Method for VNF Placement for Service Function Chaining optimized in the NFV Environment

Sang Il Kim
Electronics and Communications Engineering
KwangWoon University, Seoul, Korea
rlatkd234@kw.ac.kr

Hwa Sung Kim
Electronics and Communications Engineering
KwangWoon University, Seoul, Korea
hwkim@kw.ac.kr

Abstract- **The NFV environment allows more efficient acceptance of multiple users' requirements, making physical resources such as a CPU, memory, storage and network cards resourceful based on virtual machine technology. Since the service chaining technology that connects virtualized service functions sequentially as the users want and provides the service operates as different users' virtual machines construct different virtual network topologies on a fixed physical network structure, there is a problem in which inefficient service placement leads to resource waste and performance degradation in physical equipment. Thus, in this paper, a study was conducted on a method for VNF Placement to place the VNF in the optimum physical node when the present physical resource situation is monitored and SFC is generated, using an optimization algorithm with the monitored resource information.**

Keywords— **NFV, SDN, Reinforcement Learning, Dynamic SFC**

I. INTRODUCTION

Network traffic and devices are increasing with interest in multimedia service and IoT, and, consequently, the advancement of network operation is required. Telecommunication carriers are paying attention to NFV as a core technique for the next-generation network and 5G core technology. NFV is a technique that separates software functions from hardware-dependent network devices, which are components of the existing network, and provides services through general-purpose server equipment-based infrastructure. By managing network functions with software, carriers can carry out service deployment freely, and respond quickly to service requirements and traffic change. In addition, carriers can carry out centralized resource management efficiently, thereby obtaining the effects of saving CAPEX and OPEX [1].

The standardization of NFV is led by the European Telecommunications Standards Institute (ETSI), which holds the General Assembly and technical meetings every quarter on the basis of the 2013 NFV Industry Specification Group (ISG) initiative. At Phase-1, ETSI's first stage of NFV standardization, top 4 working groups were organized to define the concepts, requirements, use cases and architecture of NFV, aiming at establishing the concept of NFV and defining technical specifications. At Phase-2 beginning in 2015, the focus has been given to standards development and related support activities for guaranteeing interoperability in diverse vendor environments [2].

In keeping with the progress of NFV, service function chaining (SFC) for specific network service appeared, and SFC refers to a technique for the sequential abstraction of service functions. Service chaining is formed by selecting service function instances of a specific network node in order to make a service graph called Service Function Path. That is, it is a technique for routing data packets by means of network functions to be applied to the data packets. The main functions of service chaining include the Service Classifier, which analyzes and identifies the service chain of traffic according as the traffic occurs, and the Service Function Forwarder, which forwards service chain-based inflow traffic in a determined order of service functions. This service chaining technique can smoothly process control and management in combination with Software Defined Networking (SDN) [2]. The service chaining techniques are mainly discussed by the Internet Engineering Task Force (IETF) Service Function Chaining (SFC) Working Group (WG), which defines problems, use case, and standards for functions and architecture in connection with service chaining techniques. Among them, dynamic service function chaining is being discussed lively. The dynamic service function chaining is discussed to solve problems, which arise from the existing users' direct chaining of service functions, by using various methods such as machine learning, artificial intelligence, and ontology.

Thus, the present author investigated in the previous study a dynamic service function chaining method that allows the controller to generate service flow suitable to situation by learning, through reinforcement learning, the usage of physical resources used by every node and the usage of virtual resources used by every service function. However, there was a problem that different performances were shown according to the performances of physical node or application because a weight value derived from experiment was used for the weight value applied to the *r* value [3]. Therefore, this paper applied as the weight value applied to the *r* value, a weight value calculated on the basis of remaining physical resources, not a weight value derived from experiment, and handled it flexibly according to the amount of remaining resources, not being influenced by the performance of a specific physical node or the performance of application, thereby enabling more efficient dynamic service function chaining.

This paper is organized as follows: In Chapter 2, related studies are described; and in Chapter 3, elements and a method for the dynamic service function chaining proposed by this paper are explained. In Chapter 4, the evaluation of the performance of the proposed method is preceded with; and in Chapter 5, the conclusion is described.

