

Article

Reliable Service Function Chain Deployment Method Based on Deep Reinforcement Learning

Hua Qu ^{1,2}, Ke Wang ^{1,*} and Jihong Zhao ^{2,3}¹ School of Software Engineering, Xi'an Jiaotong University, Xi'an 710049, China; qh@mail.xjtu.edu.cn² School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China; zhaojihong@xjtu.edu.cn³ School of Communication and Information Engineering, Xi'an University of Posts & Telecommunications, Xi'an 710061, China

* Correspondence: wangke12345@stu.xjtu.edu.cn

Abstract: Network function virtualization (NFV) is a key technology to decouple hardware device and software function. Several virtual network functions (VNFs) combine into a function sequence in a certain order, that is defined as service function chain (SFC). A significant challenge is guaranteeing reliability. First, deployment server is selected to place VNF, then, backup server is determined to place the VNF as a backup which is running when deployment server is failed. Moreover, how to determine the accurate locations dynamically with machine learning is challenging. This paper focuses on resource requirements of SFC to measure its priority meanwhile calculates node priority by current resource capacity and node degree, then, a novel priority-awareness deep reinforcement learning (PA-DRL) algorithm is proposed to implement reliable SFC dynamically. PA-DRL determines the backup scheme of each VNF, then, the model jointly utilizes delay, load balancing of network as feedback factors to optimize the quality of service. In the experimental results, resource efficient utilization, survival rate, and load balancing of PA-DRL were improved by 36.7%, 35.1%, and 78.9% on average compared with benchmark algorithm respectively, average delay was reduced by 14.9%. Therefore, PA-DRL can effectively improve reliability and optimization targets compared with other benchmark methods.

Keywords: network function virtualization; service function chain; reliable; priority-awareness; deep reinforcement learning



Citation: Qu, H.; Wang, K.; Zhao, J. Reliable Service Function Chain Deployment Method Based on Deep Reinforcement Learning. *Sensors* **2021**, *21*, 2733. <https://doi.org/10.3390/s21082733>

Academic Editors: Seokhoon Yoon and Jangyoung Kim

Received: 20 February 2021

Accepted: 16 March 2021

Published: 13 April 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the traditional network, a plenty of network functions are relying on dedicated hardware deployment to form a variety of network services, this traditional deployment method can no longer meet the complex and changeable network requirements with the rapid growth of the number and diversity of network service requests. However, the emergence of network function virtualization (NFV) technology has solved that problem [1]. NFV makes network equipment functions no longer depend on dedicated equipment through software and hardware decoupling and function abstraction, in addition, NFV can reduce the high cost of network equipment by sharing network resources fully and flexibly. Moreover, NFV can realize the rapid development and deployment of new business, and realize automatic deployment, elastic scaling, fault isolation and self-healing of new business, etc. [2]. In NFV, several virtualized network functions (VNF) form a sequence of network functions that can handle specific network services by deploying on server nodes in a certain order, which is named as service function chain (SFC) [3]. As an important form of network and service, the SFC can construct a complete end-to-end network service through rational orchestration of VNFs [4].

Reliable deployment is an important concern for many SFCs, the traditional SFC deployment method relies on the routing strategy formulated by service providers, that

makes network traffic go through several VNFs in a certain order to provide the required network functions [5]. However, there are some defects need be solved. First, the method is lack of certain tolerance measure which is quite important for SFC deployment, so that it's difficult to meet the reliable deployment requirements of SFC. Secondly, different SFC's priority is not the same because of difference in resource requirements, so simple strategy is hard to meet all deployment requirements. Last but not least, it's difficult to response active and complex SFC deployment requirements because VNF is pre-deployed on server node in traditional network, so that the VNFs are not flexible to satisfy the SFCs.

In view of the dynamic and reliable deployment of the service function chain, the existing deployment methods have some shortcomings. Previously, most SFC deployment models only solve the problem which is determining deployment location or achieving optimization targets [6–9]. However, the reliability requirements have increased with the development of NFV and increasing of SFC requests. To this end, reliable and fault-tolerant SFC deployment methods are proposed to achieve high reliability [10–15]. Specifically, reliable SFC deployment methods usually use backup scheme to deploy more VNF instances on different network nodes. However, traditional backup scheme is too simple to satisfy complex SFC requests. On the other hand, more advanced and powerful reinforcement learning models have significantly achieved performance gains in SFC deployment problem recently [16–18]. However, Q-learning, a classical algorithm of reinforcement learning, needs to maintain a quite large Q-table because of large state and action set, so that the computing power of the algorithm will be affected with wasting of CPU and memory. To this end, the deep reinforcement learning which is more advanced than reinforcement learning is proposed to apply on SFC deployment problem [19–22]. Specifically, the deep reinforcement learning can choose better deployment and backup locations with learning process, and it can make accurate judgements about unknown situations in the sample.

To solve the dynamic and reliable SFC deployment problem, this paper formulates the reliable SFC deployment framework and proposes a novel priority awareness deep reinforcement learning (PA-DRL) algorithm to deploy and backup VNFs with high reliability and optimization performance. The main contributions of this paper are as follows.

1. Firstly, we propose the priority awareness framework to determine the priority of SFC and network node. On one hand, we calculate the SFC priority by average proportion of CPU resource and bandwidth resource requirements. On the other hand, we calculate the network node priority by average proportion of node resources and node degrees. It is the first time that determining the overall priority through SFC and node priority in our work.
2. Secondly, we design a determining scheme of backup location set for VNF. The determining scheme can predetermine the backup location set by SFC and node priority which are mentioned above. Specifically, the scheme can decide whether to backup and guarantee that deployment and backup locations are different of each VNF.
3. Lastly, we apply deep reinforcement learning algorithm on the process of VNF deployment and backup, then we propose PA-DRL model to deploy and backup VNFs with reliability requirement. Specifically, we design the feedback function with three factors including transmission delay, node balancing, and link balancing. Then, we evaluate the proposed PA-DRL model on the network topology which is randomly generated, and the extensive experimental results demonstrate that PA-DRL can effectively deploy SFC requests and achieve deployment performance compared with other four models including two widely used SFC deployment methods and other two backup schemes of PA-DRL.

The rest of this paper is organized as follows. Section 2 surveys the related works. In Section 3, the priority awareness framework used for determining backup scheme is described and then the proposed PA-DRL algorithm to solve reliable SFC deployment is discussed in details. In Section 4, the experimental designs and simulation results are shown. Finally, the conclusion and discussion of future work are given in Section 5.

2. Related Works

In the early studies, SFC deployment problem is usually modeled as an optimization problem, which is most solved by integer linear programming (ILP) and heuristic algorithm. On the one hand, a part of the author uses ILP algorithm or its deformation to solve the SFC deployment optimization problem. For example, Insun Jang et al. proposed a polynomial time algorithm based on linear relaxation and rounding to approximate the optimal solution of the mixed ILP [6]. Although they increased SFC acceptance rate and the service capacity, yet more optimization targets are needed, including transmission delay, load balancing and so on. Zhong et al. orchestrated SFCs across multiple data centers and proposed an ILP model to minimize the total cost [7]. They reduced the overall cost, however, the optimization result is too simple to apply in most cases. Although the ILP as a static algorithm can solve most SFC deployment optimization problem, yet it is difficult to apply on dynamic and complex SFC requests. On the other hand, other authors solve the optimization problem by heuristic algorithm. For example, Li et al. described the SFC deployment as a multi-objective and multi-restriction problem and proposed a heuristic service function chain deployment algorithm based on longest function assignment sequence [8]. Although they jointly optimized the number of VNF deployment and link bandwidth requirements, yet they ignored the dynamicity of the network. Mohammad Ali Khoshkholghi et al. formulated a multi-objective optimization model to joint VNF placement and link embedding in order to reduce deployment cost and service latency, then proposed two heuristic-based algorithms that perform close to optimal for large scale cloud/edge environments to solve the optimization problem [9]. They optimized both cost and delay, however, the method was needed to improve to apply in dynamic network environment. Heuristic algorithm can solve the SFC deployment problem which is NP-hard effectively, however, the heuristic algorithm is easy to fall into the local optimum, and the parameter setting is affected by the experience value, the number of algorithm iterations and convergence speed are difficult to guarantee.

