Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Modeling and Analyzing for Performance Interference Propagation in NFV

DONG ZHANG[1],XIYI PAN[2],AND WEIWEI LIN.[3]

ABSTRACT The proposal of network function virtualization(NFV) promotes network functions to run on commodity hardware in the form of software applications, which perfectly decouples the software implementation of network functions from the underlying hardware and reduces capital investment and energy consumption. With the widespread deployment of VNF applications, how to monitor and analyze NFV performance becomes a challenging issue. In present, the performance analysis method is at the VNF level, and the calculation of the delay and throughput of the user request is a simple addition after the request processing is completed. It ignores the propagation of performance interference, which will inevitably lead to incorrect performance analysis and inappropriate mapping policy. Given the above problems, this paper proposes a SFC performance inference propagation analysis algorithm. Based on the flow information of the requested service function chain and the global physical resource usage, a global SFC performance impact calculation analysis is performed on the newly arrived user request, which helping to monitor the performance of SFC indirectly in a global view and providing a reference for the mapping scheme of different mapping targets, preventing causing serious SLA violation as well.

INDEX TERMS Network Function Virtualization, Service Function Chain, Performance Interference Propagation, Performance Monitoring.

I. INTRODUCTION

IN the modern world, with the rapid progress of technology, the Internet is becoming an indispensable part of daily life. As richer feature types and higher quality of services are needed for people, it is well known that the management of the modern network is becoming increasingly difficult. Besides, The cost of taking different new service functions into the current network is growing higher. The forwarding equipment in the traditional network is tightly combined with logical control and the increasingly large network system, which makes it difficult for the developing and configuration on traditional networks. The problems above urgently need new technology to solve.

The concepts of Software Defined Networks(SDN) [1] is a new network paradigm that was recently proposed to overcome the drawbacks of the current network infrastructures. By breaking vertical integration, SDN can simplify the complexity of network management and facilitate network configuration, development, and fostering innovation, provide flexibility in network enforcement [3]. SDN gives the hope to replace traditional network and change the limitations of current network infrastructures. At the same time, motivated by the advances of virtualization, several major telecom operators embarked on an effort to combine virtualization with carrier network infrastructures and drafted the first white paper for Network Function Virtualization(NFV) [2]. By decoupling Network Functions from vendor-specific and traditional proprietary dedicated middlebox, NFV can bring lots of benefits to network carriers, such as reducing the cost of expensive network equipment [4], increasing network flexibility for fast service delivery [5], reducing Operating Expenditures(OPEX) [6] cost and improving efficiency as well as prevent middleboxes sprawl. etc. which strongly changing the infrastructure of current network. Though NFV and SDN have a different purpose, they both do indeed complement each other and enable of providing one consolidated solution for the evolution of the network [7]. The integration of NFV and SDN have been studied in many different environments (e.g., 5G, Cloud Computing, Fog Computing, Wide Area Network, etc.). Service Function Chains(SFCs) [9] is defined by the Internet Engineering Task Force (IETF) as a set of interdependent service functions(SFs) process user request

flows along with a predefined ordered list of VNFs. In the context of Network Function Virtualization(NFV) and Software Defined Network(SDN), Virtual Network Functions(VNFs) in the SFC are deployed on virtual machines and enable of interconnecting by SDN to achieve minimum cost and maximum profit for ISP.

Along with the benefits of NFV, different NFs may face numerous difficulties and challenges while moving towards network function virtualization. Meanwhile, A fundamental challenge is that the performance of processing traffic flows through virtual network function on standard high volume commodity servers may not achieve the same or a comparable performance as a proprietary delicate middlebox. Many research efforts have been devoted to improving the performance of virtual network functions hosted on standard commercial servers to improve quality of service and decrease Service Level Agreement(SLA) violations which will be illustrated in the next section. In short, SFC performance assurance issues are paramount, which is the key to replace traditional networks and promote NFV paradigm development. Even though exciting progress has been made toward high-performance provisioning in SFC, there still a lack of performance monitoring algorithm or monitoring model to discover performance issues.