II. BACKGROUND

A. Reinforcement Learning Algorithm

The reinforcement learning refers to learning a necessary action by carrying out the action to achieve a goal in an environment, which is a set of many states, and being rewarded. In the environment, there are many states necessary or unnecessary for achieving a goal. The agent experiences every state, and in case of achieving a goal, the agent will be rewarded from the environment; however, in case of failing to achieve a goal, the agent cannot be rewarded. Therefore, the agent might experience a lot of trial and error, and should experience every state. The typical algorithm of reinforcement learning is Q-learning, and the algorithm is as shown in equation (1).

The equation has been given so that a reward *r* may be obtained from an action *a* at a specific state *s*.

t refers to the previous time, and *t+1* refers to the future time. max refers to finding the maximum value, and *v* is the parameter of learning rate between 0 and 1. *l* is the discount factor, and is a number between 0 and 1. The best reward *r* among actions at a specific state *s* in the whole possible state space is stored and calculated. In this process, weight is added to *r* to raise influence on reward.

However, a higher number of states experienced with the passage of time make the experience of unnecessary states based on reward unnecessary. Learning is completed by experiencing almost all states, which enables intelligent operation.

$$Q(s,a) := Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \qquad (1)$$

III. Improved Reinforcement Learning-based Dynamic Service Function Chaining Algorithm

The service function chaining algorithm proposed in this study works the same as how the algorithm investigated previously [3] does. It obtains the information of resources of the working physical nodes and the information about resources used by each service function and stores the information in a DB. Then, based on the relevant information, reinforcement learning is conducted, and a service path is set up for the usage of service node and the bandwidth of each link concerning all service chaining paths of the service node. The differentiation from the existing algorithm is that the result value of 0.5 was applied as the weighted value of *R* value that affects learning in the existing algorithm while the algorithm proposed in this study used the weighted value in Formula (2). The reason is that since the weighted value of 0.5 fixed for *R* value is the optimum value obtained from the infrastructure for experiment, it may differ depending on the system infrastructure performance or the type of service. In other words, according to the amount or performance of physical resources in the relevant node, the value of 0.5 cannot be the optimum value. Therefore, this study calculated the weighted value that could reflect the status of the infrastructure through Formula (2). In Formula (2), CPU, memory, network resource, and the remaining storage of the relevant node are divided by n, the number of VNFs working in the relevant node. In other words, if there is much resource remaining and if there is a small number of working VNFs, the weighted value of the relevant node increases, so *R* value that allows learning the relevant node will increase and the probability of the selection of the relevant node will increase.

$$CPU_{remain} = CPU_{pn} - CPU_{SF}$$
$$MEM_{remain} = MEM_{pn} - MEM_{SF}$$
$$Network_{remain} = Network_{pn} - Network_{SF}$$
$$Storage_{remain} = Storage_{pn} - Storage_{SF}$$
$$Ci = \frac{CPU_{remain} + MEM_{remain} + Network_{remain} + Storage_{remain}}{n} \qquad (2)$$

Table 1 shows the elements used in the algorithm proposed in this study.

Table 1. Definition of the elements used in the proposed algorithm

Value	Description
$G = (PN_i, L_j)$	Service topology
PN_i	Physical node with the operating SF
L_j	Set of links that linking physical nodes and SFs
SP_x	Service path that service '*x*' passes
SF_n	Service function instance *n*
PT_{SFn}	Processing time of SF_n
PAT_{SFn}	Packet arrived time of PAT from SF_n to SF_{n+1}
TBW_j	Total bandwidth of link *j*
CBW_j	Current bandwidth of link *j*
CPU_{pn}	Current cpu usage of physical node PN
MEM_{pn}	Current memory usage of physical node PN
$Network_{pn}$	Current network usage of physical node PN
$Storage_{pn}$	Current storage usage of physical

	node PN
CPU_{SF}	Current cpu usage of SF_n
MEM_{SF}	Current memory usage of SF_n
$Network_{SF}$	Current network usage of SF_n
$Storage_{SF}$	Current storage usage of SF_n
CPU_{remain}	Remaining CPU Amount
MEM_{remain}	Remaining MEM Amount
$Network_{remain}$	Remaining Network Amount
$Storage_{remain}$	Remaining Storage Amount
C_i	Weight applied to R

Algorithm 1 proceeds with learning, increasing or decreasing the R value according to the condition based on CPU, memory, and network resources of each physical node to which the service functions that can normally be performed belong and CPU, memory, network, storage resource used by the relevant service function, of the following service functions in the service function starting first when a path is chosen. It was designed to increase R value when the lowest resource was used, of the service functions that can be performed normally and when the working node was using a low resource. Through this, each node was allowed to exhibit the optimum performance by making each service function distribute the load on the working node equally, and through the selection of a service function at the lowest rate, it was possible to improve the performance through the efficient use of resources.

