

Delay-Aware NFV Resource Allocation with Deep Reinforcement Learning

¹Ningcheng Yuan, ¹Wenchen He, ²Jing Shen, ¹Xuesong Qiu, ¹Shaoyong Guo, ¹Wenjing Li

¹Beijing University of Posts and Telecommunications, Beijing, China

²Communication Operation Center State Grid Henan Electric Power Company

²Information & Telecommunication Company, Zheng Zhou, China

Email: yuan_nc@qq.com, wche@bupt.edu.cn, shenjing1@ha.sgcc.com.cn, {xsqiu, syguo, wjli}@bupt.edu.cn

Abstract—Network Function Virtualization (NFV) can support flexible services provisioning in form of Service Function Chains (SFCs) consisting of ordered Virtual Network Functions (VNFs). The end-to-end (E2E) delay of flow traversing SFC has been an important indicator, especially for delay-sensitive E2E services, but there does not exist an analytical model that can accurately evaluate it. Moreover, the complicated network and stochastic request arrival are hard to predict and model. Therefore, quantitative delay model and dynamic NFV resource allocation method are needed. In this paper, an adaptive allocation method is designed to meet E2E delay requirements. Firstly, we devise an NFV resource allocation framework based on deep reinforcement learning (DRL) that can adapt to network changes by interacting with the network. Then a dynamic queuing model is established to determine average E2E packet delay. Based on the delay, we define the network utility function and propose a minimizing delay (MD) problem. According to the continuity of the problem, we use unsupervised reinforcement and auxiliary learning (UNREAL) to obtain the optimal allocation policy. At last, extensive simulation results show that UNREAL-MD has better performance compared to state-of-the-art methods in terms of delay, throughput and network utility.

Index Terms—Network Function Virtualization, end-to-end delay, resource allocation, deep reinforcement learning

I. INTRODUCTION

With the evolution brought about by resource virtualization, network function virtualization (NFV) has played a central role in the advancement of network infrastructure [1]. NFV supports decoupling the network functions (NFs) from proprietary hardware appliances and providing flexible provisioning of virtual network functions (VNFs) on top of optimally shared physical infrastructure. A set of VNFs are linked together by virtual links to form a service function chain (SFC) [2]. By scaling VNFs in SFCs, NFV supports a more flexible resource allocation approach among services.

Due to the rapid growth of delay-sensitive services (e.g., audio, video and game services), network needs to promise a well delay performance to guarantee reliable network service. Resource allocation strategy has a direct impact on network

delay. For embedded VNF chains, CPU and bandwidth resources sharing method among multiple flows determines processing and transmission rates [3], and thus affects average E2E packet delay. Thus minimizing delay (MD) is a problem worth considering in NFV resource allocation. However, most existing works basically focus on packet transmission delays on physical links, while do not consider the packet processing time on NFV nodes [4] [5]. So, building a model to evaluate the transmission and processing delay of packets in NFV is a challenging research issue.

The impact on delay under different resource sharing methods is illustrated and shown in Fig. 1, VNF 1 and VNF 2 are embedded on the same server. Flow 1 and flow 2 are processed by them respectively. The total amount of available resources in server is C units, assuming that if all of them are allocated to NF 1, the processing of each packet of flow 1 requires τ units of time; if all are allocated to NF 2, each packet in flow 2 needs 2τ units processing time to pass NF 2. We assume that the relationship between CPU capacity and processing rate is linear [3]. Consider two distribution ways: If $C_1 = 2C/3$, $C_2 = C/3$ like Fig. 1(a), the packet delays in flow 1 and flow 2 are 1.5τ and 6τ respectively; In Fig. 1(b), $C_1 = C_2 = 1/2$, the average packet processing time of flow 1 becomes 2τ , and the average processing time of flow 2 is 4τ , in this case, the overall network delay is reduced and it is fairer to the two flows.

Typically, NFV resource allocation problem is formulated as an optimization problem to achieve optimal allocation strategy with minimum costs (e.g., resource requirements, power consumption, etc.) and maximum total revenue (e.g., throughput, network utility, etc.) [6] [7] [8]. Some mathematical methods such as linear programming (LP) and integer linear programming (ILP) are generally used to solve this kind of problem [1] [8].

