Received January 12, 2020, accepted January 31, 2020, date of publication March 3, 2020, date of current version March 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2978002

Design and Implementation of Network-Aware VNF Migration Mechanism

BO YI[1], XINGWEI WANG[2,3], MIN HUANG[4], AND KEXIN LI[1]
[1]College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China
[2]State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110169, China
[3]College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China
[4]College of Information Science and Engineering, Northeastern University, Shenyang 110819, China

Corresponding author: Xingwei Wang (wangxw@mail.neu.edu.cn)

This work was supported by the National Key R&D Program of China No. 2019YFB1802800, the Major International (Regional) Joint Research Project of NSFC under Grant No. 71620107003, the National Natural Science Foundation of China under Grant No. 61872073, the CERNET Innovation Project of China under Grant No. NGII20190107, and the Fundamental Research Funds for the Central Universities of China under Grant No. N2016012.

ABSTRACT By separating network functionalities from the dedicated hardware and implementing them in the form of Virtual Network Function (VNF), Network Function Virtualization (NFV) provides a new service paradigm called Service Function Chain (SFC). It is unrealistic to replace the running VNFs due to the stateful information accumulated, which makes the current VNF re-deployment or replacement approaches no longer effective enough. Such a situation is even worse when VNFs are shared. Instead of replacing VNFs, this paper proposes a VNF migration approach to maintain the network balance and performance by migrating VNFs from poor-state nodes to good-state nodes. In particular, two situations are considered, which are node-aware and link-aware VNF migrations respectively. Via migrating the VNFs that are related to the nodes or links causing Quality of Service (QoS) issues, the network and service performance can be optimized. In addition, the migration impact is also minimized by perceiving the global network states. The experimental results indicate that the service performance is improved and the load balance is achieved after the VNF migration.

INDEX TERMS Network function virtualization, virtual network function, node-aware, link-aware, VNF migration, service function chain.

I. INTRODUCTION

The traditional network is fulfilled by various kinds of middleboxes which are closely coupled with the dedicated hardware. Then, it becomes very difficult to deploy new network services. Because on one hand, modifying the already exited middleboxes is almost impossible. On the other hand, deploying a set of new middleboxes to compose new services requires long time and high cost. For example, the IPv6 protocol has been proposed for many years [1], while the transition from IPv4 to IPv6 is still in progress, because modifying the IPv4 related dedicated hardware is extremely hard [2], [3]. Nowadays, users' requirements on service category and service quality both increase sharply. In this way, service providers have to periodically purchase physical infrastructure to handle the ever increasing requests and context, which directly results in high capital and operation expenditures.

Recently, the Commercial-Off-The-Shelf (COTS) hardware develops so quickly that it can now support more functions with a cheap price. In this regard, COTS hardware becomes more competitive compared with the dedicated hardware. Network Function Virtualization (NFV) [4] separates network functionalities from dedicated hardware and implements them in software in the form of Virtual Network Function (VNF), such that VNFs can be flexibly placed on the COTS hardware to achieve the fast service deployment. Hence, the overall cost is greatly reduced because the COTS hardware used to support VNFs is cheap. Besides, via introducing the centralized control of Software-Defined Networking (SDN) [5], the VNF deployment and the service provisioning can be carried out effectively and efficiently.

By assembling various VNFs in order, a new service model called Service Function Chain (SFC) [6] is constructed. Then,

The associate editor coordinating the review of this manuscript and approving it for publication was Thanh Ngoc Dinh.

how and where to deploy these VNFs become vital important for SFC provisioning. Currently, there are a lot of work proposed to address such problem. Assuming that one VNF is already deployed on a physical node which gradually becomes over-loaded during this VNF's lifecycle, if we still use the VNF deployment strategy to solve this situation, we will need to first remove it and then initiate the same kind of VNF on the other physical nodes. Although such operation could average out the computational burden on nodes, it actually ignores the important stateful information accumulated at the beginning of the VNF deployment, which naturally leads to serious issues when the new VNF is running. Instead, if we use the VNF migration strategy, we not only can average out the computational burden on nodes, but also retain the stateful information.

In addition, the QoS guarantee is also a very important aspect for SFC provisioning. The resource on each node and the bandwidth on each link in the network are actually limited. When the number of network service request increases, the actual required resource of VNFs may exceed those offered by physical nodes. In this condition, more packets would be backlogged in VNFs, which leads to many issues such as longer processing delay, link congestion, VNF crash, etc. The service quality cannot be guaranteed under these cases. In order to restore the partially overloaded network to normal level, part of VNFs that are either deployed on high-load nodes or around high-load links can be migrated to light-load ones.

The VNF migration mechanisms [7] can be not only used to relieve the network overload situations, but also applied to other situations such as failover, energy saving, etc. However, it is aware that the current state-of-the-art VNF migration researches suffer from three issues: 1)some work [11]–[13] directly apply the VM migration methods to solve the VNF migration problem and ignore the migration of the stateful information inside VNFs, so that unexpected bugs may occur; 2)some work [14]–[17] do not share VNFs among SFCs, which results in low resource utilization and high resource waste; 3)some work [18]–[24] fulfill the VNF migration in a constant environment where the traffic is unchangeable, which does not accord with the practical situations. Therefore, designing proper VNF migration mechanism is significant for practical NFV deployment. Jointly taking the three challenges into consideration, a VNF migration mechanism is proposed and the main contributions are summarized as follows:

- According to the characteristics of VNF migration and traffic dynamics, we formulate a novel network model, where the VNFs are shared among different SFCs. Besides, the service function chaining and mapping processes can be adjusted to satisfy the features of practical network to minimize the VNF migration impact on network.
- A network aware VNF migration mechanism is proposed to address the SFC performance degradation and QoS problem caused by the VNF processing rates mismatch situation. By migrating the VNFs on bottleneck nodes and links, the proposed mechanism restores the service performance to a better state and the migration impact is minimized based on the global network view.
- The system framework is implemented and simulated based on the Docker platform over two real-world topologies. The simulation results indicate that the service performance and the resource utilization can be improved after VNF migration.

The rest of this work are organized as follows. Section II presents the related work. Section III introduces the system framework, in which the network and service models are formulated. The detailed VNF migration mechanism is introduced in Section IV, while the simulation results are given in Section V. Section VI concludes this work.