Monitoring key performance is always vital in the network research area. By taking advantage of monitoring key performance metrics, a network operator can dynamically obtain real network function performance status and utilize the data that violated SLA to trigger dynamic adjustment of virtualization network function resource allocation and to improve SFC forwarding path decisions. With the advantage of monitoring key performance described above, operators can maintain acceptable application performance levels for users and provide a reference for different mapping purposes. SFC performance monitoring seems to be easy. For example, to monitor the delay, you only need to monitor the performance of each VNF, and then simply add all the VNF delays and link delays to get the delay statistics of the entire SFC. And then compare the obtained processing delay with the delay required by the user. If the processing delay is greater than the delay required by the user, the corresponding network policy of SLA violation will be triggered. However, how can we obtain possible performance violation in advance, and how to use performance inference analysis to help network operators close to different orchestration purposes? This is not a trivial matter, because of several factors lead to the problem more complex. Firstly, the processing of user request service involves multiple VNF resources along the relevant SFC forwarding path. These VNFs not only require different types of resources, but they may also require different amounts. On the other hand, in order to improve resource utilization, Network Service Providers(NSPs) usually deploy multiple VNFs on a common commodity server to achieve resource sharing, which makes each resource on the server have a different type of contenders. What's more, The dynamic change of network traffic and the mutual influence between SFC is another reason for the difficulty in locating SFC performance problems.

In this paper, we focus on proposing a performance interference analysis algorithm for monitoring SFC key performance metrics thereby benefiting the orchestration adjustment as well as reducing SLA violation. In order to be more in line with the actual employment, we considered the mutual influence of resource competition, physical equipment load, and other factors. To the best of our knowledge, we are the first to design such a related SFC performance interference algorithm for inferencing key performance metrics and decreasing SFC performance violation. The rest of the paper is organized as follows. Section II described related work about monitoring SFC performance and locating performance issues, Section III presents the conventional model including a model description and related algorithm for calculating and analyzing the SFC performance interference propagation. Section IV shows a numerical evaluation. Finally, Section V summarizes the paper and provides some directions for future research.

II. RELATED WORK

Lots of research efforts have been devoted to improving the performance of virtual network functions hosted on standard commercial servers to improve the quality of service and decrease SLA violations. For example, single root I/O virtualization (SR-IOV) [10] , Intel data-plane development kit (DPDK) [11] and Open DataPlane [12] are proposed for virtual interface of data transfer with high-performance. NetVM built based on DPDK high-throughout packet processing [13], ClickOS [14] has been developed based on Xen hypervisor and Click modular [15] router software, which is a high-performance NFV platform mainly focuses on enhancing the performance of individual VNFs. Both Service Function Chain Resource Allocation(SFCRA) algorithms and VNF Placement(VNFP) algorithms take performance issues into consideration at first [16]–[18]. In [19], the author seeks to find an orchestration scheme that can guarantee performance for every service function chain with flexible demands in underlying physical networks.

Many prior studies have investigated SFC performance detection problems. P Naik et al. [19] proposed a performance monitoring and bottleneck detection tool for NFV. They compute per-hop throughputs and delays by sniffing the packets on all VM-to VM communication paths and use "black box" measurements alone to identify performance bottlenecks in real-time. In terms of practicality, the tool for NFV performance detection

is very commendable. However, NFVPerf needs to deploy dedicated agents for every virtual machine and to collect plenty of flow statistics for calculating the delays and throughput, which consumed a large number of expensive physical resources such as bandwidth and CPU. This mechanism not enable of warning ahead of time for orchestration adjustment and optimizing the performance of SFC.

Zeng C et al. [20] investigate the performance interference among different types of co-located VNFs and analyze how VNF's competitive hardware resources and the characteristic of packet affect the performance interference. According to the measurement results, a more effective VNF placement method is given, which has significant practical significance for the research and placement of virtual network functions. Beye F et al. [21] use a machine learning technique that is fed using data generated from automatized offline performance measurements to enable fast and accurate performance prediction for VNFs. Their prediction approach has a flaw that massive flow statics must be collected for different NFV frameworks to train models and once there is an adjustment for network functions or physical equipment in data-plane, operators have to retrain their related machine learning model which cost too much extra capital expenditure and time. J Nam et al. [22] develop a system named Probius that automatically analyzes VNFs in various service function chains and inferences possible reasons for outperformance uncertainties.

III. NETWORK AND PROBLEM MODEL

In this section, we formalize the problem of inferring SFC performance issues in advance and design a dynamic deduce algorithm for possible SFC performance.

A. NETWORK MODEL

We modeled the substrate network with an undirected graph $G_s = (N_s, E_s)$ where N_s is the set of substrate nodes and E_s refer to the set of substrate links. Each substrate node n_u is capable of hosting some VNF nodes. C_u denoted the computing capcity((i.e., CPU)) of substrate node n_u , which indicates the available computing resources for hosting VNFs and define M_u as the available memory resources. For each substrate link $E_{u,w} \in E_S$, $(u, w \in N_S)$, $B_{u,w} \in Z^+$ represent the available bandwidth of $E_{u,w}$.