Algorithm 1. Proposed Algorithm

```
1: $Q \leftarrow 0$
2: $R \leftarrow$ Reward(newSP)
3: **while**
4:    Select a random state S, A
5:    $S \rightarrow SF_{src}$, $A \rightarrow SF_{dst}$
6:    **IF**(Next SF $\neq$ NULL)
7:       **IF**($PT_{SFn} > Last\ PT_{SFn}$)
8:          **while**(Number of PN)
9:             **If**($CPU_{pn} > Avg\ CPU_{SF}$ || $MEM_{pn} > Avg\ MEM_{SF}$
                  || $Network_{pn} > Avg\ Network_{SF}$ ||
                  $Storage_{pn} > Avg\ Storage_{SF}$)

10:               $r$ value is $r * C_i$

11:   Update $r$
12:   Compute Q-Learning Algorithm
13:   Update Q
14:   Update SP
```

V. Performance evaluation

The performance evaluation of the proposed algorithm was conducted using a basic scheduler provided by ODL, for three environments, including if an SFC was constructed, if a fixed weighted value 0.5 was used in the algorithm proposed in a preceding study, and if the weighted value obtained through the calculating formula of weighted value proposed in this study was applied. Three physical nodes were used, three service function instances were generated in each node, and each service function was connected with a virtual network. The items of comparison included the performance evaluation of SFC according to the number of service function instances, load by each node, and the number of service flows generated.

The performance evaluation of SFC according to the number of instances was conducted, changing the performance of each node. The performance of each node was divided into three, including low-specification, medium-specification, and high-specification, and the performance evaluation of the algorithm was conducted, according to the relevant performance. For the performance evaluation of SFC, the performance was compared with the arrival time of a file when it was sent. Fig. 1 compares the performance evaluation of SFC according to the number of the service function instances in a low-specification environment. First, if SFC is generated, using the scheduler provided by ODL, service flows are generated according to the increase of the number of service function instances; however, since service flows are generated based on distance, random, and round robin scheduling algorithm not considering the situation of resources, the performance becomes lower with the increase of the number of instances.

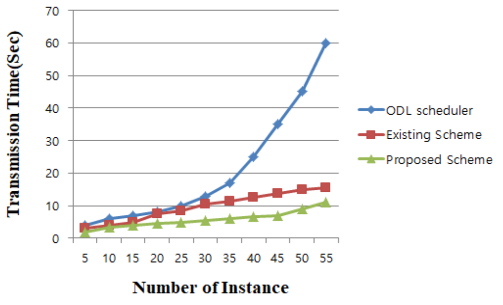

Figure 1. File transfer time according to number of instances (low performance)

Since the algorithms proposed in the preceding studies generate service flows based on the situation of the resources of each node and the situation of the service function instances working in the relevant node whenever a service flow is generated, the performance degradation according to the number of instances was less than when the basic ODL algorithm was used. However, at a specific critical point, in other words, if a resource situation is insufficient and reaches the number of instances affecting performance, the performance rapidly deteriorates. The performance of the improved algorithm proposed in this study, too, sharply decreased when reaching a certain number of instances; however, the performance degradation occurred more gradually as compared to other algorithms. In addition, for the number of service flows, too, the number of service flows generated in spite of the increase of instances in the algorithm proposed in this study because of the optimal placement of VNF through learning; however, when other algorithms were used, VNF was assigned to an overlap node with the increase in the number of instances, so it was found that if instances increased more than a specific number, the number of service flows increased. Fig. 2 and 3 are the results of experiments conducted in medium-specification and high-specification environments. The number of instances accommodated increased with the improvement of performance, so the critical point value of the performance increased in all algorithms; however, in the improved algorithm proposed in this study, it increased at the most gentle width. Fig. 4 compares the number of service flows according to the number of instances.