More recently, reinforcement learning and deep reinforcement learning models have shown their superior capabilities in SFC deployment due to their advanced feature extraction and data modeling abilities. For example, Wei et al. designed a service chain mapping algorithm based on reinforcement learning to reduce the average transmission delay and improve the load balancing, the algorithm determined location of each VNF according to the network status and reward value of feedback function after the deployment [16], however, they did not improve the reliability of SFC which is significant in current network. Sang Il Kim et al. considered the consumption of CPU and memory resources, then utilized reinforcement learning to solve SFC optimization problem dynamically [17], yet the algorithm was not satisfied with delay sensitive SFC. Sebastian Troia et al. investigated the application of reinforcement learning for performing dynamic SFC resources allocation in NFV-SDN enabled metro-core optical networks [18], they decided how to reconfigure the SFCs according to state of the network and historical traffic traces. However, reinforcement learning exists the limitation that it cannot satisfy the networks which has large-scale states. To this end, deep reinforcement learning is proposed to solve the large-scale SFC deployment problem. For example, Fu et al. decomposed the complex VNFs into smaller VNF components to make more effective decisions and proposed a deep reinforcement learning-based scheme with experience replay [19], although they solved dynamic and complex SFC deployment problem effectively, yet more optimization targets and reliability needed to be considered. Li et al. proposed an adaptive deep Q-learning based SFC mapping approach to improve CPU and bandwidth resource utilization rates [20], they improved whole system resource efficiency, however, they ignored the transmission delay in SFC deployment problem. Pei et al. used deep reinforcement learning to improve network performance, including SFC acceptance rate, throughput, end-to-end delay, and load balancing [21], the method optimized a variety of objectives, however, the SFC reliability was not considered in the optimization process, which may lead to SFC failure. Although

these researches have solved the large-scale SFC deployment problem effectively, yet they ignored the reliability of SFC, and cannot satisfy to recovery SFC after network failure.

For reliability and fault-tolerance, several reliable SFC deployment models are proposed to achieve high reliability. For example, Francisco Carpio and Admela Jukan improved service reliability using jointly replications and migrations, and then proposed a N2N algorithm based on LP to improve reliability and network load balancing [10]. However, N2N cannot solve dynamic SFC requests effectively, and it cannot guarantee performance of transmission delay in results. Abdelhamid Alleg et al. proposed a reliable placement solution of SFC modeled as a mixed ILP program, they designed the SFC availability level as a target and reduce the inherent cost which is affected by diversity and redundancy [11]. However, the proposed model is only applied on statical placement and lack of adaptability of machine learning techniques, on the other hand, the algorithm does not distinguish SFC priority, so that all SFC requests applied the same scheme which would lead to low effective utilization of CPU resources. Tuan-Minh Pham et al. designed the UNIT restoration model and the PAR protection algorithm based on ILP framework, the proposed algorithms protect SFC requests from network failures in terms of both resource restoration and recovery time efficiently [12]. Although the model achieves robustness of network, yet it does not guarantee the delay in a service demand, moreover, the solution needs to consider the dynamic parameters of online demands. Mohammad Karimzadeh-Farshbafan et al. proposed a polynomial time sub-optimal algorithm named VRSP which is based on mixed ILP in multi-infrastructure network provider environment [13]. However, VRSP solves the reliable SFC deployment problem only by statical method, moreover, VRSP does not consider transmission delay and network balancing in the optimization process. Qu et al. formulated the reliable SFC deployment problem as a mixed ILP and proposed a delay-aware hybrid shortest path-based heuristic algorithm [14]. Although they achieved high reliability and low latency through the model based on ILP and heuristic algorithm, yet the model is too simple to satisfy complex SFC deployment because it does not distinguish the priority of SFC, and the algorithm cannot predict deploy and backup scheme of unknown SFC requests accurately. Ye et al. proposed a novel heuristic algorithm to efficiently address the joint topology design and mapping of reliable SFC deployment problem [15]. However, the model does not consider the delay and load balancing as optimization targets, and heuristic algorithm is easy to fall into the local optimum unlike machine learning algorithm such as reinforcement learning and deep reinforcement learning. Mao et al. proposed a deep reinforcement learning based online SFC placement method to guarantee seamless redirection after failures and ensure service reliability [22]. Although the model automatically deploys both active and standby instances in real-time, yet it does not distinguish the priority of SFC and cannot guarantee transmission delay and network load balancing. Motivated by it, this paper will for the first-time design priority-awareness model, then applies deep reinforcement learning to solve reliable SFC deployment problem in NFV framework.

3. The Proposed PA-DRL Framework

In this section, we first formulate the reliable SFC deployment problem. Next, we discuss the priority awareness framework of SFC and network node in details, and then describe the rules for determining backup scheme of each VNF. Finally, the proposed deployment and backup algorithm based on deep reinforcement learning is explained.

3.1. Reliable SFC Deployment Formulation

The reliability of SFC deployment is aimed to find a suitable backup scheme for each VNF. In this work, we define the physical network topology as an undirected graph, which is represented as $G = (V, E)$, where V is the set of network nodes and $|V| = n_V$, E denotes the set of links that connect two network nodes directly and $|E| = n_E$. In addition, we define the set of SFC requests as $Req = \{sfc_1, sfc_2, \dots, sfc_n\}$, and each SFC request consists of several different VNFs and virtual links between two adjacent VNFs.

Because network hosts VNF and virtual link via CPU and bandwidth resources respectively, therefore we mainly consider CPU resources on network nodes and bandwidth resources on physical links. Due to the difference of computing power requirements, we assume that the number of CPUs occupied by different VNF is not same. Similarly, we assume that the amount of bandwidth resources occupied by different virtual link is different too. Moreover, in order to quantify the network load balancing, we define variance of CPU' percentage occupied on all network nodes as an indicator,

$$bal_n = \sum_{i=1}^{n_V} \left(\frac{cpu_i}{CPU_i} - \sum_{j=1}^{n_V} \frac{cpu_j}{CPU_j} / n_V \right)^2 / n_V, \quad (1)$$

where cpu_i and CPU_i indicate remaining and total vCPU resources on node i respectively. And then, we define variance of bandwidth' percentage occupied on all physical links as an indicator too,

$$bal_l = \sum_{i=1}^{n_E} \left(\frac{bw_i}{BW_i} - \sum_{j=1}^{n_E} \frac{bw_j}{BW_j} / n_E \right)^2 / n_E, \quad (2)$$

where bw_i and BW_i indicate remaining and total bandwidth resources on link i respectively. In addition, SFCs consist of several different VNFs, so the length of SFCs may has a difference, as a result the delay of different SFCs will be quite different. For that reason, we use average delay of adjacent VNFs to measure the performance of delay optimization,

$$del = \sum_{i=1}^n \sum_{j=2}^{l_i} d(f_{j-1}, f_j) / \sum_{i=1}^n (l_i - 1), \quad (3)$$

where l_i denotes the length of SFC $_i$, and $d(f_{j-1}, f_j)$ represents the delay between VNF f_{j-1} and f_j . Besides, there are limits of the network resources, it means that the resource on each node and link is non-negative and no more than the maximum resource limit.

$$\begin{cases} 0 \leq cpu_n \leq CPU_n, \quad \forall n \in V \\ 0 \leq bw_l \leq BW_l, \quad \forall l \in E \end{cases}. \quad (4)$$

3.2. Priority Awareness of SFC and Network Node

Priority awareness is aimed to determine the important level of SFC and node through resource requirements and network status. Firstly, the algorithm determines the priority of SFC in high, medium, and low levels. Then, the algorithm judges the priority of nodes according to the network status. Finally, the model decides whether to allocate CPU resources to each VNF's backup. The algorithm determines the backup scheme of each VNF through these steps, then, the deployment algorithm will deploy each VNF and its backup.

In this work, we define effective resource utilization as the proportion of deployment resources and backup resources, which is expressed as,

$$e = \sum_{i=1}^n \sum_{j=1}^{l_i} c_d / \left(\sum_{i=1}^n \sum_{j=1}^{l_i} c_d + \sum_{i=1}^n \sum_{j=1}^{l_i} c_b \right), \quad (5)$$

where c_d and c_b denote CPU resources which are occupied by VNF's deployment and backup respectively. To improve the effective utilization of resource on network nodes, we propose the process of priority awareness, which aims to choose a suitable backup scheme for each VNF.

At first, we use the number of required resources to measure a SFC's priority, which is defined as,

$$p_{sfc} = \frac{\sum_{i=1}^l rc_i / m_{rc} + \sum_{i=2}^l rb(f_{i-1}, f_i) / m_{rb}}{2}, \quad (6)$$

where rc_i denotes the required CPU resources of VNF f_i , $rb(f_{i-1}, f_i)$ denotes the required bandwidth resources of virtual link between f_{i-1} and f_i , m_{rc} and m_{rb} denote the maximum required CPU and bandwidth resources of all SFCs respectively.

Secondly, we use the resource state and node degree to measure a node's priority, which is defined as,

$$p_{node} = \frac{(CPU_i - cpu_i)/CPU_i + d_i / \left(\frac{\sum_{i=1}^N d_i}{2}\right)}{2}, \quad (7)$$

where d_i denotes the node degree of node i .

Finally, we divide SFC into three priority levels, including high, medium, and low respectively. Specifically, the SFC has high priority level if $p_{sfc} > 1/3$, then the SFC has low priority level if $p_{sfc} < 1/3$, otherwise, the SFC has medium priority level. Further, we divide network node into two priority levels, including high and low respectively. Specifically, the node has high priority level if $p_{node} > 0.5$, otherwise its priority level is low.