However, these existing works mostly assume that the arriving requests are predictable or known a priori, without considering real-time network changes. It has been proved that Deep Reinforcement Learning (DRL) framework has great advantages in solving such dynamic problems because it is able to determine the best strategy based on its experience

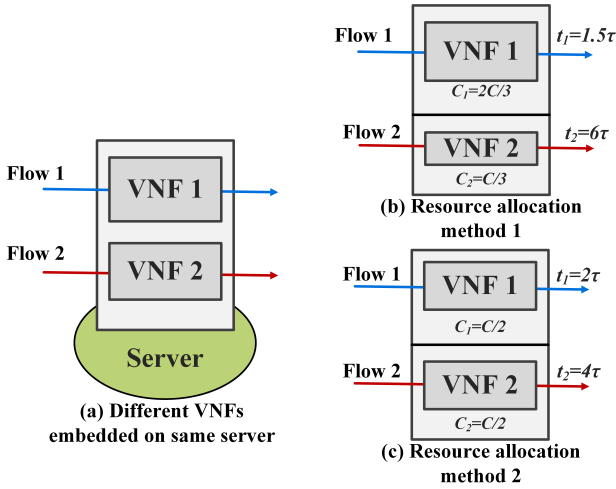


Fig. 1. The impact of resource allocation on delay

interacting with the environment [9]. Under the DRL-based resource allocation framework, the optimal resource allocation strategy with best delay performance can be obtained by using the function of E2E delay as environment feedback. However, since the resource allocation space belongs to continuous domain, basic DRL technique does not work. Although a state-of-the-art method, asynchronous advantage actor-critic (A3C) has been proposed for continuous control, it still has defects in learning speed and robustness [10].

In this paper, we establish a queueing model to extract impact of VNF scaling on E2E delay and improve A3C to unsupervised reinforcement and auxiliary learning (UNREAL) for resource allocation [11] [12]. The main contributions of this paper are as follows:

- We provide resource allocation framework based A3C that can dynamically sense and model network and services.
- We develop an analytical queueing model to evaluate E2E delay that packets experience when traversing embedded NFV chain and define the delay-related utility function as environment feedback for the allocation strategy.
- We propose an improved A3C algorithm, UNREAL-MD to achieve optimal resource allocation strategy with minimum delay.

The rest of this paper is organized as follows. The A3C-based NFV resource allocation model is introduced in Section II. The minimizing delay problem formulation is presented in Section III. In Section IV, we propose UNREAL-MD, with detailed introductions on its principle and architecture. Section V presents the evaluation results and Section VI concludes our work.

II. A3C-BASED RESOURCE ALLOCATION MODEL

There are two major components in the A3C-based architecture for NFV resource allocation: the observed environment and intelligent agent. The agent interacts with the environment to obtain samples of the environment state that are used

to update global network parameters. These parameters will in turn affect the agent's exploration strategy. The A3C-based algorithm achieves the optimal strategy of resource allocation through such repeated learning processes. These two components are described in detail as follows.

Environment: The environment of NFV resource allocation problems consists of three parts: service provider, NFV network and customers. The NFV network provides network infrastructure that supports the virtualization controller to place the VNFs on it, and adjust resources allocated to them. The service provider provides network services for customers, and customers accept these services and feedback service quality information such as network delays.

Agent: The agent learns the best strategy by interacting with the environment. In the background of NFV resource allocation problem, the virtualization controller can be considered as agent, responsible for collecting network information, making decisions and taking actions.

III. OPTIMIZATION PROBLEM MODEL

A. Network and Flow Model

An NFV network can be represented as an undirected graph $G = (N, E)$, where N is the set of nodes. N is composed of source node set S , processing node set P and destination set D , i.e., $N = S \cup P \cup D$. The z th node $n_z \in N$ has the resource capacity of C_{n_z} for packet processing. There can be multiple VNF instances placed on one NFV node and sharing its resources. The embedded VNF set in node n_z is $V(n_z)$. E is the set of edges in NFV network. $e_{ij} \in E$ represents the edge that connects node n_i and n_j , with resource capacity of $C_{e_{ij}}$ for packet transmission.

There is a set of network flows F , where $f \in F$ is the set of packets belonging to the same service with the same source and destination. Each f has its delay requirement, and its normalized maximum tolerance delay is α_f with value between 0 and 1. The stricter delay requirement of flow f , the smaller the α_f is. $F(n_z)$ is the set of flows that pass n_z . $v(f, k)$ denotes the k th VNF that f passes, with $k \in [0, |f|]$. Where $|f|$ is the length of the VNF chain that f traverses, i.e., the number of VNFs in the chain.

$Q(f, k)$ represents the NFV node where the k th VNF of f is embedded. $Q(f, 0) \in S$, $Q(f, |f|) \in D$. $P(f, k)$ represents the physical path between the nodes where the k th and $k + 1$ th VNFs of f are located. The path includes $|P(f, k)| + 1$ transmission links, denoted as $\{P_0^L(f, k), \dots, P_n^L(f, k), \dots, P_{|P(f, k)|}^L(f, k)\}$ and $|P(f, k)|$ switches: $\{P_1^S(f, k), \dots, P_n^S(f, k), \dots, P_{|P(f, k)|}^S(f, k)\}$.

