner without any human intervention.

TABLE 2. DQN and DDQN parameters.

Parameter	Value
Q-Network	[200,200] ReLu
Target sync frequency	10000
Adam stepsize	2e-0.4
Batchsize	70
Discount	0.99
Reward normalization	True
Epsilon start	1
Epsilon end	0.05
Epsilon anneal steps	50000

TABLE 3. PPO/TRPO GAE parameters.

Parameter	Value
Value-Network	[20,20] ReLu
Adam stepsize	1e-5
λ	0.985

A. APPLICATION PERFORMANCE MANAGEMENT

1) TENSORFORCE

TensorForce is a new Python framework where the agents are tuned and trained. TensorForce is based on TensorFlow, allowing for agent modularization and a control logic separation from the environment. The parameter model has a separate component, hence implemented with a standardized interface and read by all the agents. For specific autoscaling problems, the agent implementation has given off-the-shelf components in TensorForce and needs to be tuned based on the implemented model. The list of different parameters used in the model for various agents is described in Tables 2, 3, 4, and 5.

The trained agent evaluation is modified to evaluate the agent's performance on robustness with respect to environmental changes. The parameters listed in the table are fine-tuned to show the dynamics of performing scaling actions to different workloads. Furthermore, the probability

Algorithm 3 PPO Advantage Actor-Critic With GAE (λ)

```
1:  Initialize V-network as Critic with weights w
2:  Initialize the $\pi$-network as Actor with weights $\theta$
3:  Initialize memory D
4:  for episode = [1,...,M] do
5:    Sample initial state $s_1$ from environment
6:    while $s_t \neq$ terminal do
7:      //Save transitions from T steps with the old policy
8:      for t = [1,...,T] do
9:        Sample $a_t from \pi(.|s_t;\theta)$
10:       Execute $a_t$ and store $\{s_{t+1}, r_t\}$ in D
11:     end for
12:     //Update the Critic with TD(1) learning
13:     $L^v(w) - \sum_{T=0}^{T}\left(\left(V(s_t;w)\right) - sum_{n=t}T\gamma^n r(t+n)\right)^2$
14:     Perform a minimizing step in w on $L^V(w)$ (*)
15:     //Estimate the advantage function $\hat{A}_t$ using $GAE(\lambda)$
16:     for t = [1,...,T] do
17:       With $\delta_t = r_t + \gamma V(s_{t+1};w) - V(s_t;w)$
18:       $\hat{A}^t = \sum_{T=0}^{T-t}(\lambda\gamma)^t \delta_{t+T}$
19:     end fort
20:     //Update the actor
21:     $L^\pi(\theta') == \frac{1}{T}\sum_{T=1}^{T}\begin{cases} min\left(\frac{\pi(a_T|s_T;\theta')}{\pi(a_T|s_T;\theta)'}, 1+\epsilon\right)\hat{A}_T, & if\hat{A}_T > 0 \\ max\left(\frac{\pi(a_T|s_T;\theta')}{\pi(a_T|s_T;\theta')'}, 1-\epsilon\right)\hat{A}_T, & else \end{cases}$
22:     Perform a maximizing step in $\theta$ on $L^\pi(\theta')$ (*)
23:   end while
24: end for
```

of latency is set to 1 for the time it takes to call or send a package. Learning optimal scaling becomes easier in comparison with stochastic states of performance degradation. The reward signal weights are carefully tuned as of policy, and good behavior is rewarded. The rewards define the traits of scaling up at a high load and scaling down at a low load.