3.3. Rules for Determining Backup Scheme

The backup scheme is choosing whether to backup for each VNF in our work, we define (V_d, V_b) as the node pair of deployment and backup node for each VNF, so that $V_b = 0$ if the VNF had no backup, otherwise $V_b \neq 0$, besides, there is a constraint $V_d \neq V_b$ to guarantee that each VNF's deployment and backup nodes are different. Table 1 presents the backup scheme in different situations, including three SFC's priority levels and two node's priority levels as mentioned above. Specifically, when SFC priority is high or SFC priority is medium and node priority is high, VNF is backed up, otherwise, when SFC priority is low or SFC priority is medium and node priority is low, VNF is not backed up.

Table 1. Backup scheme in different priority situations.

Priority of SFC and Node	Backup Scheme
$P_S = high, P_N = high$	$V_b \neq 0$
$P_S = high, P_N = low$	$V_b \neq 0$
$P_S = medium, P_N = high$	$V_b \neq 0$
$P_S = medium, P_N = low$	$V_b = 0$
$P_S = low, P_N = high$	$V_b = 0$
$P_S = low, P_N = low$	$V_b = 0$

3.4. Deep Reinforcement Learning Deployment Algorithm

Deep reinforcement learning algorithm aims to solve the problem which has large-scale state and action set. In this work, we define the state set as,

$$S(t) = (cpu_t, bw_t, vnf_t), \quad (8)$$

where cpu_t and bw_t denote the remaining resources of all nodes and links at time t respectively, and vnf_t denotes the related information of VNF at time t , including current deployment location, species of VNF at time t , and the species of next VNF at time $t + 1$. We define action set as the node pair (V_d, V_b) as mentioned before, which is represented as,

$$A(t) = (V_d, V_b), \quad (9)$$

so, the number of all actions is n_V^2 .

We aim to optimize the transmission delay of SFC and network load balancing, including load balancing of nodes and links. To this end, we optimize all optimization

targets by minimizing the maximum value of delay, node, and link load balancing, so we define the feedback function as,

$$\begin{cases} M_D = \max(d(f_i, f_{i+1})) \\ M_C = \max(CPU_i - cpu_i) \\ M_B = \max(BW_i - bw_i) \\ R(t) = \frac{\alpha}{M_D} + \frac{\beta}{M_C} + \frac{\gamma}{M_B} \end{cases} \quad (10)$$

where α, β, γ are the weight coefficients, and need to meet this constraint,

$$\begin{cases} \alpha > 0, \beta > 0, \gamma > 0 \\ \alpha + \beta + \gamma = 1 \end{cases} \quad (11)$$

At first, we initialize all the parameter ω of neural network. Then we choose action through $\varepsilon - greedy$ method, which is defined as,

$$\varepsilon = \frac{K}{100} \quad (12)$$

where K is the exploration times. Finally, we store a 5-tuple $(S(t), A(t), S(t+1), R(t), is_{end})$ in replay memory buffer D after every action selection, where is_{end} is a binary indicates whether the state is terminated,

$$\begin{cases} is_{end} = 1, \text{ state is terminated} \\ is_{end} = 0, \text{ state is not terminated} \end{cases} \quad (13)$$

The network randomly selects several samples from D in each episode, then the Q value will be calculated by,

$$Q(t) = \begin{cases} R(t), \text{ while } is_{end} = 1 \\ R(t) + \delta * \max(Q(S(t+1), A(t+1), \omega)) \\ \text{while } is_{end} = 0 \end{cases} \quad (14)$$

where ω is the parameter of neural network, which will be updated in each episode by backward gradient propagation, and δ represents the learning rate. The training algorithm for our model is as Algorithm 1.

Algorithm 1 PA-DRL training algorithm

Input: SFC request with the requirement of VNFs and virtual links

Output: Deployment and backup locations for each VNF of SFC

Initialize the neural network N with the parameter ω , and clear the replay memory buffer D .

while $is_{end} \neq 0$ **do**

 Calculate SFC priority level p_{sfc} as (6)

 Calculate node priority level p_{node} as (7)

 Determine action set by Table 1

 Select action $A(t)$ by $\varepsilon - greedy$ method

 Calculate reward $R(t)$ as (10)

 Store the sample $(S(t), A(t), S(t+1), R(t), is_{end})$ in memory D

 Randomly select samples from D then calculate Q as (14)

 Train the network N and update ω by using back propagation of the loss function

end while

4. Simulation Results and Analysis

In this section, the network topology is introduced firstly, and then the parameters are described, and finally the optimization performance and analysis are presented and discussed.

4.1. Network Topology

To formulate the deployment process of SFC, we randomly generate a network topology, which has 6 network nodes and 7 physical links, the network topology diagram is showed in Figure 1. The black number represents CPU capacity of each network node, the red and blue numbers next by solid line represent bandwidth capacity and transmission delay of each physical link respectively. The CPU resources of nodes and bandwidth resources of links are set by random method, and we assume all link delay in order to facilitate the measurement of transmission delay.

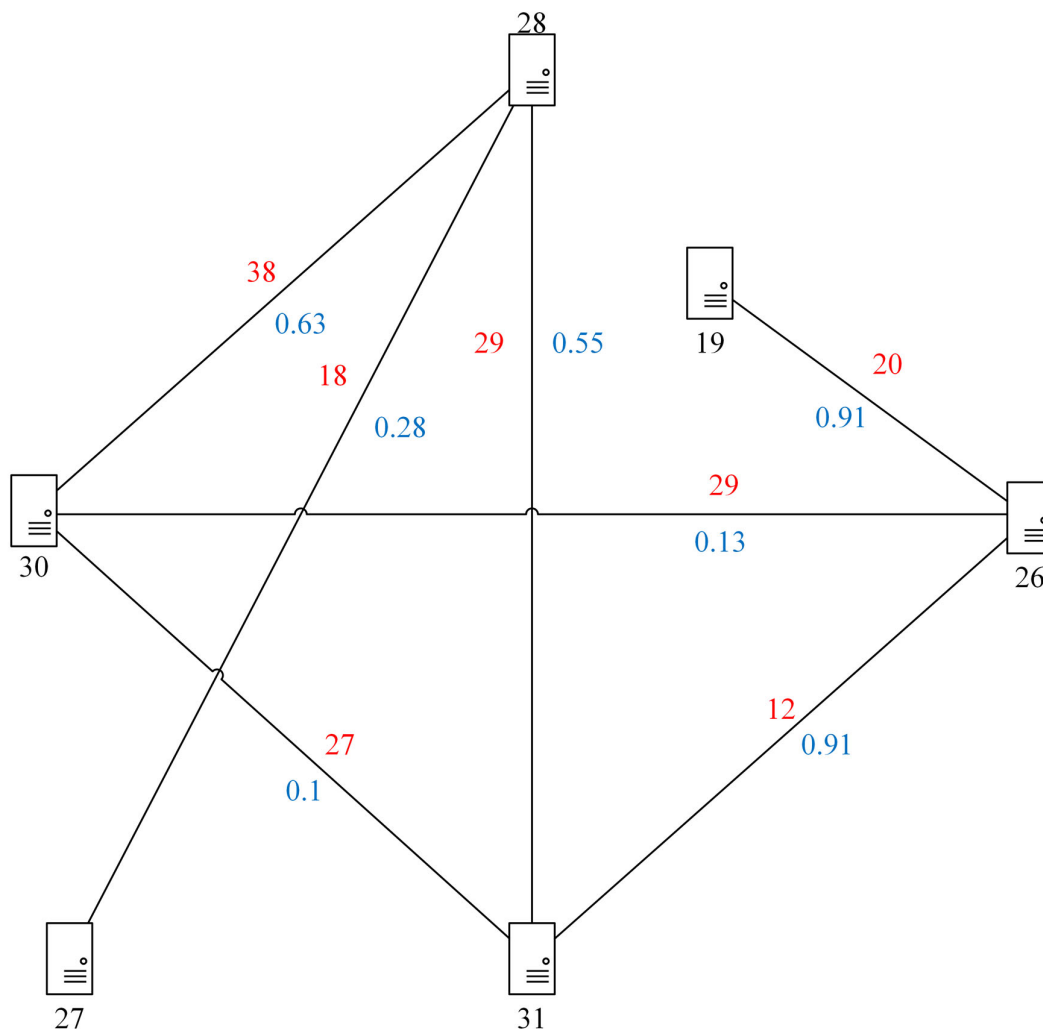


Figure 1. Network topology with CPU capacity, bandwidth capacity, and link delay.

4.2. Experimental Settings

4.2.1. Formulation Environment

The experiments are implemented with MATLAB platform on a 64-bit computer with Intel Core CPU i5-9400F @ 2.9 GHz, 16-GB RAM. In order to avoid linear results due to linear SFC requests number, we randomly generate 4000, 6000, 8000, 10,000, 15,000, 20,000, 25,000, 30,000, 40,000, 50,000 SFC requests respectively.