Packet arrivals of f at physical nodes and links are modeled as time-varying Poisson process. Considering a set of discrete time slots $\{0, 1, 2, \dots, t, \dots, T - 1\}$, at time slot t , the average arrival rate on the k th VNF (which embedded on $Q(f, k)$) of flow f is $\lambda_{Q(f, k)}(t)$. $A_{Q(f, k)}(t)$ is the number of packets of

f arriving on $Q(f, k)$ at time t , and according to the property of the Poisson process, $A_{Q(f, k)}(t)$ satisfies

$$P[N(T+t) - N(T) = A_{Q(f, k)}(t)] = \frac{e^{-\lambda_{Q(f, k)}(t)t} (\lambda_{Q(f, k)}(t)t)^{A_{Q(f, k)}(t)}}{A_{Q(f, k)}(t)!} \quad (1)$$

$B_{Q(f, k)}(t)$ represents the queue backlogs of flow f on $Q(f, k)$. Similarly, the average arrival rate of f on $P(f, k)$ is $\lambda_{P(f, k)}(t)$, the number of arriving packets and queue backlogs are $A_{P(f, k)}(t)$ and $B_{P(f, k)}(t)$.

B. Queuing Model

When a packet arrives the NFV node, the node provides CPU resources to process it, and then the link bandwidth resource is allocated for transmission. On one NFV node, different VNFs can be flexibly installed and coordinated, so that CPU and bandwidth resources are shared among different flows. The resources allocated to the flow affect the processing/transmission rate of the packets.

Since different flows have different packet structures and sizes, and different VNF types have different resource requirements, we define $R_{Q(f, k)}$ as the number of packets that the unit resource of the k th VNF of f can handle in a unit of time. Bandwidth utilization can be maximized only when the transmission rates of all switches and links on $P(f, k)$ are equal. Therefore, it is only necessary to consider the resource allocation of the first link on $P(f, k)$, i.e., $P_0^L(f, k)$. The transmission rates on other switches and links are consistent with the one on $P_0^L(f, k)$. $R_{P_0^L(f, k)}$ is the link transmission rate, indicating that $P_0^L(f, k)$ with unit resource can transfer $R_{P_0^L(f, k)}$ packets in a unit of time.

To describe the resource sharing strategy among flows, we define the resource allocation ratio $h_{Q(f, k)}(t)$ and $h_{P_0^L(f, k)}(t)$. At time slot t , ratio of CPU resources allocated to f at node $Q(f, k)$ is $h_{Q(f, k)}(t)$; Ratio of link bandwidth allocated to f at $P_0^L(f, k)$ is $h_{P_0^L(f, k)}(t)$. Since the total available resources on $Q(f, k)$ and $P_0^L(f, k)$ are $C_{Q(f, k)}$ and $C_{P_0^L(f, k)}$ respectively, according to the resource allocation ratio, we can calculate that the processing resource allocated by node $Q(f, k)$ to f is $c_{Q(f, k)}(t) = C_{Q(f, k)} h_{Q(f, k)}(t)$ and the transmission resource that $P_0^L(f, k)$ allocates to f is $c_{P_0^L(f, k)}(t) = C_{P_0^L(f, k)} h_{P_0^L(f, k)}(t)$. The processing and transmission process of packets is modeled as fixed regular interval distribution. The processing rate on node $Q(f, k)$ is $\mu_{Q(f, k)}(t) = R_{Q(f, k)} c_{Q(f, k)}(t)$ and the transmission rates on $|P(f, k)| + 1$ links and $|P(f, k)|$ switches of path $P(f, k)$ are the same and all calculated as $\mu_{P(f, k)}(t) = R_{P_0^L(f, k)} c_{P_0^L(f, k)}(t)$.

From 0 to t , the average arrival rate of f on $Q(f, k)$ is $\bar{\lambda}_{Q(f, k)}(t) = \frac{1}{t} \sum_{\tau=0}^t E\{\lambda_{Q(f, k)}(\tau)\}$ and the average processing rate is $\bar{\mu}_{Q(f, k)}(t) = \frac{1}{t} \sum_{\tau=0}^t E\{\mu_{Q(f, k)}(\tau)\}$. Similarly, on link $P(f, k)$, the average packet arrival rate and cumulative average transmission rate are $\bar{\lambda}_{P(f, k)}(t) = \frac{1}{t} \sum_{\tau=0}^t E\{\lambda_{P(f, k)}(\tau)\}$ and $\bar{\mu}_{P(f, k)}(t) = \frac{1}{t} \sum_{\tau=0}^t E\{\mu_{P(f, k)}(\tau)\}$ respectively. Based on

these parameters, we can use the M/D/1 model to calculate the average cumulative packet delay $d_{Q(f, k)}(t)$ and $d_{P(f, k)}(t)$ on the nodes and links from 0 to t .