2) TRACING

The tracing is a third-party open-source tool where virtual Probes are meant to record data logs present within the microservices. The virtual Probes have collective logging and events, thus capturing the data from the microservices of every CNF and user endpoints such as TCP RTT, retransmission rate, and DPI inspection. Unlike physical probes,

$$\begin{aligned} max_\theta L^{TRPO-AC}(\theta) &= \hat{E}_t\left\{\frac{\pi(a_t|s_t,\theta)}{\pi(a_t|s_t,\theta_{old})}A_{\pi_{old}}(s_t,a_t)\right\} \\ &= \frac{1}{T}\sum_{T=1}^{T}\frac{\pi(a_t|s_t,\theta)}{\pi(a_t|s_t,\theta_{old})}\hat{A}_{\pi_{old}}^{GAE(\lambda)}(s_t,a_t) \\ s.t \quad & \hat{E}_t\{KL(\pi(.|s_t,\theta),\pi(.|s_t,\theta))\} \leq \alpha \end{aligned} \tag{18}$$

$$L^{PPO-A}(\theta) = \begin{cases} min\left(\frac{\pi(a_T|s_T;\theta')}{\pi(a_T|s_T;\theta)'}, 1+\epsilon\right)A_{\pi_{old}}(s_t,a_t), & ifA_{\pi_{old}}(s_t,a_t) > 0 \\ max\left(\frac{\pi(a_T|s_T;\theta')}{\pi(a_t|s_t;\theta')'}, 1-\epsilon\right)A_{\pi_{old}}(s_t,a_t), & else \end{cases} \tag{19}$$

$$= \frac{1}{T}\sum_{T=1}^{T}\begin{cases} min\left(\frac{\pi(a_T|s_T;\theta')}{\pi(a_T|s_T;\theta)'}, 1+\epsilon\right)A_{\pi_{old}}^{GAE(\lambda)}(s_t,a_t), & ifA_{\pi_{old}}^{GAE(\lambda)}(s_t,a_t) > 0 \\ max\left(\frac{\pi(a_T|s_T;\theta')}{\pi(a_T|s_T;\theta')'}, 1-\epsilon\right)A_{\pi_{old}}^{GAE(\lambda)}(s_T,a_T), & else \end{cases} \tag{20}$$

TABLE 4. PPO parameters.

Parameter	Value
Policy-Network	[200,200] ReLu
Adam stepsize	1e-4
Batchsize	300
Discount	0.99
Clipping ratio	0.2
Reward normalization	True

TABLE 5. TRPO parameters.

Parameter	Value
Policy-Network	[200,200] ReLu
α	4e-4
Batchsize	300
Discount	0.99
Reward normalization	True

virtual Probes don't involve negative network performance and generate a wealth of data supporting AI and ML. The dataset for evaluation is from the customers as the VNFs are deployed at the customer premises to interact with other network functions. The dataset is from the Telecom Regulatory Authority of India (TRAI), Government of India, with the assistance of an Indian telecom operator in order to visualize and analyze. The complete dataset comprises the internal and external load of a VNF deployed during four data collection days. The average CPU load for 8 VNFs is measured with respect to the packets sent and received from the live network. The computation is performed using a GitHub TensorForce based on Google Cloud SDK with a local installation of Python 3.5 on the Linux platform.

B. EXPERIMENTS AND RESULTS

The experimental result is based on the implementation evaluation carried out in the previous section. The focus is on the agents pertaining to reward, the robustness/generalization of policies, and convergence rates applied to the model. Self-Healing using DRL on model consists of 140 timesteps that define one episode as agents trained on the same traffic pattern. The pattern was a sinusoidal function to represent the different dynamics of the different episodes encountered during traffic with a period of 140 timesteps and varied between 0.10 and 0.95 in load. The evaluation is divided into two phases, as listed below:

1) Training phase evaluation
2) Learned policy evaluation

1) EXPERIMENT 1: GATHERING STATISTICS OF TRAINING

The training agent is evaluated based on how long each agent, based on average, takes to converge. This has indeed become an important agent measurement being implemented and deployed on VNFs since they are known as slow real-time systems. The average reward in Fig. 6 shows that DQN and DDQN show the concrete result of having the highest

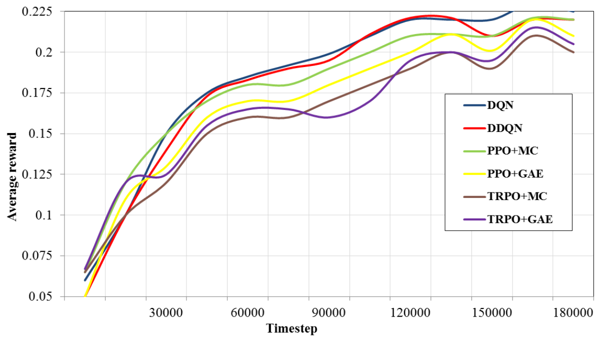


FIGURE 6. Average reward per timesteps for agents.