4.2.2. Comparison Algorithms

In order to measure the performance of PA-DRL algorithm which is proposed in this work, two state-of-the-art methods including PA-RL, N2N are selected as the baselines:

PA-RL, we apply priority-awareness model to RL proposed in [16].

N2N, which is designed based on LP algorithm in [10].

In addition, we implement three version of our solution:

ALL-BACKUP, backs up all VNFs of each SFC no matter what their priority level is.

RND-BACKUP, randomly determines VNFs of each SFC to back up.

PA-DRL, chooses VNFs of each SFC to back up through priority awareness framework as mentioned before.

4.2.3. Related Parameters

VNFs and virtual links. We assume there are five varieties of VNFs, including Firewall (FW), Net Address Transition (NAT), Deep Packet Inspection (DPI), Domain Name System (DNS), and Load Balancing (LB). The required resources of each VNFs and virtual links are presented as Tables 2 and 3. For example, the SFC FW-NAT-DPI has 4 units CPU and 3 units bandwidth resources requirement respectively. So that, the SFC NAT-DPI-DNS-LB-FW has most resources requirement, the required CPU and bandwidth resources of this SFC are 10 units CPUs and 10 units bandwidths respectively.

Table 2. All VNFs and their respective CPU resource requirements.

Varieties of VNF	CPU Resource Requirements
FW	1 unit
NAT	1 unit
DPI	2 units
DNS	3 units
LB	3 units

Table 3. All virtual links and their respective bandwidth resource requirements.

Varieties of Virtual Links between Adjacent VNFs	Bandwidth Resource Requirements
FW-NAT (NAT-FW)	1 unit
FW-DPI (DPI-FW)	1 unit
FW-DNS (DNS-FW)	2 units
FW-LB (LB-FW)	2 units
NAT-DPI (DPI-NAT)	2 units
NAT-DNS (DNS-NAT)	2 units
NAT-LB (LB-NAT)	2 units
DPI-DNS (DNS-DPI)	3 units
DPI-LB (LB-DPI)	2 units
DNS-LB (LB-DNS)	3 units

4.2.4. Parameters of Algorithm

On one hand, through several times of adjusting according to the experimental performance, we set the parameters of feedback function (10) as $\alpha = 0.1$, $\beta = 0.45$, and $\gamma = 0.45$ respectively, and then, we set up two hidden layers with 100 neurons for the neural network. On the other hand, in the training process of algorithm, the learning rate of deep reinforcement learning is usually set as $[0.9, 0.999]$, so that the learning effect will be better than other value the learning rate is. In this work, we set the learning rate δ of proposed deep reinforcement learning process as $\delta = 0.99$. Moreover, we set the size of replay memory buffer D is 8000, and the number of samples for each training episode is 1000, so as to ensure that the correlation between samples is small and the distribution of samples is uniform. In addition, to guarantee that there are enough resources to deploy VNFs and virtual links, we assume the life cycle of SFC is the time of 20 SFCs' living time, which means that the system need to release CPU and bandwidth resources of sfc_{i-20} when sfc_i comes ($i > 20$).

4.3. Experimental Results

We get the results of runtime, effective utilization of CPU resource, average survival rate of SFC, average delay of adjacent VNFs, nodes' load balancing variance, and links' load balancing variance under each SFC requests number as mentioned before.

4.3.1. Runtime

Figure 2 shows the runtime for PA-DRL and PA-RL models in terms of deploy SFCs number. It can be easily observed that the runtime of PA-DRL is much lower than PA-RL under each SFC requests number, and both curves show an increasing trend. Because the PA-RL need to maintain a large-scale Q -table in the process of learning, and need to traverse the whole Q -table when every time that an action is selected, so the time consumption is quite large. However, the PA-DRL proposed in this work uses the deep reinforcement learning framework, eliminates the curse of dimensionality in PA-RL, so that it can save a lot of time when selecting an action. On the other hand, the two methods need to complete more times of selecting action when SFC requests number is getting bigger, so the runtimes of proposed PA-DRL and PA-RL algorithms grow linearly.

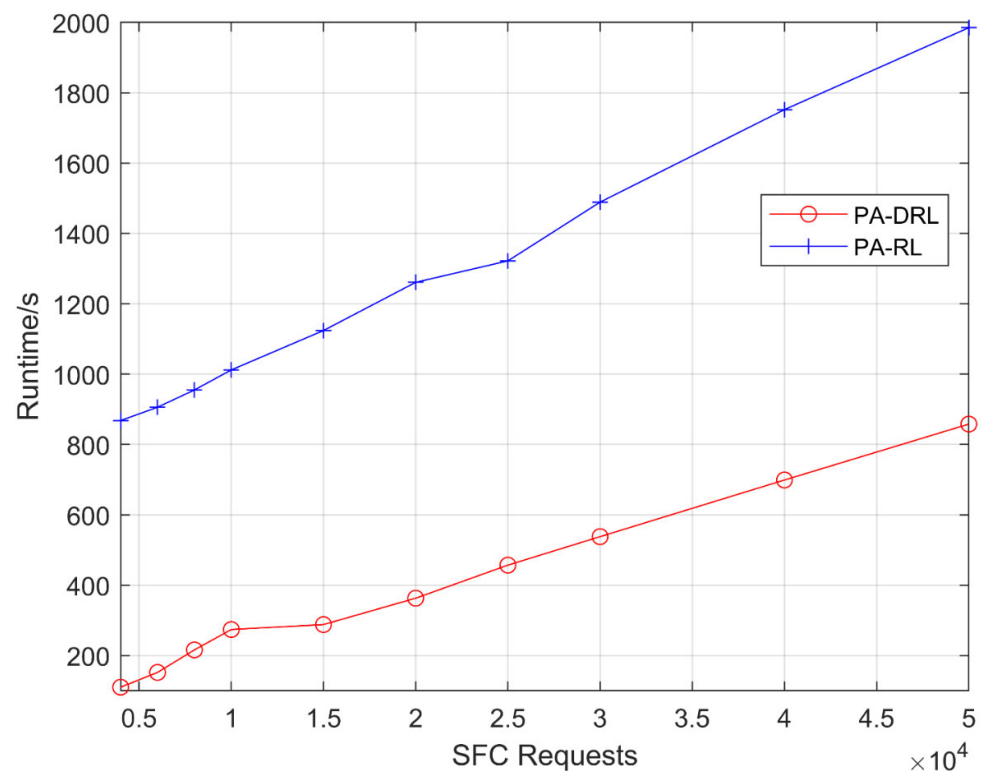


Figure 2. Runtime of PA-DRL and PA-RL under each SFC requests number.

4.3.2. Resources' Efficient Utilization

Figure 3 shows the resources' efficient utilization of proposed PA-DRL method and other four baseline algorithms. It is easily to observe that the performance of PA-DRL is better than other algorithms. Intuitively, there are three facts in Figure 3. Firstly, the efficient utilization of PA-DRL is the best, because PA-DRL determines the priority of nodes by node resource utilization and node degree, this process can avoid the circumstance that the VNF which is not important on the medium priority SFC occupies CPU resources, so that PA-DRL can avoid wasting of resources and improve efficient utilization of CPU resource. Secondly, the efficient utilizations of RND-BACKUP and PA-RL are better than N2N, because they utilize deep reinforcement learning and reinforcement learning algorithm respectively, which can improve the performance through learning process, yet N2N is based on LP algorithm, which is weaker than machine learning on optimization perfor-

mance. Last but not least, the efficient utilization performance of ALL-BACKUP method is the worst, because ALL-BACKUP backs up all VNFs no matter which the priority level is, thus, the resources' efficient utilization is keeping at 50% under each SFC requests number.