According to the queuing theory, the average packet delay on node $Q(f, k)$ is $d_{Q(f, k)}(t) = \frac{1}{\bar{\mu}_{Q(f, k)}(t)} + W_{Q(f, k)}$, the sum of the processing time and the queue waiting time $W_{Q(f, k)}$. It can be proved that $W_{Q(f, k)}$ is related to transmission process on the link $P(f, k-1)$ which is before the node $Q(f, k)$. When the processing rate on node $Q(f, k)$ is greater than the transmission rate on $P(f, k-1)$, $\bar{\mu}_{Q(f, k)}(t) \geq \bar{\mu}_{P(f, k-1)}(t)$, there is no queueing delay for processing at $Q(f, k)$, i.e., $W_{Q(f, k)} = 0$. When $\bar{\mu}_{Q(f, k)}(t) < \bar{\mu}_{P(f, k-1)}(t)$, it has been proved that we can approximate the packet arrival process at $Q(f, k)$ as Poisson process with rate parameter $\bar{\lambda}_{Q(f, k)}(t)$, and $\bar{\lambda}_{Q(f, k)}(t) = \bar{\lambda}_{Q(f, 0)}(t)$. Under the M/D/1 queuing model, the average packet waiting time before processing for flow f at $Q(f, k)$ is determined by $W_{Q(f, k)} = \frac{\bar{\lambda}_{Q(f, k)}(t)}{2\bar{\mu}_{Q(f, k)}(t)(\bar{\mu}_{Q(f, k)}(t) - \bar{\lambda}_{Q(f, k)}(t))}$ [4].

Similarly, the average total transmission time on $P(f, k)$ is $\frac{2|P(f, k)|+1}{\bar{\mu}_{P(f, k)}(t)}$. Except $P_0^L(f, k)$, there is no waiting time on the links and switches. The waiting time on $P_0^L(f, k)$ can be calculated as

$$W_{P(f, k)} = \begin{cases} \frac{\bar{\lambda}_{P(f, k)}(t)}{2\bar{\mu}_{P(f, k)}(t)(\bar{\mu}_{P(f, k)}(t) - \bar{\lambda}_{P(f, k)}(t))}, & \bar{\mu}_{P(f, k)}(t) < \bar{\mu}_{Q(f, k)}(t) \\ 0, & \bar{\mu}_{P(f, k)}(t) \geq \bar{\mu}_{Q(f, k)}(t) \end{cases} \quad (2)$$

where $\bar{\lambda}_{P(f, k)}(t) = \bar{\lambda}_{Q(f, 0)}(t)$. Therefore, the average packet delay from node $Q(f, k)$ to $Q(f, k+1)$ is $d_{P(f, k)}(t) = \frac{2|P(f, k)|+1}{\bar{\mu}_{P(f, k)}(t)} + W_{P(f, k)}$.

So, for packets in flow f , the average delay they experience between reaching node n_z and arriving the next NFV node is $d_f^z(t) = d_{Q(f, k)}(t) + d_{P(f, k)}(t)$, where k satisfies $Q(f, k) = n_z$.

The average packet delay for the entire flow from 0 to t is the sum of the average delays on all NFV nodes and paths that it passes from the source node to the destination node: $d_f(t) = \sum_k d_{Q(f, k)}(t) + \sum_k d_{P(f, k)}(t)$. Finally, the long-term average delay is $d_f = \lim_{T \rightarrow \infty} d_f(T) = \sum_k [\lim_{T \rightarrow \infty} d_{Q(f, k)}(T)] + \sum_k [\lim_{T \rightarrow \infty} d_{P(f, k)}(T)] = \sum_k d_{Q(f, k)} + \sum_k d_{P(f, k)}$.

C. Problem Formulations

Now we present the formulation of the resource allocation problem. Our goal is to minimize the overall network delay while ensuring fairness among flows. In order to balance the delay among flows, the utility function is introduced: $U_\alpha(x) = x^{1-\alpha}/(1-\alpha)$. α is a tuning parameter which can be used to tradeoff fairness and maximization. For $\alpha > 0$, $U_\alpha(x)$ is monotonically increasing with x . When $\alpha = 0$, $U_\alpha(x) = x$, the problem is degenerated to a maximization problem. As $\alpha \rightarrow 1$, in the limit $U_\alpha(x)$ becomes $\log x$, at this point, the problem objective becomes to achieve the proportional fairness. According to the above function, we define the α -fairness utility related to the tolerance delay α_f of the flow

$f: U_{\alpha_f}(d_f) = d_f^{1-\alpha_f} / (1 - \alpha_f)$. This makes the flow with a lower delay tolerance take up a larger proportion.