II. RELATED WORK

The VNF migration problem evolves from the traditional Virtual Machine (VM) migration problem which includes two cases. The traditional VM migration methods are generally offline because they need to shutdown the VMs before migration, while the VNF migration methods are usually online [8]. Nevertheless, the migration methods can be divided into three categories which are post-copy, pre-copy and the hybrid of them [9]. For post-copy, it first sends the processor status to destination and then transfers the memory content once for all. The pre-copy is opposite because it first sends the memory content to destination and then transfers the processor status [10].

In order to solve the VNF migration problem, the ideas used in VM migration are borrowed. For instance, [11] separates the VNF migration into four parts, among which the actual migration part is addressed by a static and discrete Markov decision process. Meanwhile, [12] defines a series operations to restore the network performance and QoS indicators during the VNF migration process ahead of time. Then, a two-phase method is proposed, which leverages these operations to reduce the migration cost and optimize the forwarding rules. [13] assumes that one VM only hosts one VNF, so that the VNF migration is degraded to the VM migration and it migrates the whole VM instead of a single VNF. Although these methods can be applied to address the VNF migration problem, the unique and particular characteristics of VNFs and the internal state information transfer are actually not taken into consideration.

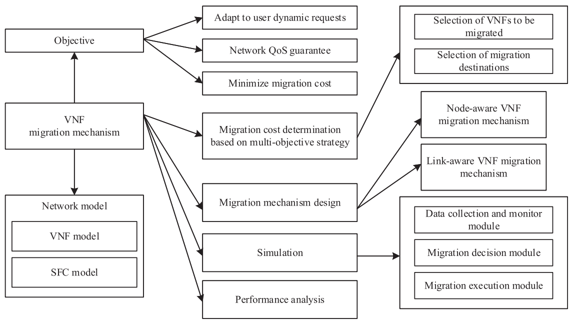
By integrating the centralized control ability of SDN, [14] proposes to carry out the VNF migration using a SDN controller. Thus, the migration cost is formulated by the controller cache consumption and the VNF internal state transferring time. Meanwhile, [15] transforms the VNF migration problem to the graph pattern matching problem, during which the delay and cost of VNF migration are not considered. [16] proposes a mapping algorithm between VNF and SFC, which can further split the physical resource according to users' requirements, and implement dynamic service deployment via virtualization technology. [17] summarizes various kinds of VNF migration algorithms, and combines the VNF migration with the VNF replication to jointly address the load balance and service disruption problems. Despite this, these work only allow to use one VNF instance by one SFC. However, some SFCs may not fully utilize the allocated resource and such idle resource also cannot be used by other services. In this case, the resource utilization will be low and some resource would be wasted.

The VNF migration exists many network scenarios. For example, [18] proposes a real-time VNF migration algorithm to address the high delay and frequently changing resource utilization issues in clouds, while [19] replaces the dedicated middleboxes by virtual machines and uses the real-time VM migration policy to deploy system configuration in the edge network environments. In particular, the VM migration can be implemented by either serial or parallel strategies, while the service downtime and overall migration time are used to judge which strategy to be used. [20] studies the relationship between the data center locations and the VNF placement strategy for the purpose of building proper traffic path between discrete sources and destinations. Furthermore, there are also many references studying the VNF migration technologies such as VM and Docker. For example, [21] elaborates the benefits and drawbacks of the current VM migration technologies, while references [22]–[24] mainly focus on using the docker technology to address the VNF migration problem.

Nevertheless, these work all assume that the network traffic is unchangeable. However, in practical network, traffic pattern or size may change when it traverses specific VNFs, which finally affects the performance evaluation. Besides, many SFCs will be affected when the migrated VNF is shared among them. In this regard, minimizing the migration impact should also be focused. This work jointly takes the above discussed factors into consideration when addressing the VNF migration problem.

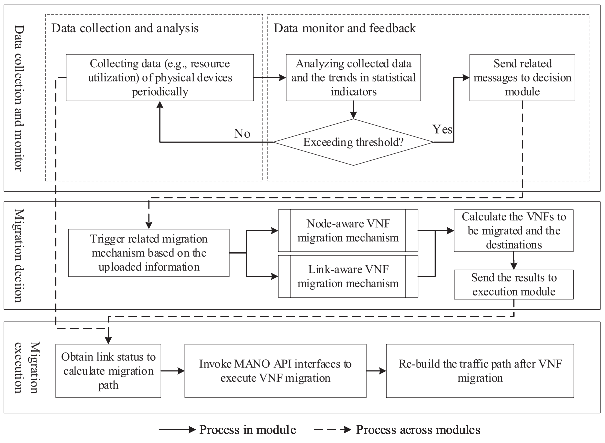
III. SYSTEM FRAMEWORK

The overall view of our system for VNF migration is shown in Fig. 1. Firstly, the VNF migration system should adapt to the dynamic changes with the minimum cost and maximum QoS guaranteed. Secondly, by designing proper network model, the VNFs and SFCs can be optimized and adjusted to satisfy the practical requirements of VNF migration, for example, flexibility and fine-granularity on management and control. Thirdly, the related constraints and migration cost can be obtained via analyzing the VNF migration problem. After that, node-aware and link-aware VNF migration mechanisms are proposed and used to decide the VNFs to be migrated and to guide the migration process. Finally, we implement three components which are data collection module, migration decision module and migration execution module, to carry out the evaluation on performance and feasibility of the proposed VNF migration mechanism.



Objective
Adapt to user dynamic requests
Network QoS guarantee
Minimize migration cost
VNF migration mechanism
Migration cost determination based on multi-objective strategy
Migration mechanism design
Simulation
Performance analysis
Network model
VNF model
SFC model
Selection of VNFs to be migrated
Selection of migration destinations
Node-aware VNF migration mechanism
Link-aware VNF migration mechanism
Data collection and monitor module
Migration decision module
Migration execution module

FIGURE 1. System framework.



Data collection and monitor
Data collection and analysis
Data monitor and feedback
Collecting data (e.g., resource utilization) of physical devices periodically
Analyzing collected data and the trends in statistical indicators
Send related messages to decision module
Exceeding threshold?
No
Yes
Migration decion
Trigger related migration mechanism based on the uploaded information
Node-aware VNF migration mechanism
Link-aware VNF migration mechanism
Calculate the VNFs to be migrated and the destinations
Send the results to execution module
Migration execution
Obtain link status to calculate migration path
Invoke MANO API interfaces to execute VNF migration
Re-build the traffic path after VNF migration
Process in module
Process across modules

FIGURE 2. Diagram of system modules' relationship.