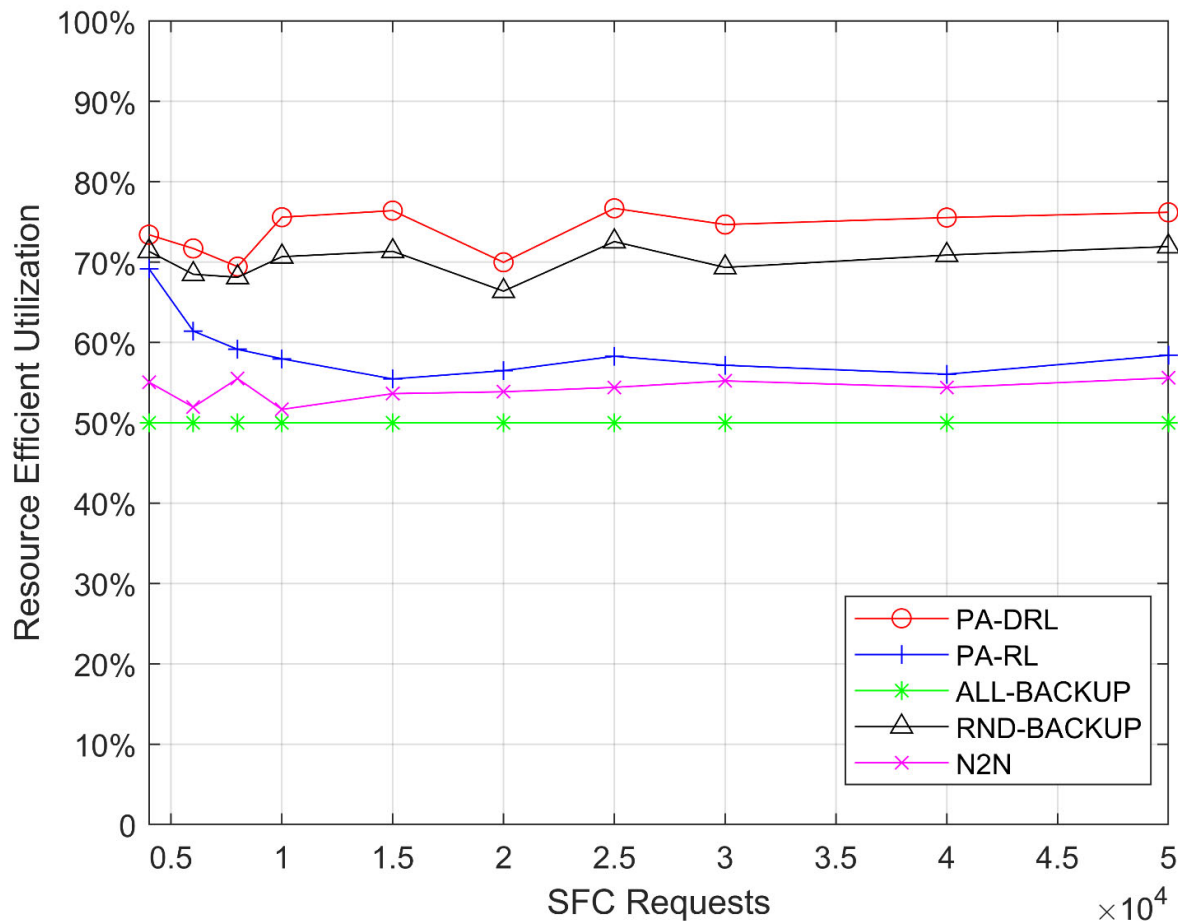


Figure 3. Efficient utilization of CPU resource under each SFC requests number.

4.3.3. Average Survival Rate

Figure 4 shows the average survival rate, which is defined as average SFC survival rate when each node fails, for PA-DRL, ALL-BACKUP, RND-BACKUP, PA-RL, and N2N models in terms of SFCs number, the average survival rate in this work is defined as the average proportion of survival SFCs and all SFC requests when each network node fails. The Figure 4 contributes four facts. At first, it can be easily observed that average survival rate of ALL-BACKUP method is the best, because each VNF in ALL-BACKUP method has one backup on other one node, so that every SFC can be recovered when each network node fails, and the average survival rate is keeping at 100% under all SFC requests. Then, the performance of PA-DRL is better than other algorithms except ALL-BACKUP, because the PA-DRL determines the backup scheme of each VNF by priority of SFC and network node, so that PA-DRL can improve the destruction resistance of high priority SFCs, furthermore, PA-DRL can avoid the circumstance of large-scale SFC failure caused by physical network malfunction. Thirdly, the average survival rate of PA-RL is lower than PA-DRL, because the reinforcement learning algorithm cannot make accurate judgements about unknown situations in the sample compared with deep reinforcement learning. Last but not least, the performances of RND-BACKUP and N2N models are the worst, because their backup schemes are worse than PA-DRL, and PA-DRL's deep reinforcement learning algorithm is more advanced than N2N.

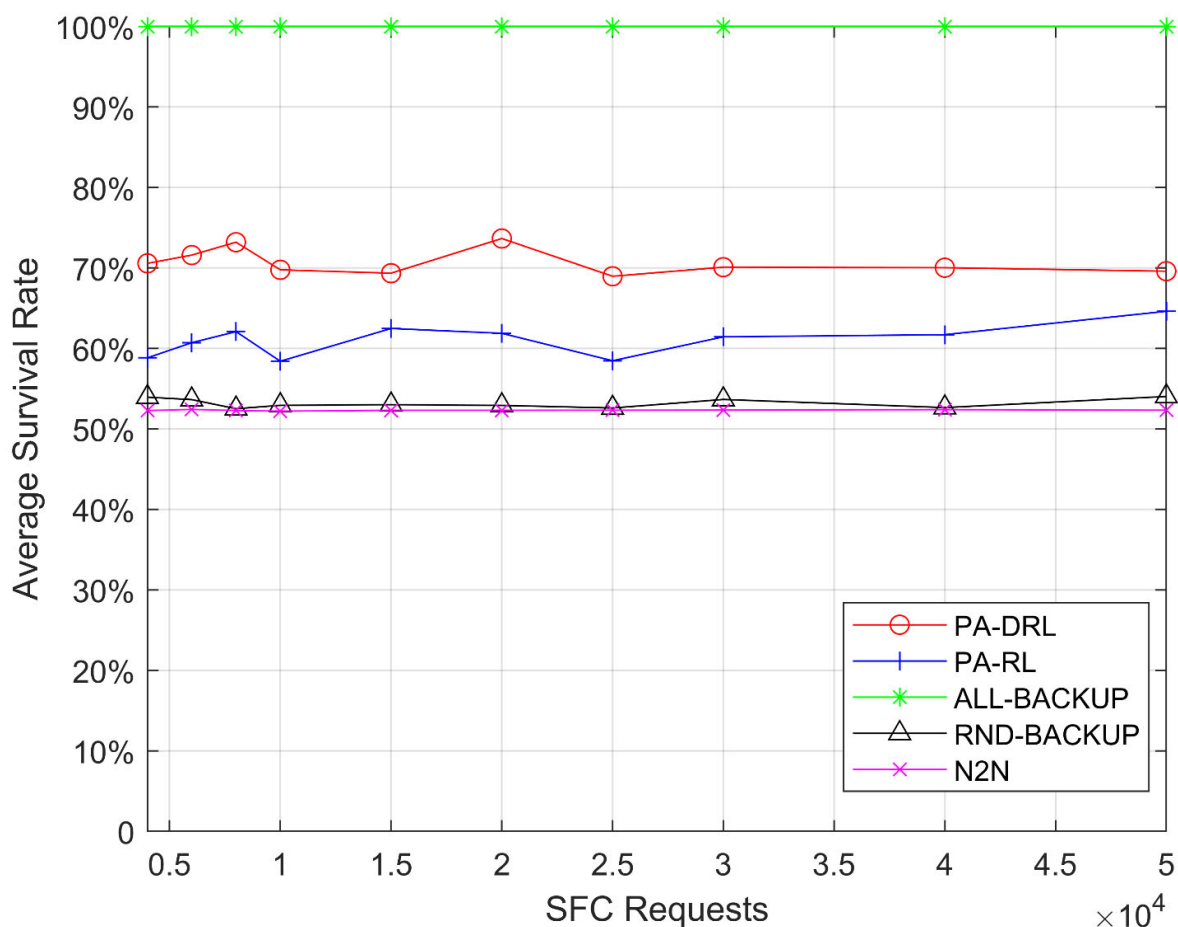


Figure 4. Average survival rate after failure of any node under each SFC requests number.

4.3.4. Average Delay

Figure 5 shows the average delay of proposed PA-DRL algorithm and four compared algorithms. From Figure 5, it is easily to observe three facts. Firstly, the average delay of N2N is the largest, this is because N2N is based on LP algorithm, this algorithm is satisfied for static environment and cannot improve the optimization performance gradually like deep reinforcement learning and reinforcement learning. Therefore, the performance of N2N is the worst under each SFC requests number. Secondly, this result agrees with the theoretical analysis above that the proposed PA-DRL algorithm is better than PA-RL, because we use deep reinforcement learning algorithm which is more advanced than reinforcement learning, and define maximum delay of adjacent VNFs as one feedback factor of feedback function. Thirdly, the average delay of ALL-BACKUP is less than RND-BACKUP algorithm, because ALL-BACKUP algorithm will back up all VNFs and consume more network resources, the algorithm is more inclined to deploy the VNFs of the same SFC on close nodes, and deploy the backup VNFs on other nodes, so as to balance the network load and reduce the transmission delay.