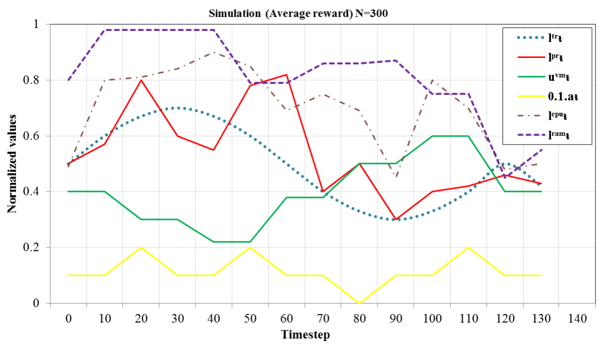


FIGURE 7. Average reward for traffic pattern.

converged reward amongst the agents. The graph displays that PPO A2C has a fast initial increase in reward and stands tall until 60000 timesteps. The convergence rate of all agents happens at approximately 180000 timesteps in 70 days on a real-time system model. The DQN and DDQN have the most stable value of convergence and have a small margin of variance when they are stable. The interesting thing to note is that TRPO+GAE has not reduced the variance compared to TRPO+MC to see an increase in converged reward. PPO shows a greater impact on being applied A2C with faster convergence, smaller variance, and higher reward.

The training results analyze as to why one algorithm performs better than the others. Initially, the episodic rewards show how well the algorithm performs during training. The result shows how each algorithm explores new paths and solutions for 1000 episodes. We used a ϵ -greedy policy with ϵ decreasing after each episode. While ϵ decreases, so does the number of random actions taken, and the algorithms instead choose actions to maximize the reward. The concentration of higher rewards in the mean rewards for DQN than the comparative algorithms are listed in Table 7. These rewards are the total returns from the reward functions after 285 steps without any random actions. The DQN/DDQN has the best training mean and end reward for 1000 episodes; hence, the rest of the algorithms fall behind. From the simulation performed and from the table, we found that DQN/DDQN is the best among the rest of the algorithms.

TABLE 6. Episode reward.

Algorithm	Mean Reward	End Reward	Simulation Reward
DQN/DDQN	-142.36	-85.39	-83.13
PPO+MC	-141.26	-87.99	-86.43
PPO+GAE	-144.40	-89.23	-88.34
TRPO+MC	-139.93	-88.43	-87.44
TRPO+GAE	-140.44	-86.83	-86.84

TABLE 7. Episode cost.

Algorithm	Mean Cost	End Cost
DQN/DDQN	0.565	0.545
PPO+MC	0.576	0.515
PPO+GAE	0.652	0.610
TRPO+MC	0.688	0.710
TRPO+GAE	0.645	0.630

TABLE 8. Episode speed.

Algorithm	Time (s)	% faster than TRPO+GAE
DQN/DDQN	987s	27.0%
PPO+MC	997s	26.0%
PPO+GAE	1015s	25.8%
TRPO+MC	1256s	24.6%
TRPO+GAE	1452s	———

Compared to episode rewards, the episode cost covers running the VM cloud clusters online. The comparative analysis of various algorithms is carried out based on the mean and end costs after training for 1000 episodes. The training reward of DQN/DDQN outperforms the rest of the algorithms, as listed in Table 8. The VNF agents had acted optimally with respect to the learned policies during training, and each experiment was carried out 300 times with different random seeds and statistics, and it was saved and compared for each VNF agent. During each experiment, the statistics consist of average reward count with percentiles, average resource utilization, maximum resource utilization, and average latency problems for every timestep.