The relationship among data collection, migration decision and migration execution modules is illustrated in Fig. 2. In particular, the data collection module is responsible for collecting and analyzing data periodically to form a positive feedback. Generally, the collecting data includes the resource utilization, QoS indicators, etc. Based on the collected data, analysis can be made to judge whether to send request to the migration decision module. According to the changing trend of the historical data, we can discover the nodes and links that affect the service performance most. Then, the information of current statistics and the resource utilization will be sent to the decision module for decision making.

The migration decision module is the core of this system, which is mainly in charge of deciding whether and where to migrate VNFs based on the data offered by the data collection module. Firstly, according to the status of the current network, we can calculate the overloaded nodes and links based on the QoS constraints, and determine those on which VNFs and traffic should be migrated. Secondly, via executing the corresponding VNF migration mechanism, we can obtain the VNFs to be migrated and the corresponding migration destinations. Then, we send the obtained results to migration execution module.

Once the migration execution module receives the message from the migration decision module, it will invoke the NFV MANO related Application Program Interfaces (APIs) to

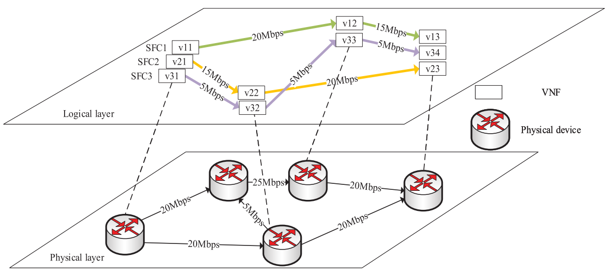


FIGURE 3. Network model.

fulfill the VNF migration process. In particular, the traffic path between the migrated VNF and its neighbor VNFs should be determined and built based on the source, destination, and the links between them.

IV. NETWORK AND SERVICE DESCRIPTION

In order to minimize the overall impact caused by VNF migration, we presents a redesigned model for service and network according to the unique VNF characteristics.

A. NETWORK

According to the demands of VNF migration, we redesign the network node and service structure, and formulate a migration friendly network model.

As indicated in Fig. 3, the formulated network model is mainly composed of the logical layer and the infrastructure layer. The infrastructure is the network foundation and consists of many connected physical devices, while the logical layer reflects specific service descriptions. Each SFC represents a kind of service which consists of many sequential VNFs. In addition, when the traffic of one SFC traverses one VNF, some properties may be changed, for example, the traffic bandwidth may be reduced.

B. SERVICE FUNCTION CHAIN

The SFC provisioning process includes not only deploying corresponding VNFs, but also steering traffic through these VNFs in order. Currently, many researches [14]–[16] have been proposed to share the same VNF among different SFCs as shown in Fig. 4(a). Such sharing pattern also exists in traditional networks, where network functions are coupled with dedicated hardware. In NFV-enabled networks, network functionalities are extracted from the dedicated hardware and used to form various VNFs to supply the corresponding services. In this way, network resource can be fully utilized by sharing VNFs among different SFCs. However, such VNF sharing model is very unfriendly to VNF migration, since the migration process will inevitably affect all the SFCs using the migrated VNF, during which the service reliability and consistency offered by those SFCs have to be considered.

Based on the virtualization technology enabled by NFV, we split the shared VNF in the way that each separated part is used by one single SFC. The corresponding model is illustrated in Fig. 4(b), where the separated parts are actually

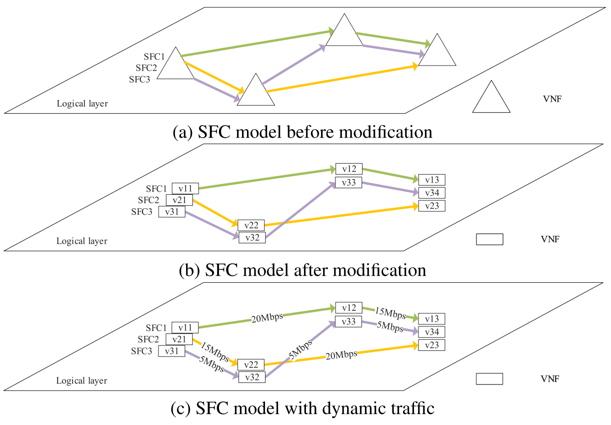

SFC1
SFC2
SFC3
Logical layer
VNF
(a) SFC model before modification
v11
v21
v31
v12
v33
v13
v34
v23
v22
v32
(b) SFC model after modification
20Mbps
15Mbps
5Mbps
(c) SFC model with dynamic traffic

FIGURE 4. Service function chain model.

fine-granularity VNFs that demand less resource than the coarse-granularity VNFs in the migration process, so that we can easily find the candidate migration destination nodes for them. More importantly, using one VNF independently only needs to consider one related SFC, because the corresponding process will not affect the other SFCs. Following this rule, we can minimize the migration impact on network by improving the stability of VNF migration and the consistency of service provisioning.

In order to simplify the problem, most current researches [12], [14], [16] assume that the amount of traffic traversing each SFC is constant, namely, the traffic will not change before and after being processed by one VNF. Nevertheless, in practical networks, the traffic change depends on the type of VNFs. For example, if it is a kind of forwarding VNF (e.g., virtual switch), the traffic will not change after passing this VNF. However, if it is a kind of filtering VNF (e.g., firewall), the traffic will be reduced after passing this VNF. Likewise, if it is a kind of caching and distributing VNF, the traffic will be increased. Therefore, we cannot simply use a constant value to define the amount of traffic.

Therefore, taking the traffic changes into consideration when designing the VNF migration mechanism accords with the practical situation. Fig. 4(c) shows a more practical SFC model, where the traffic of SFCs will change according to the VNFs they traverse. Based on this model, the relationship between traffic and VNF migration is built and the influence caused by traffic alteration on is fully considered.