Consider the following constraints:

a) *Capacity Constraint*: The total resources allocated to all flows on a node or link cannot exceed their available resources. Therefore, the sum of resource allocation ratios on a node or link cannot exceed 1:

$$C_1: 0 \leq h_{Q(f,k)}(t) \leq 1, \quad (3)$$

$$C_2: 0 \leq h_{P(f,k)}(t) \leq 1, \quad (4)$$

$$C_3: \sum_{f \in F(n_z)} h_{Q(f,k)}(t) \leq 1, \quad (5)$$

$$C_4: \sum_{f \in F(n_z)} h_{P(f,k)}(t) \leq 1. \quad (6)$$

b) *Queueing Theory*: Without loss of generality, we assume that no flow arrives before time slot 0, and all queue buffers are infinite. However, in order to ensure the system stability, the total number of queued packets must be limited:

$$C_5: B_{Q(f,k)}(0) = 0, \quad (7)$$

$$C_6: B_{P(f,k)}(0) = 0, \quad (8)$$

$$C_7: \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} E\{B_{Q(f,k)}(\tau)\} < \infty, \quad (9)$$

$$C_8: \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} E\{B_{P(f,k)}(\tau)\} < \infty. \quad (10)$$

In summary, our MD problem can be formulated as follows:

$$\begin{aligned} \min & \left[\sum_{f \in F} U_{\alpha_f}(d_f) \right] \\ \text{s.t.} & : C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8. \end{aligned} \quad (11)$$

IV. ALGORITHM DESCRIPTION

A. A3C Framework

In this section, we present the proposed DRL framework. In a standard DRL setup, the problem is modeled as a Markov decision process (MDP) [9], including state space, action space, instantaneous reward, and state transition probabilities, expressed as $\{s, a, r, p\}$. The agent interacts with the environment, observes the state s , and executes the action a to make the environment change and get instantaneous reward r . From the current state s_i to the future, discounted future reward is $G_i = \sum_{k=0}^{\infty} \gamma^k r_{i+k+1}$. Where γ is a discount factor with value between 0 and 1. The goal of the agent is to find the policy $\pi(s)$, under which the expected return $J(\pi) = E\{G|\pi\} = E\{\sum_{i=0}^{\infty} \gamma^i r_i | s_0, a_0, \pi\}$ is maximized.

In the optimization problem model mentioned above, each NFV node n_z can be regarded as an agent to make resource allocation decisions. We define state space, action space and reward function as follows:

State space: We define state vector as: $s = \{s_Q, s_P\}$. where $s_Q = \{A_{Q(f,k)}(t), B_{Q(f,k)}(t), C_{Q(f,k)}, R_{Q(f,k)}(t)\}$,

$s_P = \{A_{P(f,k)}(t), B_{P(f,k)}(t), C_{P(f,k)}, R_{P(f,k)}(t)\}$, and $\forall f \in F(n_z), \forall k$ satisfying $Q(f, k) = n_z$.

Action space: We define action as allocating a certain ratio of resources to each f . $a = \{h_{Q(f,k)}(t), h_{P(f,k)}(t)\}$, where $\forall f \in F(n_z), \forall k$ satisfying $Q(f, k) = n_z$.

Reward function: Since the goal of DRL is to maximize cumulative reward, we use the opposite of delay utility: $r = - \sum_{f \in F(n_z)} U_{\alpha_f}(d_f^z(t))$.

The parameterized actor function is expressed as $\pi(s_i|\theta^\pi)$ and parameterized critic $V(s_i|\theta^V)$ measures the quality of the existing policy $\pi(s_i)$. The parameters of the actor and critic are learned by iteratively minimizing a sequence of loss functions, which are respectively defined as:

$$L_{actor} = E[\log \pi(s_i|\theta^\pi)(G_i - V(s_i|\theta^V))] , \quad (12)$$

$$L_{critic} = E[(G_i - V(s_i|\theta^V))^2] . \quad (13)$$

Finally, the loss function of A3C is defined as $L_{A3C} = L_{actor} + L_{critic} - E[\beta H(\pi(s_i|\theta^\pi))]$, where H is the entropy. The hyperparameter β controls the strength of the entropy regularization term, which is used to prevent the premature convergence of the policy.

B. UNREAL-MD

However, both the learning speed and the robustness of the A3C algorithm are not ideal enough. Some studies have added the concept of auxiliary task to the A3C algorithm, allowing the agent to maximize other pseudo-reward functions to improve the performance of the A3C algorithm [11] [12]. This method is called unsupervised reinforcement and auxiliary learning (UNREAL).

Given a set of auxiliary control tasks C , we define an auxiliary control task $c \in C$ by reward function $r^{(c)}$. For each task c , we optimize an n -step Q-learning loss $L_Q^{(c)} = E[(\sum_{k=0}^{n-1} \gamma^k r_{i+k}^{(c)} + \gamma^n \max_{a'} Q^{(c)}(s', a', \theta_{i-1}) - Q^{(c)}(s, a, \theta_i))^2]$.