Supposed that there are φ NFV service requests in network, denoted them as a set $Req = \{\varsigma^1, \varsigma^2, ..., \varsigma^\varphi\}$. The κth NFV service request is represented as 5-tuple $\varsigma_\kappa = (Nv^\kappa, Ev^\kappa, Rcpu^\kappa, Rmem^\kappa, Rbd^\kappa)$, where $Nv^\kappa = (vn_1^\kappa, vn_2^\kappa, ..., vn_t^\kappa)$ is the ordered set of required VNF nodes and $Ev^\kappa = \left(e_1^\kappa, e_2^\kappa, ..., e_{t-1}^\kappa\right)$ is the set of virtual links that connect VNF nodes. And the ith e_i^κ will connect the nodes vn_i^κ and vn_{i+1}^κ. $Rcpu^\kappa = (c_1^\kappa, c_2^\kappa, ..., c_t^\kappa)$ is the set of required computing resource,where c_i^κ represents the computing resource required for node vn_i^κ.Similar to the set of $Rcpu^\kappa$,$Rmem^\kappa = (m_1^\kappa, m_2^\kappa, ..., m_t^\kappa)$ is the set of required memory resource for each node vn_i^κ,respectively.Lastly, $Rbd^\kappa = \left(bd_{i,j}^\kappa | \langle i, j\rangle \in Ev^\kappa, vn_i^\kappa, vn_j^\kappa \in Nv^\kappa\right)$ represents the set of bandwidth requirement,where $bd_{i,j}^\kappa$ is the bandwidth required for link $\langle i, j\rangle$ of the κth request.

In our investigation of virtual network function performance inference, the total delay is used as the performance metrics of the possible performance issue of SFC, which mainly consists of processing delay and transmission delay. Actually, The network delay consists of sending delay and transmission delay, the sending delay is mainly brought by sending a data frame which mainly determined by the sending capacity of physical nodes and length of a data frame. The transmission delay is determined by packet size and transmission capacity that computed with $D_i \div b_i$ in which D_i is the size of the packet and b_i is the available transmission rate. In more detail, the overall network service delay also needs to include the queue delay. The queue delay is mainly determined by the output interface link load and adjustment strategy, queue model (such as the classic M/M/1 model [23]) as well as the size of the buffer area. There are plenty of studies on the queue of the packet which will be outside the scope of this article. For simplicity, we will consider the main part of the processing delay of the physical node and transmission delay between nodes in our problem model, which are denoted as T_{ideal_pro} and T_{ideal_tran} ,respectively.

B. PROBLEM FORMULATION

We start with the network architecture described in the previous section, namely, a substrate network consists of a set of physical nodes(e.g. servers or switches, etc.), each of which can hold one or more virtual machines to run virtual network functions. In the interest of simulation of real network application scenarios, we assume that the physical server environment is heterogeneous. Typically, Heterogeneous environments where physical machines may have different resource capacities, e.g., CPU, memory, etc, enable flexible deployment of VMs to improve resource utilization. From the above, we defined SFC as an ordered set of interdependent SFs process user request flows while SF will realize through virtualization technology that we abstract it as virtual network function(VNF). Generally, each virtual machine holds one VNF application on physical machines. Due to workload changes, resources used by VMs, (memory, disk, computing capability, etc.) will vary, possibly leading to SLA violations. In the interest of simplification of presentation, we only take the delay performance into our investigation while taking other performance metrics as future work. Specifically, we supposed that the network provides services for the

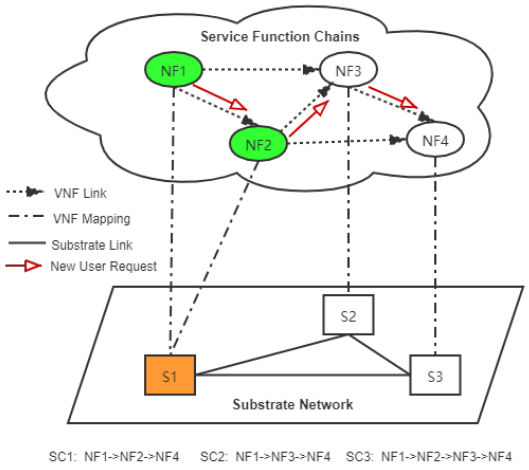

FIGURE 1. Examples of performance impact due to resource competition.