C. MAPPING MODEL

The mapping relationship between VNFs and the physical devices is built for SFC provisioning. Besides, the forwarding paths between any two neighbor VNFs are also planned. However, any two adjacent VNFs may not be directly connected in the physical network environment, because the traffic between them may need to be forwarded by multiple devices. One way to handle such situation is to add one or more forwarding VNFs between the two neighbor VNFs. These forwarding VNFs are designed to forward

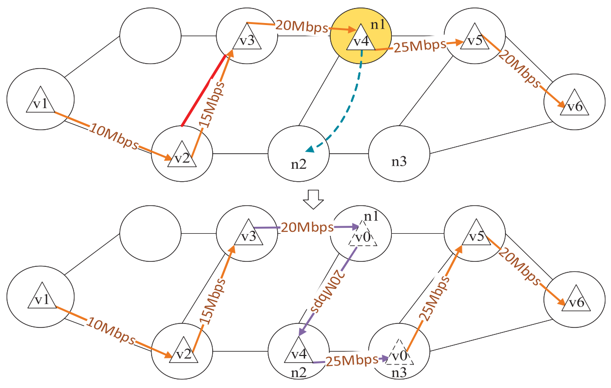


FIGURE 5. SFC mapping model.

traffic without occupying any computing resource. In the same time, the traffic will not change after passing these forwarding VNFs. Regarding this idea as the foundation, it is improved and updated as another novel service function mapping model as shown in Fig. 5.

In this model, we first map the VNFs of one SFC on adjacent nodes in order. The traffic between any two neighbor VNFs will only traverse the link between the two directly connected nodes they map on. For example, there are six VNFs mapped on six adjacent nodes in the upper side of Fig. 5. When there exists a resource shortage in node $n1$ where VNF $v4$ has been mapped on, $v4$ will be migrated to the adjacent node $n2$. When such migration process is finished, we need to rebuild the traffic paths between $v3$ and $v4$, and between $v4$ and $v5$. According to the shortest path algorithm, we need to deploy forwarding VNFs (indicated by $v0$) on $n1$ and $n3$. Since $v0$ on $n1$ and $n3$ do not require any computing resource, the overloaded situation of $n1$ can be eased.

V. VNF MIGRATION MECHANISM DESIGN

In this section, we first analyze the application scenarios of VNF migration. Then, by formulating the related constraints and migration costs, we define the objective of the VNF migration problem. Finally, the network-aware VNF migration mechanism is designed and presented respectively to address the physical node and link overloaded situations.

A. PROBLEM ANALYSIS

VNF migration mechanism is usually regarded as a mean to optimize and improve the service performance in NFV-enabled networks. For example, when physical nodes and links are overloaded, we may need to migrate VNFs on these nodes and the traffic on these links, so that their burden could be eased. Besides, VNF migration can also be applied to many situations such as failover, disaster recovery and maintenance. In order to save energy, we may need to shutdown several physical servers and migrate their services to a centralized one.

Therefore, designing efficient VNF migration mechanism is important. Instead of globally adjusting the whole SFC, we intend to adjust the local mapping relationship between VNFs and physical nodes, because the former demands high cost and long time to finish the global operation. During this process, services may suffer a long interruption, which in turn affects the reliability and consistency of service delivery. In order to focus on the VNF migration problem, we assume that all VNFs are already placed in network to offer specific functionalities. Due to the dynamic change of network traffic, some physical nodes and links may not be able to provide sufficient resource, which decreases the overall performance. In this paper, we mainly focus on addressing the VNF migration problem in node and link overloaded scenarios. By designing and introducing proper VNF migration mechanism, we can locally adjust the mapping relationship between SFC and the physical network for the purpose of easing resource shortage.

TABLE 1. Notations.

Notation	Meaning
N, E, C, V, P	the sets of physical nodes, links, SFCs, VNFs and paths
F_c, V_c	the sets of traffic and VNFs of SFC c
$f^c_{v_1,v_2}$	the traffic between v_1 and v_2 of SFC c
cap_u	the computing resource of node u
$band^l_{u_1,u_2}$	the bandwidth resource of link l between u_1 and u_2
$delay^l_{u_1,u_2}$	the delay of link l between u_1 and u_2
a_c, t_c	the maximum allowed and the actual delay of SFC c, $a_c, t_c \geq 0$
req_v, pt_v	the computing resource requirement and the processing delay of VNF v
ρ_u, ρ_l, σ^2	the load rates of node u and link l, and the variance of load rate of physical nodes
x^v_u, y^v_c, z^l_c	the binary variables indicating whether VNF v is placed on node u, whether VNF v belongs to SFC c,and whether link l is used by SFC c
$w^l_{v_1,v_2}$	the binary variables indicating whether the traffic between VNF v_1 and v_2 traverses link l

1) MIGRATION CONSTRAINTS

As indicated in Fig. 3, the proposed framework is composed of physical network layer and logical layer. The underlying network is abstracted as a graph denoted by $G(N, E)$, where N indicates the set of physical devices and E indicates the set of physical links between any two directly connected nodes. For SFCs in the logical layer, the notation C is used to denote the set of all SFCs and V is used to denoted the set of all VNFs. The general notations of this work are summarized in Table 1.

First of all, the allocated amount of computing resource should not exceed the maximum capacity of physical nodes, as follows:

$$\sum req_v \times x^v_u \leq cap_u, \quad \forall v \in V,\ u \in N \tag{1}$$

Secondly, the total amount of bandwidth resource allocated for VNF migration should not exceed the maximum capacities of physical links, as follows:

$$\sum_{c\in C}\sum_{k=1}^{|V_c|-1} w^l_{(v_k,v_{k+1})} x^{v_k}_{u_1} x^{v_{k+1}}_{u_2} f^c_{(v_k,v_{k+1})} \le b^l_{(u_1,u_2)}, \quad \forall l \in E, u_1, u_2 \in N, f^c_{(v_k,v_{k+1})} \in F_c \tag{2}$$

Thirdly, the service traffic should traverse all the required VNFs and the corresponding delay should not exceed the maximum allowed one. Since the propagation delay is usually constant, we mainly consider the processing and queueing delay, as follows:

$$\sum_{v\in V_c} y^v_c pt_v + \sum_{l\in E} z^l_c delay^l \le a_c, \quad \forall c \in C \tag{3}$$

In addition, each VNF instance can only be placed on one physical node, which is expressed as follows:

$$\sum_{v\in V_c} x^v_u = 1, \forall c \in C, \quad u \in N \tag{4}$$

Finally, the coarse-granularity VNFs are split into smaller ones, so that we have to make sure that one smaller VNF is used only by one SFC, as follows:

$$\sum y^v_c = 1, \forall c \in C, \quad v \in V \tag{5}$$

All these constraints are taken into consideration when designing the VNF migration mechanism.