To make a proper comparison between algorithms, speed is the desired attribute for measuring the performance of the cloud infrastructure. The computation time for various algorithms is listed in Table 8. The DQN/DDQN algorithm has taken minimum time with a higher percentage of faster computation in completing the 1000 episodes of training with TRPO+GAE as the baseline for the speed comparison.

2) EXPERIMENT 2: CHANGING THE MODEL PARAMETERS

For this experiment, the best weights amongst DQN, DDQN, PPO, and PPO A2C agents are saved with changes in the traffic pattern and internal dynamics of the model being used in the environments based on the training data. To evaluate the performance of agents in critical (or) new situations, i.e., measurement of generalization properties in the change of

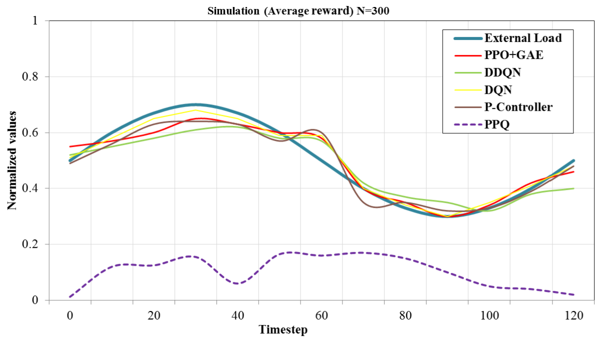


FIGURE 8. Traffic load and fluctuations in process load.

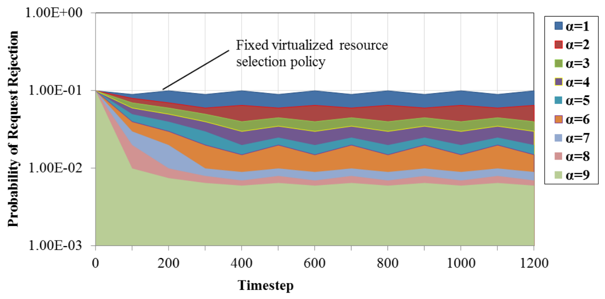


FIGURE 9. Automatic adaptation of virtualized resource selection policy.

parameters in the model. This change has become important in the process of self-healing of VNFs since there are different traffic patterns and configurations on their respective VNFs. A simple P-controller is added to the trained agents, and it's been deployed in the environment. The P-controller does not tune the external load (gain = 1) in the feedback of action selection.

The baseline for deploying the most influential fundamental control algorithm compared to the performance of other agents. The agents act optimally due to the policy being learned during training, where the agents are evaluated for each state and action without feedback. In Fig. 7, the plot compared statistics with respect to the policy deployed by each agent. The blue line indicates the external load, and the rest of the solid lines are the average resource allocation by each agent. The plot shows resource utilization and handles the average latency problems represented in dashed lines and maximum VM utilization (dotted) for each timestep.

The DQN and DDQN show consistency in getting the highest reward during training. In Fig. 8, the customer network is applied for mimicking the pattern. This pattern acts as a training pattern and stretches much longer over three days in the model, representing 5000 timesteps. In Fig. 9, the basis of the fixed virtualized resource selection policy deployed by our agents we conclude the fact that DQN receives the highest reward. In Fig. 10, the graph shows how the process load is updated in two different ways depending on whether the system performs a scaling action.

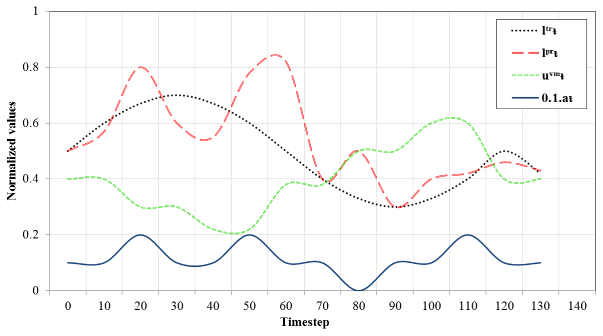


FIGURE 10. Visualisation of process load during random scaling actions.