4.3.5. Load Balancing of Network Nodes

Figure 6 shows the network node load variance for PA-DRL, ALL-BACKUP, RND-BACKUP, RL, and N2N models in terms of SFCs number. Firstly, it can be easily observed that the performance of PA-DRL is the best, because we use maximum CPU usage as optimization goal in feedback function, the performance will be the best through learning process. Secondly, the variance of RND-BACKUP and ALL-BACKUP models are higher than PA-DRL and lower than PA-RL. On one hand, because they utilize maximum node load to optimize the load balancing of nodes, and then, deep reinforcement learning

algorithm is more advanced than reinforcement learning in choosing action for unknown samples. On the other hand, because they determine the backup scheme without priority level, the backup scheme of each VNF is worse than PA-DRL, thus, their performances of nodes' load balancing are worse than PA-DRL. Thirdly, the PA-RL's performance is worse than PA-DRL algorithm, because reinforcement learning is worse than deep reinforcement learning when making accurate judgements about unknown situations in the sample. Last but not least, the variance of N2N algorithm is the highest, because the feedback functions of PA-DRL and PA-RL consider nodes' load balancing by maximum CPU usage, these algorithms tend to deploy VNF on node with more resources. Thus, they are more advanced than N2N's optimization function under each SFC requests number.

We use the D-value between maximum and average resource utilization to measure the network load balancing, therefore, the smaller the D-value is, the better the load balancing is. In order to measure the load balancing of nodes, Figure 7 shows the D-value between maximum and average node resource utilization of proposed PA-DRL algorithm and other four baseline algorithms. From Figure 7, it is easily to observe three facts. First of all, it can be seen that the performance of PA-DRL is the best, because PA-DRL utilizes deep reinforcement learning to deploy VNFs and backups, which is more efficient in optimizing nodes' load balancing. Then, RND-BACKUP, ALL-BACKUP, and PA-RL algorithms' performances are similar, because PA-RL determines the backup scheme by priority, which is more advanced than RND-BACKUP and ALL-BACKUP, however, the optimization efficiency of reinforcement learning in large-scale unknown cases is weaker than that of deep reinforcement learning. Thirdly, the performance of N2N is the worst, because N2N is based on LP algorithm, which cannot optimize the target with better and better effect.

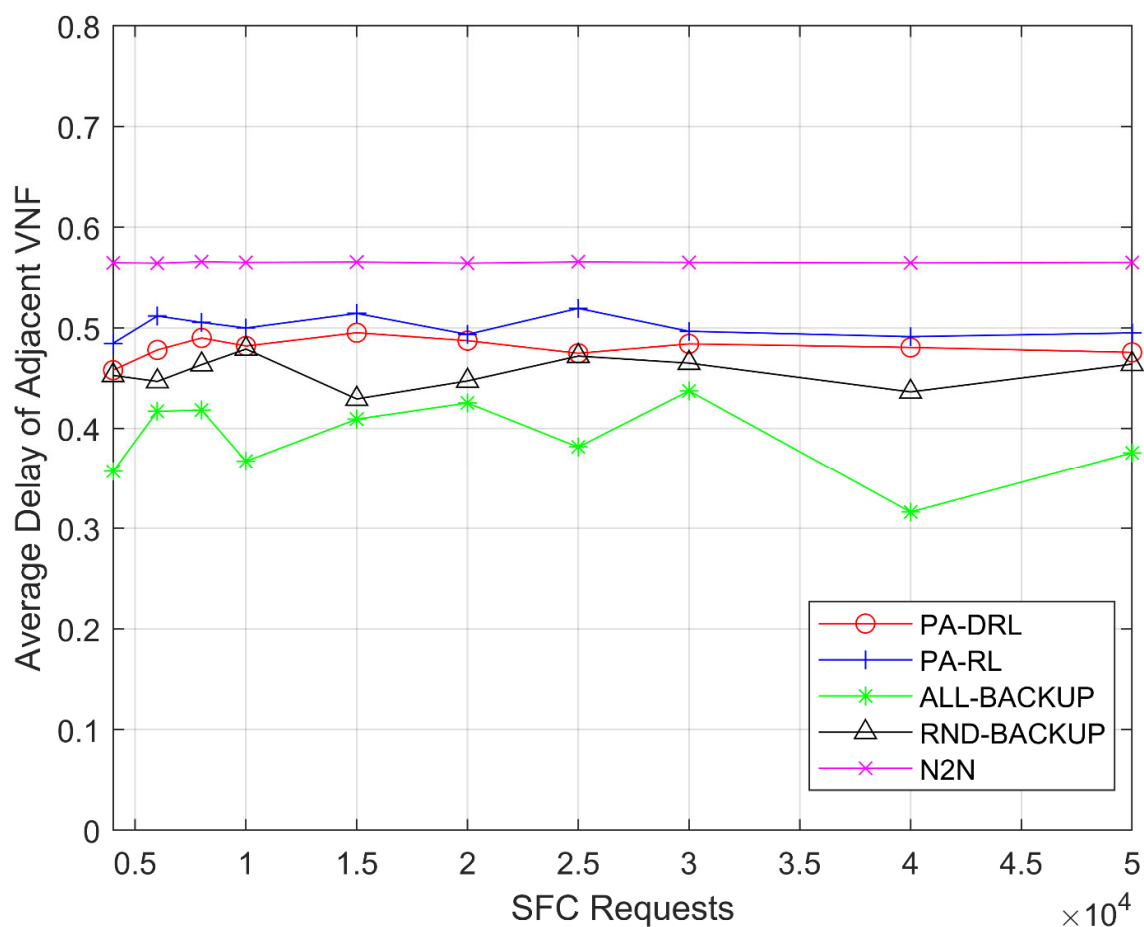


Figure 5. Average delay of adjacent VNFs under each SFC requests number.

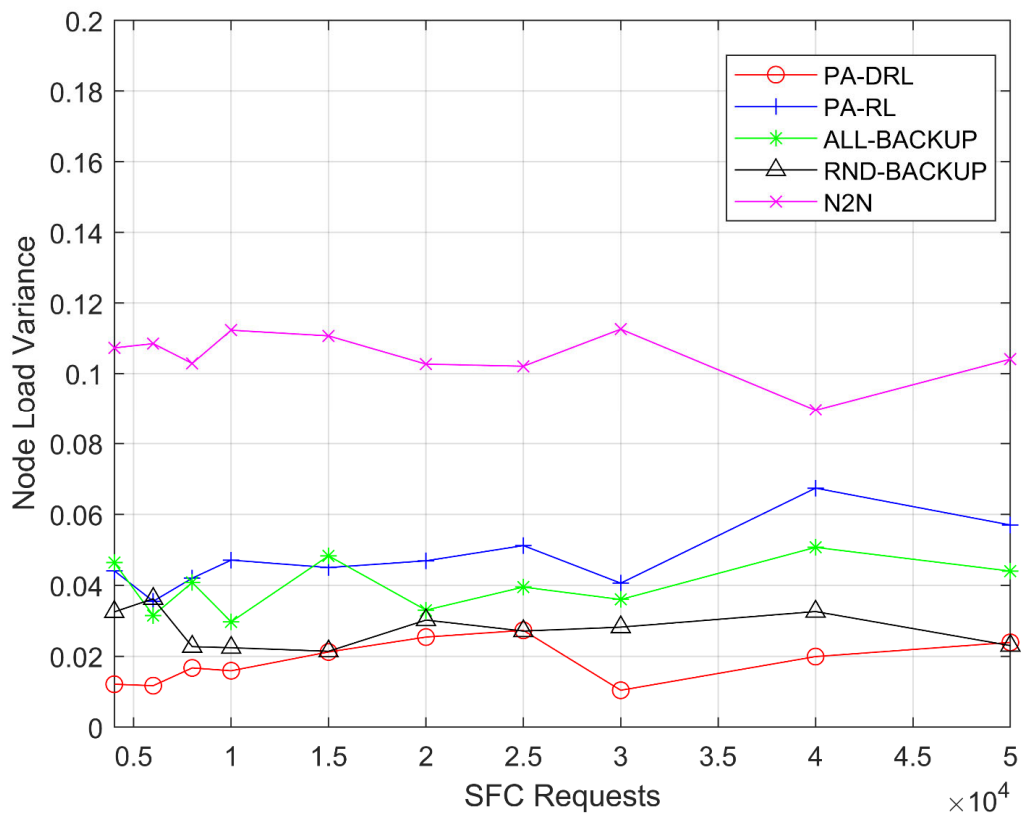


Figure 6. Node load balancing variance under each SFC requests number.