2) MIGRATION COST

The migration cost is used to describe the impact that VNF migration may have on the network, which is very important to decide the migrated VNFs and the destinations. In this paper, we intend to quantify the migration cost by integrating the resource load rate, the carried traffic of migrated VNF and the hops of VNF migration. Firstly, the resource load rate caused by the VNF migration is denoted by wg^r. Assuming that we need to migrate the VNF v_i from the node u_1 to u_2, then, the resource load rate of u_2 after the VNF migration can be expressed as follows:

$$wg^r = \frac{req_{v_i} + \sum_v x^v_{u_2} req_v}{cap_{u_2}}, \tag{6}$$

where req_{v_i} indicates the required resource of v_i, while $\sum_v x^v_{u_2} req_v$ indicates the already allocated resource. cap_{u_2} means the maximum capacity of u_2.

To calculate the carried traffic of VNFs, we need to take all the input and output traffic into consideration. However, it is aware that the traffic pattern of the first VNF is different from the other VNFs. Using the notation f^c_i to indicate the output traffic of VNF v_i that belongs to SFC c, the normalization process is operated as follows:

$$wg^f = \begin{cases} \dfrac{f^c_i}{\max f}, & i = 1; \\ \dfrac{f^c_{i-1} + f^c_i}{\max f}, & i \ge 2. \end{cases} \tag{7}$$

The migration hop is also a factor affecting the migration cost. In this regard, $hop(u_1, u_2)$ is used to describe the minimum hops between nodes u_1 and u_2. Now, jointly taking the traffic load, required computing resource and migration hops into consideration, we formulate the migration cost as follows:

$$cost^{v_i}_{u_1,u_2} = hop(u_1, u_2) \times (wg^f + wg^r) \tag{8}$$

B. VNF MIGRATION MECHANISMS

The VNF migration are usually triggered by two kinds of situations which are node and link overload. Such two cases have different characteristics and shall be solved differently.

1) NODE-AWARE VNF MIGRATION

Aiming at the physical node overload situation, we propose a Node Aware VNF Migration Mechanism (NAVMM). The trigger condition of NAVMM relies on monitoring VNFs periodically. If there are physical nodes that cannot satisfy the computing resource requirements of all VNFs on it, NAVMM is invoked to ease the burden of the overloaded nodes.

Given an overloaded node u, the breadth-first strategy is used by NAVMM to search all the candidate nodes for u. After the searching, all nodes satisfying the migration requirement are obtained. However, there maybe many VNFs on u, and we should decide which one to be migrated. In this regard, the migration cost defined in equation (8) is used and the VNF with the smallest migration cost will be migrated. Repeating the above process until no overloaded nodes can be found.

The flow chart of NAVMM is shown in Fig. 6. the key objective of NAVMM is to minimize the impact on network caused by VNF migration. By locally adjusting the mapping relationship between VNFs and physical nodes, we can minimize the migration cost when selecting the VNFs to be migrated and executing the VNF migration. Specifically, the more VNFs to be migrated, the larger cost will be. Therefore, the VNFs to be migrated should be carefully selected. Two criteria are used to select the migrated VNFs. First of all, the VNF migration should cause as less cost as possible. Secondly, the number of VNFs to be migrated should be as fewer as possible. Jointly taking the two criteria into consideration, we may only need to migrate one VNF towards minimizing the impact caused by VNF migration with as less cost as possible.

As explained, the proposed mechanism operates based on the data collected. Thus, by analyzing these data, we first construct the SFC list (denoted by *sfc_list* which includes the demands on computing resource, bandwidth, etc.), the mapping table between VNFs and physical nodes (denoted by *vnf_map*), the SFC flow path (denoted by *sfc_flow_path*), and the physical nodes and links exceeding the predefined thresholds (denoted by *exceed_res_nodes* and *exceed_band_links* respectively). All the nodes in the set *exceed_res_nodes* suffer from the overloaded situation and need to be optimized. Selecting one node from *exceed_res_nodes*, let's say node u, then, we search all the

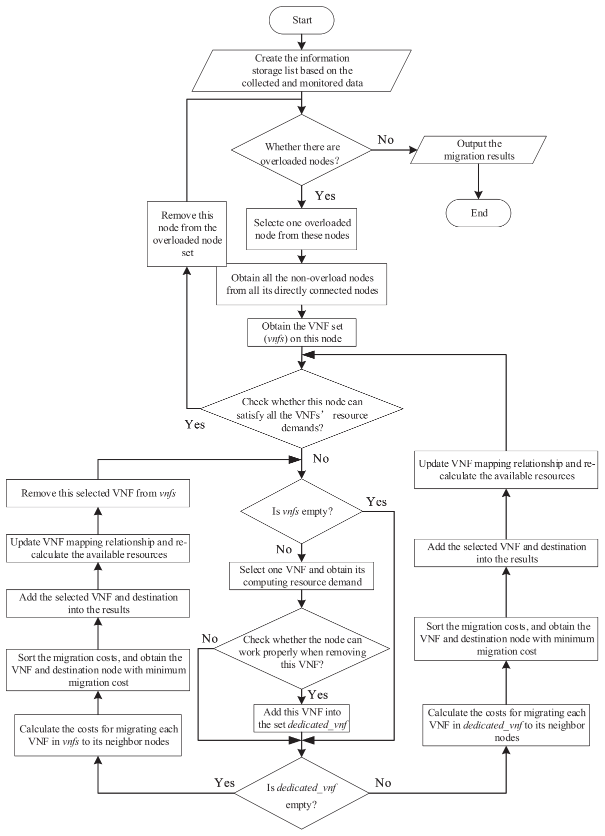

Start
Create the information storage list based on the collected and monitored data
Whether there are overloaded nodes?
No
Output the migration results
End
Yes
Selecte one overloaded node from these nodes
Remove this node from the overloaded node set
Obtain all the non-overload nodes from all its directly connected nodes
Obtain the VNF set (*vnfs*) on this node
Check whether this node can satisfy all the VNFs' resource demands?
Yes
No
Remove this selected VNF from *vnfs*
Update VNF mapping relationship and re-calculate the available resources
Is *vnfs* empty?
Yes
No
Select one VNF and obtain its computing resource demand
Add the selected VNF and destination into the results
Check whether the node can work properly when removing this VNF?
No
Yes
Sort the migration costs, and obtain the VNF and destination node with minimum migration cost
Add this VNF into the set *dedicated_vnf*
Calculate the costs for migrating each VNF in *vnfs* to its neighbor nodes
Calculate the costs for migrating each VNF in *dedicated_vnf* to its neighbor nodes
Is *dedicated_vnf* empty?
Yes
No

FIGURE 6. Flow chart of Node-aware VNF migration mechanism.

neighbor nodes of u and those satisfying the migrated VNF resource demands will be stored in the set *neighbour_list*. Besides, all the VNFs on u are stored in the set *vnfs*. Now, we compare the total required amount of computing resource by VNFs in *vnfs* with the maximum capacity offered by u using the following condition:

$$cap_u < \sum_{v \in vnfs} req_v \tag{9}$$

If (9) is satisfied, then u is not in a normal state and we should migrate the VNFs on it to other nodes. However, considering the fact that there are many VNFs in *vnfs*, we need to determine which one to be migrated. In this regard, we first define a set notation *dedicated_vnf*. Then, given a selected VNF v, if

$$cap_u > \sum_{v' \in (vnfs - v)} req_{v'}, \quad \forall v \in vnfs \tag{10}$$

is satisfied, storing v in *dedicated_vnf*.