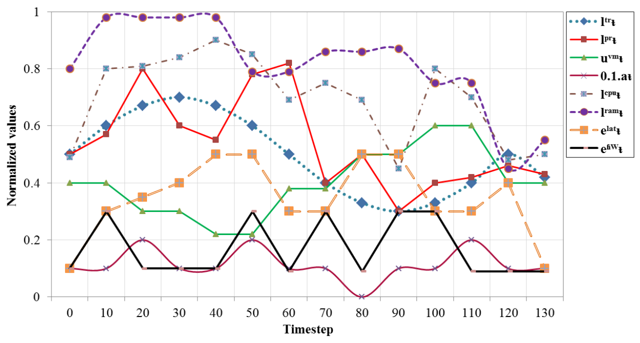


FIGURE 11. Visualization of the errors together with the other states.

In Fig. 11, the visualization of errors is shown together with the other states during random scaling operations. The graph clearly shows that errors are present when a low number of VMs are allocated and when the external load is high, a state where we can expect QoS issues in a real VNF. The episodic reward-based performance measurement is based on the computation time in seconds and speed in calculating percentages compared to DQN/DDQN with various other algorithms. The DQN/DDQN uses the two neural networks for calculating the TD target, and DDQN predicts the output computation of 27 % greater with 128 training samples, from 0.5ms to 1ms for each DQN/DDQN compared to PPO+MC, PPO+GAE, TRPO+MC, TRPO+GAE.

VI. CONCLUSION AND FUTURE WORK

The six state-of-the-art algorithms are trained for 1000 episodes and evaluated based on performance, rewards, cost, and speed. The PPO+MC agents marginally improve the cost by 1.7 % but it still raise the overhead for resource efficiency during autoscaling operations. TRPO+GAE agents are fairly good in auto-scaling, with an improvement of cost by 3.2 %. In comparison, our proposed DQN/DDQN learning approach best optimizes the price and lowers the cost by 18.4%. The adaptive self-healing of VNFs enhances the computation performance by about 27%, which is faster than the baseline of TRPO+GAE and other comparative state-of-the-art algorithms. These RL algorithms are developed in Python, using the TensorForce framework, and their performance is compared based on cost and stability. The algorithms were implemented to work on VMs with Apcera installed and were trained with data collected through Apceras API, and the simulation was carried out through a cloud cluster. We could see that DQN/DDQN outperforms PPO+MC, PPO+GAE, TRPO+MC, and TRPO + GAE compared to the applied agents. We note that TRPO and PPO with GAE estimation show better results than Monte Carlo estimation concerning stability and convergence rate. The comparison of DQN with other agents is strongly based on the relative performance in completing the task. The self-healing of VNF is solved using DRL, where the cost of development and maintenance has resulted in a performance gain.

The limitation that comes with the deployment of DRL to VNF is due to the fact that traffic patterns differ between customers, which results in uncertainty due to varied configurations and load patterns. The learning performance is bad, and there is a high risk of divergence as the VNF agents need to work in a multi-agent context. Even though we had achieved the optimal results of 27 % compared to all the state-of-the-art algorithms, the result proves that the simpler method based on control theory is equally good. Furthermore, using various configurations, VNF chaining process and load patterns results in a tedious validation for DRL with a heavy bottleneck of training the policies. We could embed the control methods with the classical machine learning properties in the future.

REFERENCES

[1] H. Fawaz, J. Lesca, P. T. A. Quang, J. Leguay, D. Zeghlache, and P. Medagliani, "Graph convolutional reinforcement learning for collaborative queuing agents," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 1, pp. 1363–1377, 2023, doi: 10.1109/TNSM.2022.3226605.

[2] D. Ma, B. Zhou, X. Song, and H. Dai, "A deep reinforcement learning approach to traffic signal control with temporal traffic pattern mining," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11789–11800, Aug. 2022, doi: 10.1109/TITS.2021.3107258.