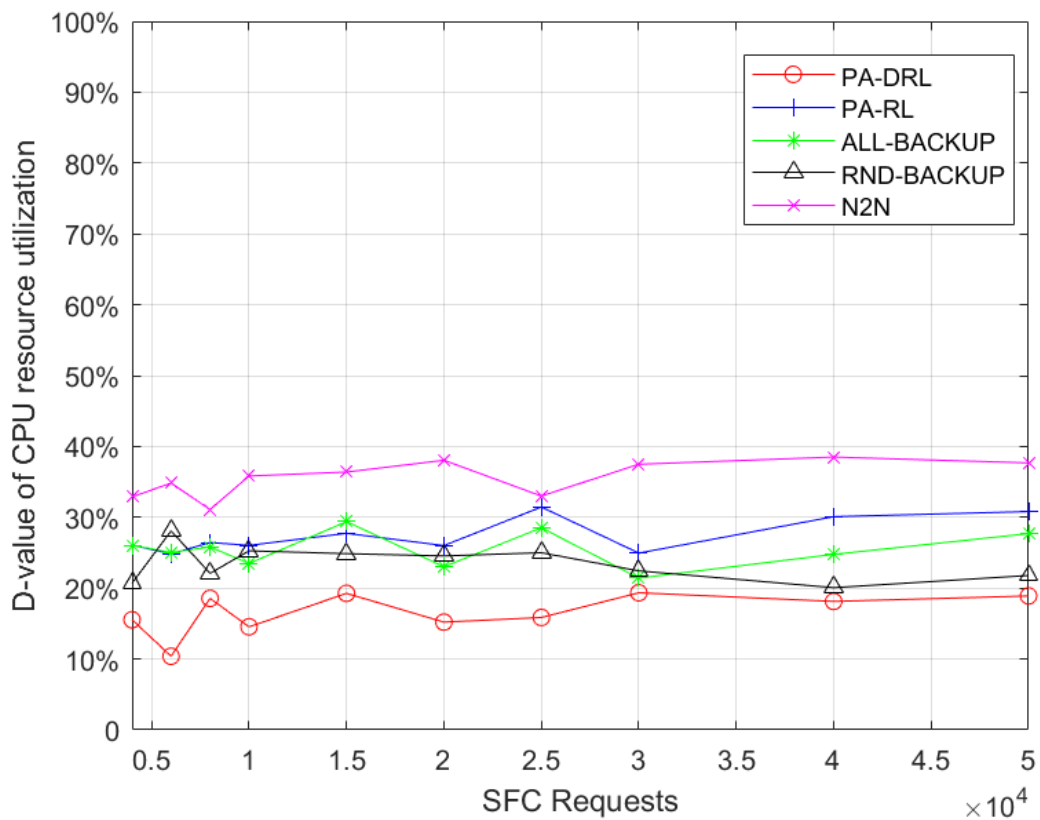


Figure 7. D-value between maximum and average nodes' resource utilization under each SFC requests number.

4.3.6. Load Balancing of Network Links

Figure 8 shows the network link load variance for PA-DRL, ALL-BACKUP, RND-BACKUP, RL, and N2N models in terms of SFCs number. From Figure 8, it is easily to observe that the PA-DRL's variance is the lowest like Figure 6, because we use maximum bandwidth usage as optimization goal in feedback function, the performance will be the best through learning process. The RND-BACKUP and ALL-BACKUP models' curves are higher than PA-DRL, because the deployment location determining schemes of them are not better than PA-DRL, so the link which is chose to deploy virtual link is not the best, which leads to poor results than PA-DRL. The PA-RL's curve is higher than PA-DRL, because deep reinforcement learning is better than reinforcement learning when making accurate judgements about unknown situations in the sample. The N2N's performance is the worst, because the feedback functions of PA-DRL and PA-RL are considered links' load balancing by maximum bandwidth usage, which will get better optimization result in large-scale SFC requests, so they are more advanced than N2N's optimization function.

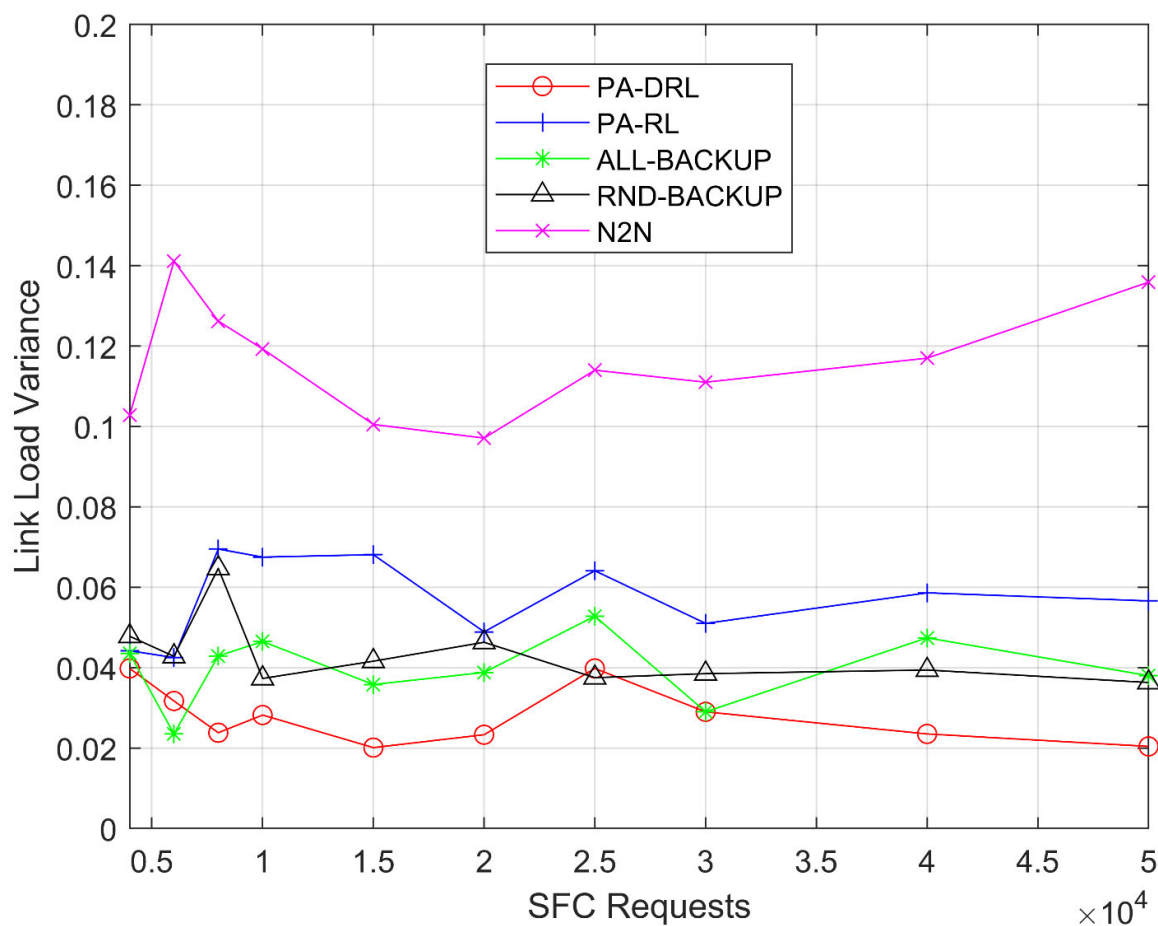


Figure 8. Link load balancing variance under each SFC requests number.

As mentioned above, in order to measure the load balancing of links, Figure 9 shows the D-value between maximum and average link resource utilization of proposed PA-DRL algorithm and other four baseline algorithms. Intuitively, there are three facts in Figure 9. Firstly, the D-value of PA-DRL is the lowest, because PA-DRL uses deep reinforcement learning to process deployment, which can optimize the target better compared with other algorithms, so the performance is the best. Secondly, RND-BACKUP, ALL-BACKUP, and PA-RL algorithms' performances are similar and better than N2N, because they all use machine learning to optimize target, which can get better performance in large-scale requests environment than LP algorithm. Finally, the performance of N2N is the worst, because

N2N is based on LP algorithm, its optimization performance can only be maintained to a certain extent, and cannot get better results with the increase of SFC requests.