TABLE 1. Ideal Processing Time

Time / SC / NF	SC1	SC2	SC3
NF1	t	2t	4t
NF2	3t	-	t
NF3	-	2t	t
NF4	2t	2t	2t

TABLE 2. CPU Resource Allocation of SCs on S1 and Impacts Caculation

SC	CPU of S1	ReAllocate	T_{real} on S1	impacts
SC1	18	12	9	0.083
SC2	8	8	8	0
SC3	12	24	30	0.125

set of NFV service request Req. At the same time, a new user request ς^i arrives which consumes a certain amount of resources along with the corresponding service function chain and may form competition with the original service request. The objective of our research is to design algorithms that will be able to analyze the influence level that processing new user request ς^i for each original existing service function chain, which provides a reference for orchestration adjustment and reducing the probability of SLA violations.

In order to understand the problem more vividly, we will combine Figure 1 and Table 1 with an easy example to present the problem.

As illustrated in Figure 1, there are four virtual network functions, denoted as NF1, NF2, NF3, NF4, respectively. Between different VNF nodes, the black dotted line arrows indicate the logical path that current VNF nodes forward packets to the next VNF node for further processing. Besides, S1, S2, S3 are three physical servers, each of which contains a certain amount of resources, which can be used by the virtual machine to host virtual network functions. Similarly, the black solid line represents the physical link used to connect the physical servers and the dash-dotted line represents the VNF mapping to the corresponding physical nodes. Assumed that there are three SFCs in Figure 1, namely, NF1->NF2->NF4, NF1->NF3->NF4, NF1->NF2->NF3->NF4, which are called Service Chain1 (SC1), Service Chain2 (SC2), Service Chain3 (SC3). Moreover, We will use the shortest path first(SFP) algorithm to map the links between virtual network function nodes to actual physical links such that the virtual link NF2->NF3 will map the physical link $\langle S2, S1\rangle$ instead of the physical link $\langle S2, S3, S1\rangle$.

Table 1 shows the ideal time of network function required by each network function for processing each unit packet and we assumed that the minimum time slice is t. Presumed that the computing resource of S1 is 45 and Service Chain1, Service Chain2, Service Chain3 are processing 6, 2, 2 unit packets, respectively. Also, We assumed that Service Chain1, Service Chain2, and Service Chain3 require NF1 to provide 3, 4, 6 units of computing resources, respectively. Supposed that there is a new user request which required Service Chain 3 providing service with 3 unit packets to process. In the interest of simplification of presentation, we do not consider server load influencing factors that will be discussed in detail in the next section and only consider computing resources which influences the processing delay. Similarly, other resources(e.g. memory, Network I/O bandwidth, etc.) that influence the processing delay will consider the same as a computing resource. Of course, to some extent, the analysis of transmission delay affected by bandwidth resources is the same as processing delay. Table 2 illustrates the details of cpu resource allocation on S1 and the impacts of different SFCs. At first, We will caculate the current CPU resources used by each SFC, they are $6 \times 3 = 18$, $2 \times 4 = 8$, $2 \times 6 = 12$, respectively. Then, Due to the new user request, the cpu resource of S1 is insufficient, which lead to reallocate the cpu resource of S1. Unit data packets of different service function chains will consume physical resources on a fair basis. Therefore, SC1, SC2, and SC3 can process data packets of 4, 2, and 4 units, respectively. Thus, The computing resources of S1 allocated by Service Chain1, Service Chain2, and Service Chain3 on NF1 are $4 \times 3 = 12$, $2 \times 4 = 8$, and $4 \times 6 = 24$, respectively. While the ideal processing time of SC1, SC2, SC3 on S1 are $6 \times 1 = 6$, $2 \times 2 = 4$, $5 \times 4 = 20$, respectively, and the real processing time of SC1, SC2, SC3 on S1 are $6 \times 1 \times (12/8) = 9$,

$2 \times 2 \times (8/8) = 4$, $5 \times 4 \times (30/24) = 25$, respectively. According to the weight ratio, the impacts of Service Chain1, Service Chain2, and Service Chain3 on NF1 are $(9-6)/(6 \times 6) = 0.083$, $(8-8)/(2 \times 6) = 0$, and $(25-20)/(5 \times 8) = 0.125$, respectively. Next, the link delay impact and the processing delay impact of other nodes are calculated similarly to obtain the overall delay impact of each SFC. The specific calculation method will be described in the problem model.

C. PROBLEM MODEL

In this section, we will describe the overall delay calculation method in detail while analyzing the impact of physical server load on the delay, and discuss the performance impact of resource competition between service functions. Supposed that there are m Service Chains in a network, denoted them as a set $SC = \{sfc_1, sfc_2, ..., sfc_m\}$ and define $VF = (vf_1, vf_2, ..., vf_n)$ as the set of n VNF nodes. For each service request ς^{κ} ,$Nv^{\kappa} \subseteq VF, Nv^{\kappa} \equiv sfc_i, \exists \kappa \in |Req|, \exists i \in |SC|$.