The VNFs in *dedicated_vnf* are primary choices to be migrated. If *dedicated_vnf* is empty, we should select the migrated VNFs from *vnf*. Otherwise, the migrated VNF is determined from *dedicated_vnf*. Nevertheless, the searching strategy is the same. Assuming that $dedicated\_vnf \neq \emptyset$, we can first calculate a migration cost matrix (denoted by *cost*) based on equation (8), where $cost_{i,j}$ indicates the cost of migrating the i-th VNF to the j-th neighbor node.

Then, searching through *cost*, we can easily find the VNF to be migrated (denoted by *migrate_vnf*) and the destination node (denoted by *dest*) with minimum migration cost. After that, storing the pair $< migrate\_vnf, dest >$ to the result set and migrating *migrate_vnf* from the source node to *dest*. Finally, update the related information which include the computing resource demands on both source and destination nodes and the mapping relationship between *migrate_vnf* and physical nodes. Nevertheless, one thing that we should be aware of is that if $migrate\_vnf \in dedicated\_vnf$, the overload situation of u will be released and there is no need to migrate another VNF. However, if $migrate\_vnf \in vnfs$, we need to repeatedly migrate other VNFs on u until u goes back to a normal status.

Once the resource shortage situation on node u is addressed, we can also apply the above process to other physical nodes in *exceed_res_nodes* until the network becomes stable and balanced.

2) LINK-AWARE VNF MIGRATION

Aiming at solving the physical link overload situation, we propose a Link Aware VNF Migration Mechanism (LAVMM). In particular, by splitting the traffic on overloaded links and migrating them to other light-loaded links, LAVMM balances the traffic load in network. Despite this, LAVMM is still based on the VNF migration to implement traffic splitting and migration, because we eventually need to migrate the VNFs on the nodes at either end of an overloaded link. Hence, one major target of LAVMM is to reduce the high delay of SFCs via migrating the related VNFs. To fulfill this purpose, we should try to reduce the number of hops of the physical path between any two logically adjacent VNFs. Via proper VNF migration, LAVMM can reduce the bandwidth consumption of the overloaded links and bring them back to a normal status. The flow chart of LAVMM is described in Fig. 7.

The traffic path of SFC is calculated by the Dijkstra algorithm which regards the residual effective bandwidth of each link as the link weight. At the beginning, each link constituting the traffic path is calculated to be light-loaded and can satisfy the QoS requirements of SFC. With the rapid increasing of traffic, many physical links are gradually suffering from the overload situation. In this regard, it is better to migrate related VNFs instead of adjusting the traffic path, because we can re-adjust the mapping relationship between VNFs and physical nodes, thus to separate the traffic of SFCs from the heavy-loaded links.

According to the data collection and monitor module, we can obtain necessary information which include the SFCs exceeding the maximum allowed delay (stored in *exceed_sfc_list*), the overloaded links (stored in *exceed_band_links*) and the full mapping path set of SFCs (denoted by *sfc_path_list*). If

$$exceed\_sfc\_list \neq \emptyset \tag{11}$$

is satisfied, there are SFCs that need to be optimized.

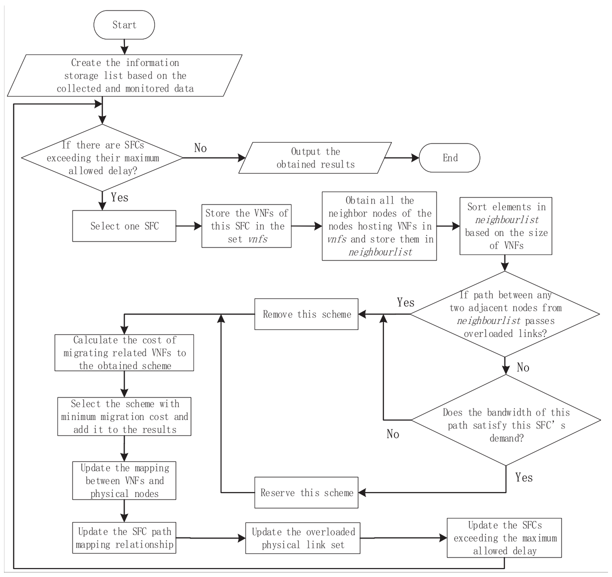

Start
Create the information storage list based on the collected and monitored data
If there are SFCs exceeding their maximum allowed delay?
No
Output the obtained results
End
Yes
Select one SFC
Store the VNFs of this SFC in the set vnfs
Obtain all the neighbor nodes of the nodes hosting VNFs in vnfs and store them in neighbourlist
Sort elements in neighbourlist based on the size of VNFs
If path between any two adjacent nodes from neighbourlist passes overloaded links?
Yes
Remove this scheme
No
Does the bandwidth of this path satisfy this SFC's demand?
No
Yes
Reserve this scheme
Calculate the cost of migrating related VNFs to the obtained scheme
Select the scheme with minimum migration cost and add it to the results
Update the mapping between VNFs and physical nodes
Update the SFC path mapping relationship
Update the overloaded physical link set
Update the SFCs exceeding the maximum allowed delay

FIGURE 7. Flow chart of Link-aware VNF migration mechanism.

Now, selecting one SFC from *exceed_sfc_list* and obtaining its service path *sfc_path* from *sfc_path_list*, then calculating the migration cost of migrating any VNF ($\in$ *vnfs*) to any neighbor node. Such process requires a lot of calculation, because there are lots of combinations. However, based on the currently overloaded nodes and links, we can reduce the calculation by filtering unnecessary combinations. Specifically, the filtering process should consider two cases: 1)if the traffic path between any two logically adjacent VNFs owns the links belonging to *exceed_band_links*; 2)if VNF migration will cause overload for the destination node. The combinations related to the two cases shall be removed. After this, we can calculate the migration cost for migrating any VNF to the destination with less time.