Based on the above ideas, according to the characteristics of the proposed MD problem, we designed and used the following auxiliary task:

- **Throughput control**: There is a certain correlation between throughput and delay. If the total throughput of the network is larger, it means that more traffic is processed per unit time, and the packet delay may be relatively small. We set maximizing network throughput as an auxiliary task, train agents that learn a separate policy for maximizing throughputs at each node. The reward of this task is the α -fairness utility of throughput: $r^{(TC)} = \sum_{f \in F(n_z)} U_{\alpha_f}(A_{Q(f,k)}(t))$
- **Features control**: The more units a neural network uses, the more intelligent it may be, and better choices it can make. The features control task defines the number of active units in the hidden layer as reward to maximize the learning ability of agents.

The UNREAL algorithm optimizes a single combined loss function with respect to the joint parameters θ of the agent,

Algorithm 1 UNREAL-MD

Input: $G = (N, E)$, $V(n_z)$, F **Output:** Fraction of CPU/link bandwidth resources allocated to flow f : $H_{Q(f,k)}(t)$, $H_{P(f,k)}(t)$

```
1: Initialize:  $C_{Q(f,k)}$ ,  $C_{P(f,k)}$ ,  $\alpha$ ,  $\theta^\pi$ ,  $\theta^V$  and other parameters.
2: repeat
3:   Reset gradients:  $d\theta^\pi \leftarrow 0$ ,  $d\theta^V \leftarrow 0$ 
4:   Synchronize thread-specific parameters  $\theta^{\pi'} = \theta^\pi$  and  $\theta^{V'} = \theta^V$ 
5:    $t_{start} = t$ 
6:   Get state  $s_t$ 
7:   repeat
8:     Get  $a_t$  according to policy  $\pi(s_t|\theta^\pi)$ 
9:     Execute action  $a_t$ , calculate E2E delay and throughput to achieve the reward  $r_t$ ,  $r_t^{(c)}$  and receive new state  $s_{t+1}$ 
10:     $t \leftarrow t + 1$ 
11:     $T \leftarrow T + 1$ 
12:    until terminal  $s_t$  or  $t - t_{start} == t_{max}$ 
13:     $G = \begin{cases} 0, & \text{for terminal } s_t \\ V(s_t|\theta^{V'}), & \text{for non-terminal } s_t \end{cases}$ 
14:     $G^{(c)} = \begin{cases} 0, & \text{for terminal } s_t \\ V^{(c)}(s_t|\theta^{V'}), & \text{for non-terminal } s_t \end{cases}$ 
15:    for  $i \in \{t-1, \dots, t_{start}\}$  do
16:       $G \leftarrow r_i + \gamma G$ 
17:       $G^{(c)} \leftarrow r_i^{(c)} + \gamma G^{(c)}$ 
18:      Loss function of actor:  $L_{actor} = E[\log \pi(s_i|\theta^\pi)(G_i - V(s_i|\theta^V))]$ 
19:      Loss function of critic:  $L_{critic} = E[(G_i - V(s_i|\theta^V))^2]$ 
20:      Loss function of auxiliary tasks:  $L_Q^{(c)} = E[(G_{i:i+n} + \gamma^n \max_{a'} Q^{(c)}(s', a', \theta_{i-1}) - Q^{(c)}(s, a, \theta_i))^2]$ 
21:      Accumulate gradients w. r. t.  $\theta^{\pi'}$  and  $\theta^{V'}$ 
22:    end for
23:    Perform asynchronous update of  $\theta^\pi$  using  $d\theta^\pi$  and of  $\theta^V$  using  $d\theta^V$ 
24: until  $T > T_{max}$ 
```

which combines the A3C loss L_{A3C} together with auxiliary throughput control loss L_{TC} and features control loss L_{FC} :

$$L_{UNREAL} = L_{A3C} + \lambda_{TC} L_{TC} + \lambda_{FC} L_{FC} \quad (14)$$

In summary, our UNREAL-MD algorithm is expressed as algorithm 1.

V. PERFORMANCE EVALUATION

In the experiment, a random network topology with 20 nodes and 80 links is established. Each node has 50 to 100 units of CPU resources depending on its type, and the link bandwidth is between 100 to 300. Based on real world trace, there are several different commonly-deployed VNFs hosted on NFV nodes, such as firewall, NAT, IDS, load balancer, WAN optimizer and flow monitor, which are simulated for

composing SFCs, each containing 1 to 5 VNFs. We generate 30 flows with different SFC requirements. The packets arrival at source node of each flow follows Poisson process, with their mean arrival rates value distributed within a window with a size of 10.

We compare our algorithm with the following two basic algorithms and A3C:

- Random Allocation Algorithm: The proportion of resources allocated randomly to each VNF on each node.
- Dominant-resource generalized processor sharing (DRGPS): The dominant resource required by the flow is determined according to the structure of the data packet, and the proportion of the dominant resource is allocated according to the priority of the service.
- A3C: For fair comparison, all settings (such as state, action, reward, and CNN parameters) are consistent when replacing UNREAL-MD with the A3C algorithm.