First of all, the objective is to find out the SFC that affected most by the new user request, which can be described as Eq.(1). Among them,T_{real_i} is the inferred real delay of the ith SFC while T_{ideal_i} is the ideal delay of the ith SFC which will be predefined by an operator according to user requirement as ideal processing delay in Table 1. To calculate T_{real_i}, we consider the processing delay of VNF node denoted as $Tpro_i$ and the transition time in the link presented as $Tran_i$ which are described as Eqs. (2),(3) and (4). $R^{ij}_{bd_{j,j+1}}$ represents the required bandwidth resource from the jth VNF to the jth SFC while $AR^{ij}_{bd_{j,j+1}}$ described that the actually allocated bandwidth for $R^{ij}_{bd_{j,j+1}}$. The $\iota_{u,v}$ is a binary variable that equals 1 if the virtual link $\langle vf_j, vf_{j+1} \rangle$ map to physical link $\langle u, v \rangle$ and 0 otherwise. R^{ij}_{cpu} and R^{ij}_{mem} are required computing resource and required memory resource of the jth VNF of sfc_i ,respectively. AR^{ijk}_{cpu} and AR^{ijk}_{mem} are the actual allocated computing resource and memory resource for the jth VNF of sfc_i,respectively.Eqs. (5),(6) and (7) are the calculation formulation for R^{ij}_{cpu}, R^{ij}_{mem} and $R^{ij}_{bd_{j,j+1}}$. If sfc_i is a service provision of ς^{κ}, we have γ^{κ}_i as 1 and 0 otherwise. In Eqs. (9) and (10) are the calculation formulation for total processing delay and total link delay of ς^{κ}, respectively, while $Tpro_{ij}$ and $Tran^{j,j+1}_i$ denoted the processing time for VNF node vf_j and the total transition delay for the link $\langle vf_j, vf_{j+1} \rangle$ of sfc_i.Eqs. (12) , (13) and (14) are the constraints for computing resources, memory resources, and bandwidth resources. If the jth VNF of sfc_i have mapped to n_k, we have τ^{κ}_{ij} as 1 and 0 otherwise.

$$E_{sfc_i} = \frac{(T_{real_i} - T_{ideal_i})}{T_{ideal_i}}, i \in [1, |SC|]. \quad (1)$$

$$T_{real_i} = Tran_i + Tpro_i, i \in [1, |SC|]. \quad (2)$$

$$Tran^{j,j+1}_i = \frac{R^{ij}_{bd_{j,j+1}}}{AR^{ij}_{bd_{j,j+1}}} \times T^{ij}_{ideal\_tran}, \quad v, u \in [1, |E_S|], j, j+1 \in [1, |sfc_i|], i \in [1, |SC|]. \quad (3)$$

$$Tpro_{ij} = \frac{1}{2} \times Max\left(\frac{R^{ij}_{cpu}}{AR^{ijk}_{cpu}}, \frac{R^{ij}_{mem}}{AR^{ijk}_{mem}}\right) \times \delta_z \times T^{ij}_{ideal\_pro} + \frac{1}{2} \times \theta_k \times T^{ij}_{ideal\_pro}, \quad k \in [1, |N_S|], j \in [1, |sfc_i|], i \in [1, |SC|]. \quad (4)$$

$$R^{ij}_{cpu} = \sum_{\kappa=1}^{m} c^{\kappa}_j \gamma^{\kappa}_i, i \in [1, |SC|], \quad j \in [1, |sfc_i|], \kappa \in [1, |Req|] \quad (5)$$

$$R^{ij}_{mem} = \sum_{\kappa=1}^{m} m^{\kappa}_j \gamma^{\kappa}_i, i \in [1, |SC|], \quad j \in [1, |sfc_i|], \kappa \in [1, |Req|] \quad (6)$$

$$R^{ij}_{bd} = \sum_{\kappa=1}^{m} bd^{\kappa}_{u,u+1} \gamma^{\kappa}_i, u, u+1 \in Nv^{\kappa}, \quad i \in [1, |SC|], j \in [1, |sfc_i|], \kappa \in [1, |Req|] \quad (7)$$

$$\theta_k = \frac{L_k - n_k + \frac{\sqrt{(L_k - n_k)^2 + \epsilon}}{1 - L_k}}{L_k - L_n}, k \in [1, |N_S|] \quad (8)$$