Finally, based on the migration cost calculated for any VNF-destination combination, we can easily obtain the one with the minimum cost and execute the VNF migration. After the migration, many indicators should be updated, which include the mapping relationship between VNFs and physical nodes (*vnf_map*), the service path (*sfc_flow_path*) and the overloaded link set (*exceed_band_links*).

At this point, the SFC selected is adjusted to a better condition. Repeatedly executing the above process for all the SFCs in *exceed_sfc_list* until the following constraint is satisfied.

$$exceed\_sfc\_list = \emptyset \tag{12}$$

VI. PERFORMANCE EVALUATION

A. SETUP

The simulation environment is based on Ubuntu 18.04.1 LTS operation system and the Docket platform [25] is used to host various VNFs. The development kit is PyCharm [26] which relies on using the Python language to simulate different modules.

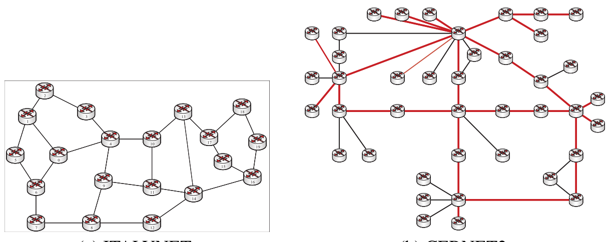


FIGURE 8. Experiment topologies.

Two real-world topologies are used to simulate the sparse and the dense network environments. As shown in Fig. 8, ITALYNET has 20 nodes and 27 links, while CERNET2 it has 42 nodes and 47 links. In particular, the computing resource of each physical node follows a uniform distribution between 50 and 100 units, while the bandwidth of each physical link is set to be 100Mbps full-duplex. Besides, the number of VNFs required by each SFC is randomly determined between 1 and 8, and the computing resource required by each VNF is randomly determined between 20 and 50 units. The processing delay of each VNF is randomly set between 10ms and 20ms. The experiments are carried out 1000 times and the average results are presented.

B. METRIC

Several metrics are additionally defined to evaluate the proposed mechanism, which are SFC optimization rate (δ_{sfc}), node optimization rate (δ_{node}), node load variance (σ^2) and link optimization rate (δ_{band}), as follows:

- δ_{sfc}: It is defined to measure the impact that the VNF migration may have on SFCs, as follows.

$$\delta_{sfc} = (\frac{|C'| - |C''|}{|C'|}) \times 100\%, \forall c \in C', c \in C'', t_c > a_c$$

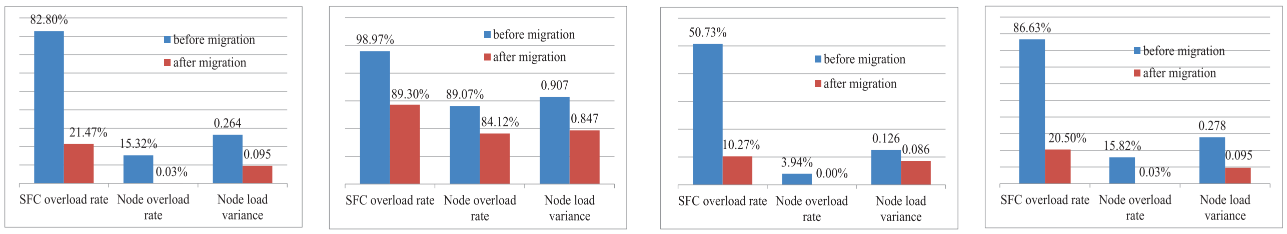
  where $|C'|$ means the number of SFCs exceeding the maximum allowed delay threshold before migration and $|C''|$ means the number of SFCs exceeding the delay threshold after migration.
- δ_{node}: It is defined to measure the effects that the VNF migration may have on physical nodes, as follows:

$$\delta_{node} = (\frac{|N'| - |N''|}{|N'|}) \times 100\%, \forall u \in N', u \in N'', \rho_u > 1,$$

  where $|N'|$ indicates the number of overloaded nodes before migration and $|N''|$ indicates the number of overloaded nodes after migration.
- σ^2: It is defined to measure the load balance condition of network, as follows:

$$\sigma^2 = \sum(\rho_u - \overline{\rho_u})^2, \quad \forall u \in N,$$

  where $\rho_u = \frac{\sum x_u^v req_v}{cap_u}$ is the rate between the occupied resource and the maximum capacity of u.



(a) `case 1` in ITALYNET (b) `case 2` in ITALYNET (c) `case 1` in CERNET2 (d) `case 2` in CERNET2

FIGURE 9. Effectiveness of node-aware VNF migration.

- δ_{band}: It is defined to measure the effects that the VNF migration may have on physical links, as follows:

$$\delta_{band} = (\frac{|E'| - |E''|}{|E'|}) \times 100\%, \forall l \in E', l \in E'', \rho_l > 1,$$

where $|E'|$ and $|E''|$ indicate the number of links exceeding the predefined bandwidth usage threshold before and after VNF migration.

C. RESULTS

In this section, we first evaluate the effectiveness of the proposed mechanisms which include NAVMM and LAVMM. After that, the proposed mechanism is compared with the state-of-the-art methods.

1) EFFECTIVENESS

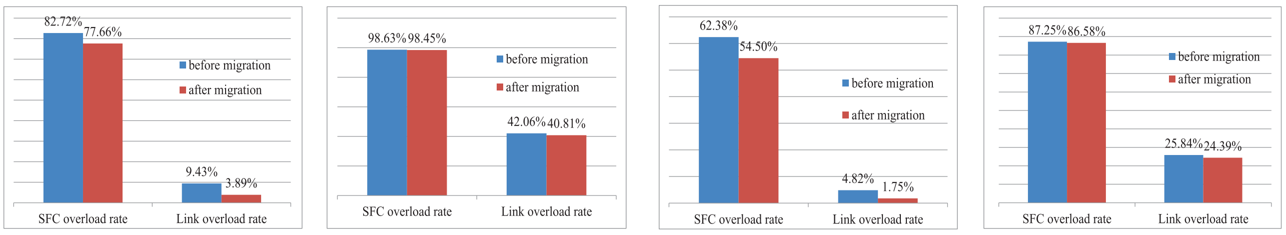
NAVMM and LAVMM should be first validated before we can compare them with other methods. In this condition, two cases of parameter combinations are considered. Specifically, `case 1` selects 5 SFCs and each SFC demands at most 5 VNFs randomly, while `case 2` selects 8 SFCs and each SFC demands 8 VNFs at most. Hence, the maximum number of VNFs required in `case 1` is 25, while that in `case 2` is 64, which can simulate the different load situations.