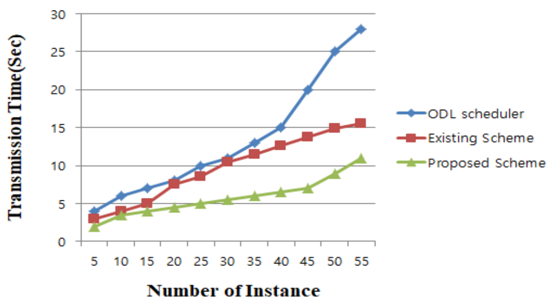

Figure 2. File transfer time according to number of instances (medium performance)

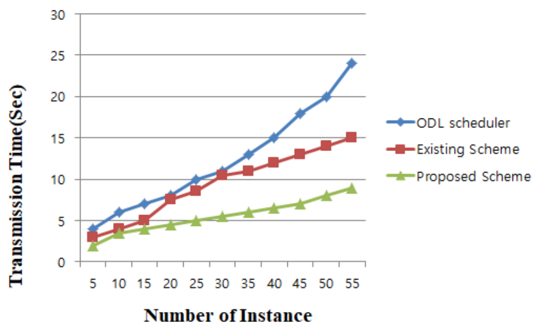

Figure 3. File transfer time according to number of instances (high performance)

Fig. 4, 5 compares load by node according to the number of instances. As for the load by node as well, the improved algorithm suggested by this paper showed uniform load and the phenomenon of overhead at a specific node was not found. In addition, the improved algorithm shows a flatter graph unlike the algorithm suggested by the existing paper. From this, it can be seen that the improved algorithm distributes SFs more equally than the algorithm suggested by the existing paper.

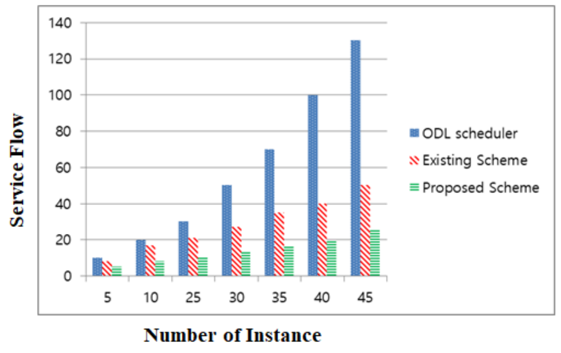

Figure 4. Number of service flows per instance

VI. Conclusion

This study investigated a plan for dynamic service function chaining that allows learning the usage of the physical resources used in each node and learns virtual resource usage used by each service function through reinforcement learning and the controller to create service flows for the condition. It was found that the method proposed through the performance evaluation provided a service with more equal and better performance through the reduction of overhead in each node than the ODL scheduler provided basically. As follow-up research, a study will be conducted on a plan for the creation of dynamic SFC in which similar services can form a chain with each other, automatically, even though the user does not directly choose service functions for SFC based on metadata.

ACKNOWLEDGMENT

This research was supported in part by FNCP Infrastructure Buildup Program funded by the Ministry of Science, ICT & Future Planning (2018-2019).

REFERENCES

[1] "ETSI", Network Functions Virtualisation White - Paper #1, 2012.
[2] ETSI ISG NFV, GSNFV-MAN 001 v0.6.2 (2014-07), "Network Function Virtualization (NFV) Management and Orchestration", 2014. 07
[3] Sang il Kim, Hwa Sung Kim," A research on dynamic service function chaining based on reinforcement learning using resource usage", ICUFN 2017, July, 2017
[4] S. Lee, S. Pack, M. Shin, and E. Paik, "Resource management for dynamic service chain adaptation, internet-draft draft-lee-nfvrg-resource-management-service-chain-00", Oct. 2014.
[5] G. Lee, I. Jang, W. Kim, S. Joo, M. Kim, S. Pack, and C. Kang, "SDN-based middlebox management framework in integrated wired and wireless networks," J. KICS, vol. 39B, no.6, pp. 379-386, Jun. 2014.