[3] J. Yang, Y. Chen, K. Xue, J. Han, J. Li, D. S. L. Wei, Q. Sun, and J. Lu, "IEACC: An intelligent edge-aided congestion control scheme for named data networking with deep reinforcement learning," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 4, pp. 4932–4947, Dec. 2022, doi: 10.1109/TNSM.2022.3196344.

[4] R. Souza, K. Dias, and S. Fernandes, "NFV data centers: A systematic review," *IEEE Access*, vol. 8, pp. 51713–51735, 2020, doi: 10.1109/ACCESS.2020.2973568.

[5] G. Yunjie, H. Yuxiang, D. Yuehang, and X. Jichao, "Joint optimization of resource allocation and service performance in vEPC using reinforcement learning," in *Proc. IEEE 4th Int. Conf. Cloud Comput. Big Data Anal. (ICCCBDA)*, Apr. 2019, pp. 306–310, doi: 10.1109/ICCBDA.2019.8725620.

[6] M. Nakanoya, Y. Sato, and H. Shimonishi, "Environment-adaptive sizing and placement of NFV service chains with accelerated reinforcement learning," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, Apr. 2019, pp. 36–44.

[7] H. T. Nguyen, T. V. Do, A. Hegyi, and C. Rotter, "An approach to apply reinforcement learning for a VNF scaling problem," in *Proc. 22nd Conf. Innov. Clouds, Internet Netw. Workshops (ICIN)*, Feb. 2019, pp. 94–99, doi: 10.1109/ICIN.2019.8685866.

[8] Q. Jin, S. Ge, J. Zeng, X. Zhou, and T. Qiu, "ScaRL: Service function chain allocation based on reinforcement learning in mobile edge computing," in *Proc. 7th Int. Conf. Adv. Cloud Big Data (CBD)*, Sep. 2019, pp. 327–332.

[9] P. T. A. Quang, A. Bradai, K. D. Singh, and Y. Hadjadj-Aoul, "Multi-domain non-cooperative VNF-FG embedding: A deep reinforcement learning approach," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 886–891, doi: 10.1109/INFCOMW.2019.8845184.

[10] S. Lange, H. G. Kim, S. Y. Jeong, H. Choi, J. H. Yoo, and J. W. K. Hong, "Machine learning-based prediction of VNF deployment decisions in dynamic networks," in *Proc. 20th Asia–Pacific Netw. Operations Manage. Symp. (APNOMS)*, Sep. 2019, pp. 1–6, doi: 10.23919/APNOMS.2019.8893073.

[11] H. G. Kim, S. Y. Jeong, D. Y. Lee, H. Choi, J. H. Yoo, and J. W. K. Hong, "A deep learning approach to VNF resource prediction using correlation between VNFs," in *Proc. IEEE Conf. Netw. Softw. (NetSoft)*, Jun. 2019, pp. 444–449, doi: 10.1109/NETSOFT.2019.8806620.

[12] D. K. Jain, A. Mahanti, P. Shamsolmoali, and R. Manikandan, "Deep neural learning techniques with long short-term memory for gesture recognition," *Neural Comput. Appl.*, vol. 32, no. 20, pp. 16073–16089, Oct. 2020.

[13] Ardiansyah, Y. Choi, M. R. K. Aziz, K. Cho, and D. Choi, "Latency-optimal network intelligence services in SDN/NFV-based energy Internet cyberinfrastructure," *IEEE Access*, vol. 8, pp. 4485–4499, 2020.

[14] P. Chemouil, P. Hui, W. Kellerer, Y. Li, R. Stadler, D. Tao, Y. Wen, and Y. Zhang, "Special issue on artificial intelligence and machine learning for networking and communications," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1185–1191, Jun. 2019, doi: 10.1109/JSAC.2019.2909076.