$$Tpro_i = \sum_{j=1}^{N} Tpro_{ij}, i \in [1, |SC|], N = |sfc_i| \quad (9)$$

$$Tran_i = \sum_{j=1}^{N-1} Tran^{j,j+1}_i, i \in [1, |SC|], N = |sfc_i| \quad (10)$$

$$L_k = Max(\frac{u^k_{mem}}{M_k}, \frac{u^k_{cpu}}{C_k}), k \in [1, |N_S|] \quad (11)$$

$$\sum_{i=1}^{m} \tau^k_{ij} c^k_{ij} \leq C_k, j \in [1, |sfc_i|], k \in [1, N_S] \quad (12)$$

$$\sum_{i=1}^{m} \tau^k_{ij} m^k_{ij} \leq M_k, j \in [1, |sfc_i|], k \in [1, N_S] \quad (13)$$

$$\sum_{i=1}^{m} \tau^k_{i,j} \tau^k_{i,j+1} bd^k_{j,j+1} \leq B_{j,j+1}, \quad j \in [1, |sfc_i| - 1], k \in [1, N_S] \quad (14)$$

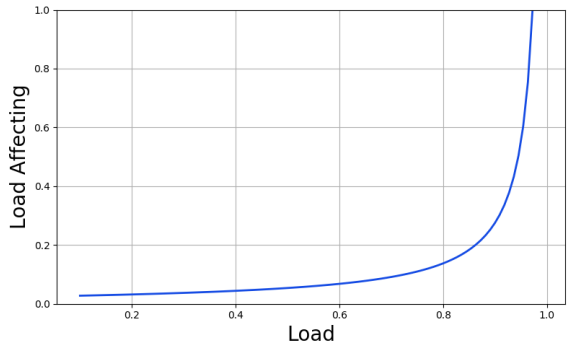

FIGURE 2. Load Effecting VS Applied Load on a system dimension.

Few research efforts consider the impact of the load on the physical server. In Eqs. (4), we introduce the θ_k denoted load impact of the kth server while in Eqs. (8) we described the calculation of the load impact. The Eqs. (8) approximates the non-linear behavior of the ideal processing time impact as it related to the maximum resource load impact on the kth physical machine. L_k denoted the maximum resource utilization impact. And the n_k is the knee of the system beyond which the impact will rise exponentially and approaches infinity gradually. The variable $\epsilon > 0$ is the harmonic parameter used to adjust the initial graph. In [24] authors have used a similar function to describe the relationship between customer utility associated with a given allocation of resources and the response time while the function is a linear growth before the inflection point. To model application scenarios in real systems as depicted in Figure 2, we propose Eqs. (1) which asymptotic increase as utilization moves close to 1.

D. ALGORITHM DESIGN

For each new user service request, We design the following algorithm to calculate the influence level of each SFC.

Algorithm 1 SFC Performance Impact Propagation Analysis Algorithm

Input: $G_s = \{N_s, E_s\}$,ς^κ
Output: $E = \{E_{sfc_1}, E_{sfc_2}, ..., E_{sfc_n}\}$

for each vn_i^κ,$bd_{i,j}^\kappa$ in ς^κ **do**
  allocate resource for vn_i^κ and $bd_{i,j}^\kappa$
  caculate R_{cpu}^{ij},R_{mem}^{ij} and $R_{bd_{j,j+1}}^{ij}$ after mapping ς^κ
end for
for each n_k in N_s **do**
  caculate θ_k for server n_k
  if $C_k < Threshold_k^{cpu}$ **then**
    for each vf_j in n_k **do**
      if vf_j in n_k **then**
        get competition impact factor δ_z
        caculate $Tpro_{ij}$ with θ_k and δ_z
      end if
    end for
  else
    $Tran_i^j \leftarrow T_{ideal\_pro}^{ij}$
  end if
end for
for each E_u^w in E_s **do**
  if $B_{u,w} < 0$ **then**
    reallocate link resource with fair competition princciple
    for each $\langle vf_j, vf_{j+1}\rangle$ in E_u^w **do**
      if $\langle vf_j, vf_{j+1}\rangle$ in sfc_i **then**
        caculate $Tran_i^{j,j+1}$
      end if
    end for
  else
    $Tran_i^{j,j+1} \leftarrow T_{ideal\_tran}^{ij}$
  end if
end for
for each E_{sfc_i} in SC **do**
  $E_{sfc_i} \leftarrow (Tran_i + Tpro_i)\,/T_{ideal_i}$
  $E \leftarrow E_{sfc_i}$
end for
return E

IV. EXPERIMENTS AND RESULTS

In this section, we will describe the details of the experiment and give some numerical results. Our experiments use C++ language to simulate the VNF mapping problem based on different resource allocation algorithms and verify the feasibility of our SFC effect calculation model. We based on the Iterative Greedy Algorithm(IGA), Simulated Annealing Algorithm(SAA), and Genetic Algorithm(GA) to allocate resources for the different user requests. Then, we compute different SFC affecting levels according to different mapping algorithms with different sizes of the physical network.