We use average E2E packet delay, E2E throughput and the sum of the delay and throughput utilities as indicators to measure the performance of each algorithm under different traffic arrival rates and runtimes. The units of delay and throughput are ms and Mbps respectively. In addition, we compare the performance of two DRL algorithms (A3C and UNREAL-MD) in terms of online learning efficiency and cumulative reward.

As shown in Fig. 2, we change the central values of the traffic arrival rate windows and run the simulation for 20,000 time slots. We can see that compared to the two basic algorithms, UNREAL-MD significantly reduces network latency, while maintaining great performance in increasing network throughput and utility due to the consideration of throughput in training. At the same time, the overall trend is more stable under our method.

When comparing the performance of different algorithms over time, the corresponding traffic demand is generated using arrival rate window [20,30]. During the operation, UNREAL-MD leads to consistently lower delay and higher throughput than all the other methods as shown in Fig. 3. Our method adaptively provides allocation strategy based on the environment state, making network delay tends to be stable.

In Fig. 4, we show how L_{A3C} and reward values change to prove that UNREAL-MD performs better than A3C in online learning. Using UNREAL-MD, loss function converges faster, with an optimal solution that gives high reward quickly reached; While A3C uses more time and only obtains lower cumulative reward, corresponding to a local optimal solution.

VI. CONCLUSION

In this paper, we study the average E2E packet delay minimization problem in NFV system. To improve the E2E delay satisfaction of the services and ensure fairness among flows, we firstly present an analytical model to determine average E2E packet delay in embedded VNF chains, and define network utility function about E2E delay. Then we propose UNREAL-MD, an improvement of A3C algorithm