[15] A. K. Bashir, R. Arul, S. Basheer, G. Raja, R. Jayaraman, and N. M. F. Qureshi, "An optimal multitier resource allocation of cloud RAN in 5G using machine learning," *Trans. Emerg. Telecommun. Technol.*, vol. 30, no. 8, p. e3627, Aug. 2019.

[16] Y. T. Woldeyohannes, A. Mohammadkhan, K. K. Ramakrishnan, and Y. Jiang, "ClusPR: Balancing multiple objectives at scale for NFV resource allocation," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 4, pp. 1307–1321, Dec. 2018, doi: 10.1109/TNSM.2018.2870733.

[17] F. B. Mismar and B. L. Evans, "Deep Q-learning for self-organizing networks fault management and radio performance improvement," in *Proc. 52nd Asilomar Conf. Signals, Syst., Comput.*, Oct. 2018, pp. 1457–1461, doi: 10.1109/ACSSC.2018.8645083.

[18] M. Latah and L. Toker, "Artificial intelligence enabled software-defined networking: A comprehensive overview," *IET Netw.*, vol. 8, no. 2, pp. 79–99, Mar. 2019, doi: 10.1049/iet-net.2018.5082.

[19] R. Li, Z. Zhao, Q. Sun, I. Chih-Lin, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74429–74441, 2018, doi: 10.1109/ACCESS.2018.2881964.

[20] H. R. Khezri, P. A. Moghadam, M. K. Farshbafan, V. Shah-Mansouri, H. Kebriaei, and D. Niyato, "Deep Q-learning for dynamic reliability aware NFV-based service provisioning," 2018, *arXiv:1812.00737*.

[21] B. Kar, E. H. Wu, and Y.-D. Lin, "Energy cost optimization in dynamic placement of virtualized network function chains," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 1, pp. 372–386, Mar. 2018, doi: 10.1109/TNSM.2017.2782370.

[22] L.-V. Le, D. Sinh, B. P. Lin, and L.-P. Tung, "Applying big data, machine learning, and SDN/NFV to 5G traffic clustering, forecasting, and management," in *Proc. 4th IEEE Conf. Netw. Softwarization Workshops (NetSoft)*, Jun. 2018, pp. 168–176, doi: 10.1109/NETSOFT.2018.8460129.

[23] F. Schmidt, F. Suri-Payer, A. Gulenko, M. Wallschläger, A. Acker, and O. Kao, "Unsupervised anomaly event detection for VNF service monitoring using multivariate online arima," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2018, pp. 278–283.

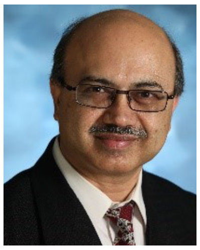
[24] Z. Zhou and T. Zhang, "Applying machine learning to service assurance in network function virtualization environment," in *Proc. 1st Int. Conf. Artif. Intell. Industries (AI4I)*, Sep. 2018, pp. 112–115, doi: 10.1109/AI4I.2018.8665716.

[25] S. Jacob, V. G. Menon, S. Joseph, P. G. Vinoj, A. Jolfaei, J. Lukose, and G. Raja, "A novel spectrum sharing scheme using dynamic long short-term memory with CP-OFDMA in 5G networks," *IEEE Trans. Cognit. Commun. Netw.*, vol. 6, no. 3, pp. 926–934, Sep. 2020, doi: 10.1109/TCCN.2020.2970697.

[26] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar, "A deep reinforcement learning perspective on Internet congestion control," in *Proc. Int. Conf. Mach. Learn.*, May 2019, pp. 3050–3059, doi: 10.1109/TVT.2020.2984038.