Figure 3 illustrates the detail of the average SFC impact for every SFC mapping based on IGA, SAA, and GA in 10, 20, 30, 40 substrate physical network nodes scale. For each fixed number of physical nodes, we will randomly generate a proper number of corresponding links to connect substrate nodes composing the substrate network. After that, We randomly generate 500 sets of service test requests, each set of requests has a certain probability to cause SFC impact propagation. At the same time, we conducted multiple sets of experiments and compute the average SFC affecting level to reduce the impact of randomness.

According to the experimental results, we can infer that when the scale of the network is not very large, IGA is more suitable for resource mapping which can lead to smaller SFC affecting impact. The reason may be that

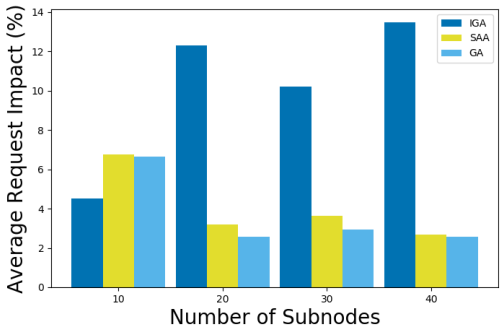

FIGURE 3. Average SFC Impact Effect Level.

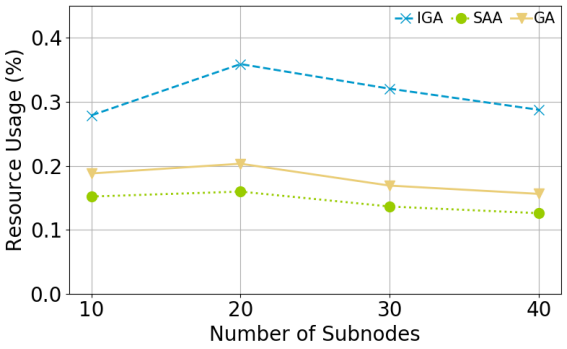

FIGURE 4. Detail of Resource Utilization.

the SAA and GA tend to reduce the use of resources and reserve more resources for the subsequent user request. When there are few physical resources, it tends to be a compact deployment. However, as the critical link resources are quickly used up and no alternative nodes and paths can be found, the service request can only be deployed on the link with insufficient resources, resulting in the SFC affecting level risen sharply. While the IGA chooses the strategy of maximizing available resources for deployment, which avoids the above-mentioned situation to a certain extent, so the average influence of SFC is much lower than the GA and SAA.

With the expansion of physical topology, we find that the average level of SFC influence decreases rapidly based on the SAA and GA resource allocation, while the greedy algorithm decreases slowly. When the physical node is 20, the influence of SFC caused by the SAA and GA is much smaller than the result of GA mapping, which is in line with the SAA and GA mapping strategy.

Figure 4 illustrates the average resource usage for every set of user requests. From the figure, we can see that the resources used by the IGA are always greater than other algorithms. And the mapping based on GA is more stable than SAA, which is also the characteristics of the algorithms themselves. Combining Figure 3 and Figure 4, SAA seems to be more suitable for large-scale network topology deployment which can cost fewer resources and lead to lower SFC affecting level relatively. If you pay more attention to stability, GA will more proper instead of SAA or IGA. Last but not least, Figure 5 describes the average rate of success mapping, from which we can see the IGA algorithm need more resource but didn't get a better mapping result. In contrast, SAA and GA usually can guarantee a better mapping success rate and lead to lower performance mutual impact, especially when a physical node is greater than 20.

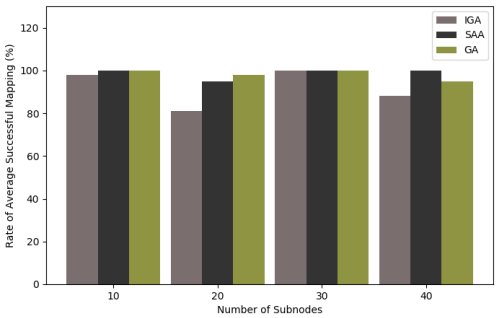

FIGURE 5. Average Rate of Success Mapping.