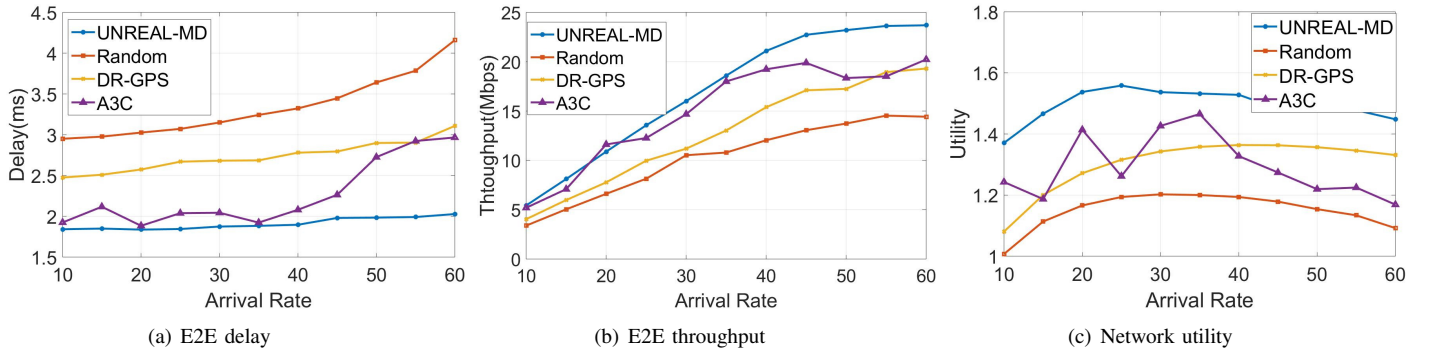


Fig. 2. Performance of all methods with different average packet arrival rate

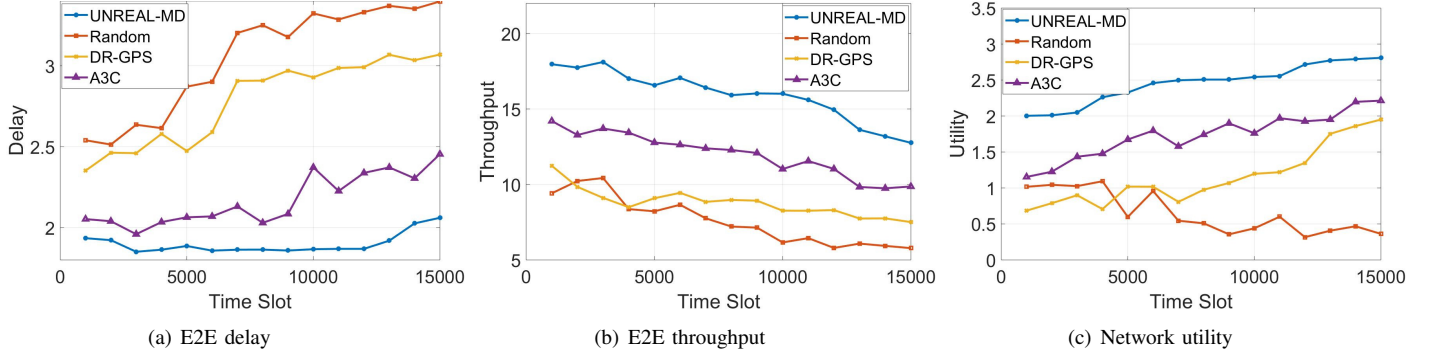


Fig. 3. Performance over time of all methods

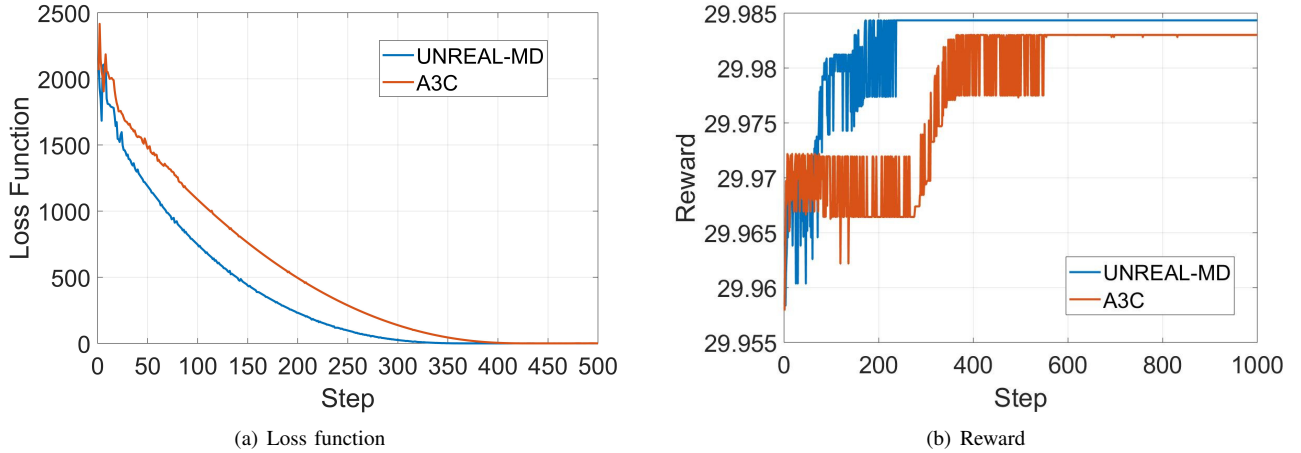


Fig. 4. Learning curve of the DRL-based methods

to reach the best VNF resource allocation strategy by optimizing cumulative reward. Simulation results demonstrate that UNREAL-MD reduces E2E delay and improves the network throughput and utility than other methods.

REFERENCES

- [1] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236-262.
- [2] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos and L. P. Gaspar, "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, 2015, pp. 98-106.
- [3] S. Dräxler, H. Karl and Z. Á. Mann, "JASPER: Joint Optimization of Scaling, Placement, and Routing of Virtual Network Services," in *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 946-960, Sept. 2018.
- [4] Q. Ye, W. Zhuang, X. Li and J. Rao, "End-to-End Delay Modeling for Embedded VNF Chains in 5G Core Networks," in *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 692-704, Feb. 2019.
- [5] L. Wang, Z. Lu, X. Wen, R. Knopp and R. Gupta, "Joint Optimization of Service Function Chaining and Resource Allocation in Network Function Virtualization," in *IEEE Access*, vol. 4, pp. 8084-8094, 2016.
- [6] Z. Xu, X. Zhang, S. Yu and J. Zhang, "Energy-Efficient Virtual Network Function Placement in Telecom Networks," 2018 IEEE International

- Conference on Communications (ICC), Kansas City, MO, 2018, pp. 1-7.
- [7] R. Gouareb, V. Friderikos and A. Aghvami, "Virtual Network Functions Routing and Placement for Edge Cloud Latency Minimization," in *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2346-2357, Oct. 2018.
 - [8] Q. Sun, P. Lu, W. Lu and Z. Zhu, "Forecast-Assisted NFV Service Chain Deployment Based on Affiliation-Aware vNF Placement," 2016 *IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, 2016, pp. 1-6.
 - [9] Y. Li, "Deep reinforcement learning: An overview".arXiv preprintar Xiv:1701.07274 (2017).
 - [10] V. Mnih, Badia, P. Adrià, M. Mirza, et al. "Asynchronous Methods for Deep Reinforcement Learning." arXiv preprintar arXiv:1602.01783v2 (2016).
 - [11] M. Jaderberg,V. Mnih, W. M. Czarnecki, T. Schaul,,J. Z. Leibo and D. Silver, et al. "Reinforcement Learning with Unsupervised Auxiliary Tasks". arXiv preprintar arXiv:1611.05397 (2016).
 - [12] Mirowski. P, Pascanu. R, Viola. F, et al. "Learning to Navigate in Complex Environments." arXiv preprint arXiv:1611.03673 (2016).