ARUNKUMAR ARULAPPAN (Member, IEEE) received the B.Tech. degree in information technology from Anna University, Chennai, India, the M.Tech. degree in computer science and engineering from the Vellore Institute of Technology (VIT), Vellore, India, and the Ph.D. degree from the Faculty of Information and Communication Engineering, Anna University, in 2023. He is an Assistant Professor with the School of Computer Science Engineering and Information Systems (SCORE), VIT University. He is proficient with simulator tools MATLAB, ns-3, Mininet, OpenNet VM, and P4 programming. He is exposed to open source tools, such as OpenStack, Cloudify, OPNFV, and Cloud-Native Computing Foundation (CNCF). His research interests include the cloud-native deployment, SDN, NFV, 5G/6G networks, AI/ML based networking, the Internet of Vehicles, and UAV communications.

ANIKET MAHANTI (Senior Member, IEEE) received the B.Sc. degree (Hons.) in computer science from the University of New Brunswick, Canada, and the M.Sc. and Ph.D. degrees in computer science from The University of Calgary, Canada. He is a Senior Lecturer (an Associate Professor) of computer science with The University of Auckland, New Zealand. His research interests include network science, distributed systems, and internet measurements.



KALPDRUM PASSI (Senior Member, IEEE) received the Ph.D. degree in parallel numerical algorithms from the Indian Institute of Technology, Delhi, India, in 1993. He is a Full Professor with the School of Engineering and Computer Science, Laurentian University, Sudbury, ON, Canada. He has collaborative work with faculty in Canada and U.S. and the work was tested on the CRAY XMP's and CRAY YMP's. He transitioned his research to web technology, and more recently has been involved in machine learning and data mining applications in bioinformatics, social media, and other data science areas. His research in bioinformatics has been on improving the accuracy of predicting diseases, such as different types of cancer using microarray data. He has published many papers on parallel numerical algorithms in international journals and conferences. He has published several papers related to machine learning applications in medical image processing, natural language processing, sports analytics, and social media platforms. He received funding from NSERC and Laurentian University for his research. He is a member of ACM and IEEE Computer Society.

THIRUVENKADAM SRINIVASAN (Senior Member, IEEE) was born in India. He received the B.E. degree in electrical and electronics engineering from Bharathiar University, in 1999, the M.E. degree in power systems from Annamalai University, in 2004, and the Ph.D. degree in power distribution system reconfiguration from Anna University, India, in 2011. Currently, he is a Professor with the School of Electrical Engineering, Vellore Institute of Technology, Vellore, India. He was a Visiting Professor with Kunsan National University, Gunsan, South Korea, from September 2019 to October 2022. His research interests include smart grids, microgrids, wireless networks, and machine learning. He is a Life Member of ISTE.

RANESH NAHA (Member, IEEE) received the M.Sc. degree in parallel and distributed computing from Universiti Putra Malaysia, and the Ph.D. degree in information technology from the University of Tasmania, Australia. He is a Lecturer (TFA) of Information Technology with Victoria University, Melbourne, Australia. He is also a Research Fellow at Centre for Smart Analytics (CSA), Federation University Australia. Prior to these roles, he worked as a grant-funded Researcher at the School of Computer Science, The University of Adelaide. He has authored more than 50 peer-reviewed scientific research articles. His research interests include distributed computing (fog/edge/cloud), the Internet of Things (IoT), AI & ML, software-defined networking (SDN), cybersecurity, and blockchain.

GUNASEKARAN RAJA (Senior Member, IEEE) received the Ph.D. degree from the Faculty of Information and Communication Engineering, Anna University, Chennai, India, in 2010. He is a Professor with the Department of Computer Technology, Anna University, where he is the Principal Investigator of NGNLab. He was a Postdoctoral Fellow with the University of California, Davis, USA. His research interests include the Internet of Vehicles, UAV communications, 5G/6G networks, blockchain, cyber security, computer vision, and the Medical IoT. He was a recipient of the Young Engineer Award from the Institution of Engineers, India, in 2009; the FastTrack Grant for Young Scientist from the Department of Science and Technology, in 2011; the Professional Achievement Award from IEEE Madras Section, in 2017; the Visvesvaraya Young Faculty Research Fellowship from the Ministry of Electronics and Information Technology, Government of India, in 2019 and 2020; and the IEEE Publication Award from IEEE Madras Section, in 2021.