V. CONCLUSION

With the improvement of virtualization technology, the NFV service is provisioned in the form of a service function chain. At the same time, multiple SFC crossing mapping on the same physical node to improve resource utilization makes mutual SFC performance interference more complex. Besides, as the load of the physical node gradually becomes high, which is always ignored when considering the SFC mapping problem, the performance interference becomes more intense and negligible. In this paper, we have presented a novel concept named SFC performance inference which is different from the traditional performance monitoring analysis method. Besides, we have studied the factors that affecting SFC performance and proposed a performance inference algorithm that considers performance interference propagation. Based on the proposed algorithm, we did a comparative experiment with IGA, SAA, and GA. By analyzing the specific reason for numerical results which follow the characteristics of various mapping algorithms and verify the correctness of our algorithm. We believe that our algorithm provides a reference for orchestration adjustment and optimizing the performance of SFC.

REFERENCES

[1] Feamster N, Rexford J, Zegura E. The road to SDN: an intellectual history of programmable networks[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(2): 87-98.
[2] http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[3] Kreutz D, Ramos F M V, Verissimo P E, et al. Software-defined networking: A comprehensive survey[J]. Proceedings of the IEEE, 2014, 103(1): 14-76.

[4] Xie Y , Liu Z , Wang S , et al. Service Function Chaining Resource Allocation: A Survey[J]. 2016.

[5] Batista D M, Blair G, Kon F, et al. Perspectives on software-defined networks: interviews with five leading scientists from the networking community[J]. Journal of Internet Services and Applications, 2015, 6(1): 22.

[6] John W, Pentikousis K, Agapiou G, et al. Research directions in network service chaining[C]//2013 IEEE SDN for future networks and services (SDN4FNS). IEEE, 2013: 1-7.

[7] Mijumbi R, Serrat J, Gorricho J L, et al. Network function virtualization: State-of-the-art and research challenges[J]. IEEE Communications surveys & tutorials, 2015, 18(1): 236-262.

[8] Bonfim M S, Dias K L, Fernandes S F L. Integrated NFV/SDN architectures: A systematic literature review[J]. ACM Computing Surveys (CSUR), 2019, 51(6): 1-39.

[9] Quinn P, Nadeau T. Problem statement for service function chaining[C]//RFC 7498. RFC Editor, 2015.

[10] SR-IOV. PCI Special Interest Group,http://www.pcisig.com/home

[11] Intel, "Guide: Data plane development kit for linux," Guide, April 2015.

[12] Linaro Networking Group, "Opendataplane introduction and overview," January 2014.

[13] Hwang J, Ramakrishnan K K, Wood T. NetVM: High performance and flexible networking using virtualization on commodity platforms[J]. IEEE Transactions on Network and Service Management, 2015, 12(1): 34-47.

[14] Martins J, Ahmed M, Raiciu C, et al. ClickOS and the art of network function virtualization[C]//11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14). 2014: 459-473.

[15] Kohler E, Morris R, Chen B, et al. The Click modular router[J]. ACM Transactions on Computer Systems (TOCS), 2000, 18(3): 263-297.

[16] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in Computer Communications (INFOCOM), 2015 IEEE Conference on. IEEE, 2015, pp. 1346–1354.

[17] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in Network and Service Management (CNSM), 2015 11th International Conference on, 9-13 Nov. 2015 2015, pp. 50–56.

[18] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for nfv chaining in packet/optical datacenters," Journal of Lightwave Technology, vol. 33, no. 8, pp. 1565–1570, 2014.

[19] Naik P, Shaw D K, Vutukuru M. NFVPerf: Online performance monitoring and bottleneck detection for NFV[C]//2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). IEEE, 2016: 154-160.

[20] Zeng C, Liu F, Chen S, et al. Demystifying the performance interference of co-located virtual network functions[C]//IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2018: 765-773.

[21] Beye F, Shinohara Y, Shimonishi H. Towards Accurate and Scalable Performance Prediction for Automated Service Design in NFV[C]//2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC). IEEE, 2019: 1-7.

[22] Nam J, Seo J, Shin S. Probius: Automated approach for vnf and service chain analysis in software-defined nfv[C]//Proceedings of the Symposium on SDN Research. 2018: 1-13.

[23] Mijumbi R, Serrat J, Gorricho J L, et al. Design and evaluation of algorithms for mapping and scheduling of virtual network functions[C]//Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft). IEEE, 2015: 1-9.

[24] Chandra A, Gong W, Shenoy P. Dynamic resource allocation for shared data centers using online measurements[C]//International Workshop on Quality of Service. Springer, Berlin, Heidelberg, 2003: 381-398.