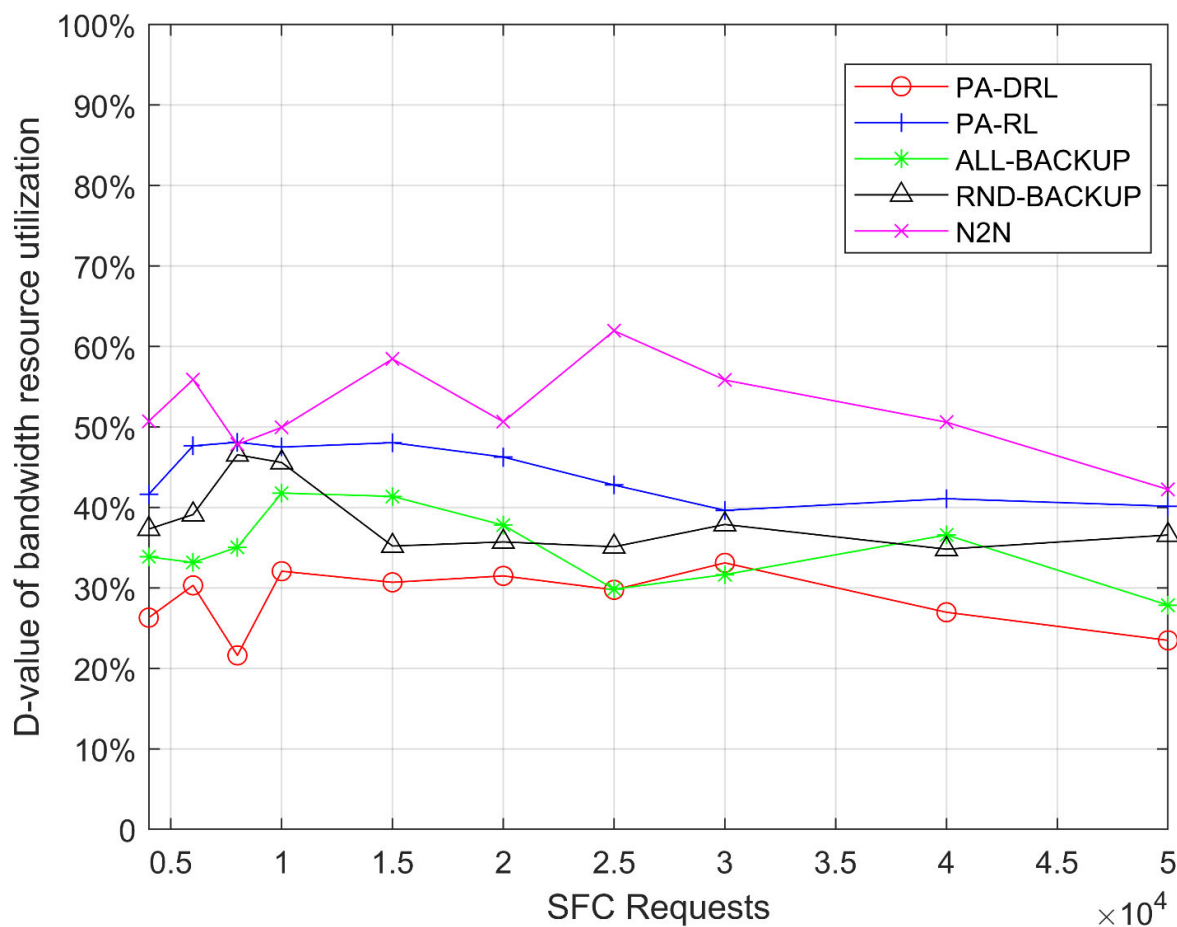


Figure 9. D-value between maximum and average links' resource utilization under each SFC requests number.

5. Conclusions

In this paper, we propose a novel PA-DRL for reliable SFC deployment. PA-DRL determines the backup scheme through the priority of SFC and network node, then PA-DRL use deep reinforcement learning algorithm to choose deployment and backup locations of VNF, and update the neural network dynamically. The results show that compared with the other four models (ALL-BACKUP, RND-BACKUP, PA-RL, and N2N), PA-DRL can improve the reliability of SFC and the efficient utilization of CPU resource. In addition, PA-DRL uses delay and network load balancing as feedback factors, so that it can reduce the transmission delay, and improve the load balancing of nodes and links to a certain extent.

Although our proposed PA-DRL method can achieve reliable SFC deployment, there are still some shortcomings that need to be resolved. At first, the PA-DRL deploys SFCs on the network topology which is designed in our work, the experiment results show that the performance of PA-DRL is better than others, however the network topology is too small to have limitations. Therefore, the PA-DRL model need to apply on another expanded network topology. Secondly, the parameters of neural network and deep reinforcement learning are designed through minor adjustments, including hidden layers, neurons number of neural networks, and weight coefficients of deep reinforcement learning, we can't guarantee the performance is the best. Therefore, we need to adjust the parameters through more experimental comparison. Finally, we will apply PA-DRL method on actual network environment, deploy our algorithm on hardware devices to achieve the experimental results in line with actual conditions.

Author Contributions: Conceptualization, K.W.; Investigation, K.W.; Methodology, K.W.; Writing-original draft, K.W.; Funding acquisition, H.Q. and J.Z.; Writing-review and editing, H.Q. and J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Nature Science Foundation of China under Grant 61531013 and by the National Science and National Key R&D Program of China under Grant 2018YFB1800300.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yi, B.; Wang, X.; Li, K.; Das, S.K.; Huang, M. A comprehensive survey of Network Function Virtualization. *Comput. Netw.* **2018**, *133*, 212–262. [CrossRef]
2. Joshi, K.; Benson, T. Network Function Virtualization. *IEEE Internet Comput.* **2016**, *20*, 7–9. [CrossRef]
3. Halpern, J.; Pignataro, C. Service Function Chaining Architecture, Document RFC 7665 of The IETF Service Function Chaining Working Group. Available online: <http://datatracker.ietf.org/doc/rfc7665/> (accessed on 21 January 2020).
4. Li, Y.; Chen, M. Software-Defined Network Function Virtualization: A Survey. *IEEE Access* **2015**, *3*, 2542–2553. [CrossRef]
5. Quinn, P.; Guichard, J. Service Function Chaining: Creating a Service Plane via Network Service Headers. *Computer* **2014**, *47*, 38–44. [CrossRef]
6. Jang, I.; Suh, D.; Pack, S.; Dan, G. Joint Optimization of Service Function Placement and Flow Distribution for Service Function Chaining. *IEEE J. Sel. Areas Commun.* **2017**, *35*, 2532–2541. [CrossRef]
7. Zhong, X.X.; Wang, Y.; Qiu, X.S. Service function chain orchestration across multiple clouds. *China Commun.* **2018**, *15*, 99–116. [CrossRef]
8. Li, D.; Lan, J.; Wang, P. Joint service function chain deploying and path selection for bandwidth saving and VNF reuse. *Int. J. Commun. Syst.* **2018**, *31*, e3523. [CrossRef]
9. Khoshkholghi, M.A.; Khan, M.G.; Noghani, K.A.; Taheri, J.; Bhamare, D.; Kassler, A.; Xiang, Z.; Deng, S.; Yang, X. Service Function Chain Placement for Joint Cost and Latency Optimization. *Mob. Netw. Appl.* **2020**, *25*, 2191–2205. [CrossRef]
10. Carpio, F.; Jukan, A. Improving reliability of service function chains with combined VNF migrations and replications. *arXiv* **2017**, arXiv:1711.08965.
11. Abdelhamid, A.; Toufik, A.; Mohamed, M.; Raouf, B. Joint diversity and redundancy for resilient service function chain provisioning. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 1490–1504.
12. Tuan-Minh, P.; Serge, F.; Thi-Thuy-Lien, N.; Hoai-Nam, C. Modeling and analysis of robust service composition for network functions virtualization. *Comput. Netw.* **2020**, *166*, e106989.
13. Karimzadeh-Farshbafan, M.; Shah-Mansouri, V.; Niyato, D. Reliability Aware Service Placement Using a Viterbi-Based Algorithm. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 622–636. [CrossRef]
14. Qu, L.; Assi, C.; Khabbaz, M.J.; Ye, Y. Reliability-Aware Service Function Chaining with Function Decomposition and Multipath Routing. *IEEE Trans. Netw. Serv. Manag.* **2019**, *17*, 835–848. [CrossRef]
15. Ye, Z.; Cao, X.; Wang, J.; Yu, H.-F.; Qiao, C. Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization. *IEEE Netw.* **2016**, *30*, 81–87. [CrossRef]
16. Wei, L.; Huang, T.; Zhang, J.; Wang, Z.N.; Liu, J.; Liu, Y.J. Service chain mapping algorithm based on reinforcement learning. *J. Commun.* **2018**, *39*, 90–100.
17. Kim, S.I.; Kim, H.S. A research on dynamic service function chaining based on reinforcement learning using resource usage. In Proceedings of the 2017 9th International Conference on Ubiquitous and Future Networks (ICUFN), Milan, Italy, 4–7 July 2017; pp. 582–586.
18. Troia, S.; Alvizu, R.; Maier, G. Reinforcement Learning for Service Function Chain Reconfiguration in NFV-SDN Metro-Core Optical Networks. *IEEE Access* **2019**, *7*, 167944–167957. [CrossRef]
19. Fu, X.; Yu, F.R.; Wang, J.; Qi, Q.; Liao, J. Service Function Chain Embedding for NFV-Enabled IoT Based on Deep Reinforcement Learning. *IEEE Commun. Mag.* **2019**, *57*, 102–108. [CrossRef]
20. Li, G.; Feng, B.; Zhou, H.; Zhang, Y.; Sood, K.; Yu, S. Adaptive service function chaining mappings in 5G using deep Q-learning. *Comput. Commun.* **2020**, *152*, 305–315. [CrossRef]
21. Pei, J.; Hong, P.; Pan, M.; Liu, J.; Zhou, J. Optimal VNF Placement via Deep Reinforcement Learning in SDN/NFV-Enabled Networks. *IEEE J. Sel. Areas Commun.* **2019**, *38*, 263–278. [CrossRef]
22. Mao, W.X.; Wang, L.; Zhao, J.; Xu, Y.D. Online fault-tolerant VNF chain placement: A deep reinforcement learning approach. In Proceedings of the 2020 IFIP Networking Conference (Networking), Paris, France, 22–26 July 2020.