For NAVMM, the validation results are presented in Fig. 9, where the SFC and node overload rates are mainly calculated. Via comparing the results before and after the VNF migration, we can discover that the SFC and node overload rates are both decreased either in ITALYNET or CERNET2. For example, in ITALYNET with `case 1`, the SFC overload rate decreases from 82.8% to 21.47%, and the physical node overload rate decreases from 15.32% to 0.03%, so that the SFC optimization rate is $\frac{82.8\% - 21.47\%}{82.8\%} = 74.07\%$ and the node optimization rate is $\frac{15.32\% - 0.03\%}{15.32\%} = 99.78\%$. Meanwhile, the load variance of physical nodes also decreases from 0.264 to 0.095, which means that the network becomes more stable after executing NAVMM. Likewise, the SFC and node optimization rates with `case 2` are 9.77% and 5.56%, which are smaller than that in `case 1`. That is because the the number of service requests in `case 2` is large and they would consume a lot of computing and bandwidth resource. In this way, migrating VNFs from heavy overloaded nodes to moderate overloaded nodes will not improve the SFC performance very much. Moreover, given the same parameters, NAVMM has better performance in CERNET2 than in ITALYNET when comparing (c)/(d) with (a)/(b). The reasons are 1)CERNET2 is large-scale with a lot of available resource, so that it can accommodate more traffic; 2)Part of VNFs on the overloaded nodes are migrated to achieve the global network balance.

For LAVMM, the validation results are shown in Fig. 10, where the SFC and link overload rates are mainly calculated. By migrating the VNFs at either end of a busy link, LAVMM can relieve the link congestion situation. For instance, given `case 1` in ITALYNET, the link overload rate decreases from 9.43% before migration to 3.89% after migration, so that the link optimization rate is $\frac{9.43 - 3.89}{9.43} = 58.73\%$. Similarly, observing the results in Fig. 10(b), (c) and (d), it is easy to discover that the network situation is more or less improved after executing LAVMM which intends to steer the migration traffic from heavy-loaded links to light-loaded ones. Nevertheless, we should be aware that LAVMM and NAVMM have independent focuses and they are actually complementary, because the node and link overload cases usually happen at the same time in the practical situation. Thus, the joint usage of them (denoted by NLAVMM) is validated next.

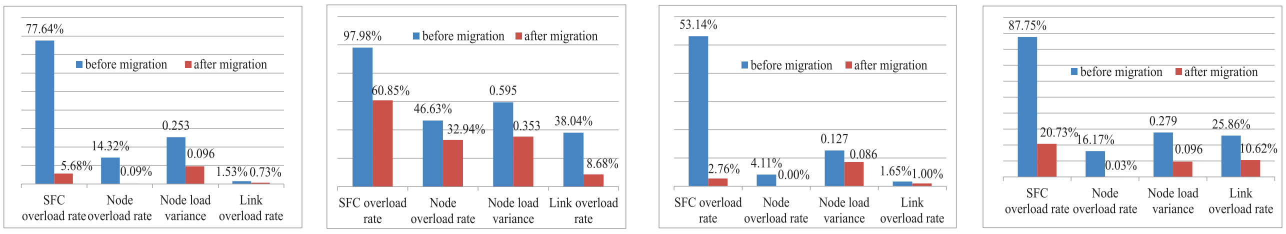
The effectiveness results of NLAVMM are shown in Fig. 11, where the additionally defined four metrics are all calculated. In principle, the average performance achieved by NLAVMM should be higher than that of NAVMM and LAVMM. Firstly, for `case 1` in ITALYNET, NLAVMM achieves 92.68% SFC optimization rate, 99.41% node optimization rate and 52.54% link optimization rate. Secondly, for `case 2` in ITALYNET, NLAVMM achieves 37.89% SFC optimization rate, 29.37% node optimization rate and 77.19% link optimization rate. Thirdly, for `case 1` in CERNET2, NLAVMM achieves 94.81% SFC optimization rate, 100% node optimization rate and 39.46% link optimization rate. Lastly, for `case 2` in CERNET2, NLAVMM achieves 76.38% SFC optimization rate, 99.81% node optimization rate and 58.95% link optimization rate. These indicators are summarized in Tables 2 and 3, where the X symbol means no statistics of this item.

Via comparing and analyzing these values in Tables 2 and 3, we can conclude that 1)The proposed mechanisms are efficient and effective; 2)NLAVMM is more efficient than using either NAVMM or LAVMM independently.



(a) `case 1` in ITALYNET (b) `case 2` in ITALYNET (c) `case 1` in CERNET2 (d) `case 2` in CERNET2

FIGURE 10. Effectiveness of link-aware VNF migration.



(a) `case 1` in ITALYNET (b) `case 2` in ITALYNET (c) `case 1` in CERNET2 (d) `case 2` in CERNET2

FIGURE 11. Effectiveness of the joint-usage of node and link aware VNF migration.

TABLE 2. Performance comparison - `case 1`**.**

Metrics	ITALYNET					CERNET2				
	NAVMM	LAVMM	NLAVMM	OVMM	RVMM	NAVMM	LAVMM	NLAVMM	OVMM	RVMM
SFC optimization rate (δ_{sfc})	74.07%	6.12%	92.68%	84.15%	90.02%	79.8%	12.63%	94.81%	92.8%	96%
Node optimization rate (δ_{node})	99.78%	X	99.41%	55.72%	13.23%	100%	X	100%	87.4%	54.13%
Node load variance (σ^2)	0.095	X	0.096	0.55	0.87	0.086	X	0.086	0.34	0.6
Link optimization rate (δ_{band})	X	58.73%	52.54%	52.1%	57.24%	X	63.76%	39.46%	39.5%	41.2%

TABLE 3. Performance comparison - `case 2`**.**

Metrics	ITALYNET					CERNET2				
	NAVMM	LAVMM	NLAVMM	OVMM	RVMM	NAVMM	LAVMM	NLAVMM	OVMM	RVMM
SFC optimization rate (δ_{sfc})	9.77%	0.18%	37.89%	31.3%	40.1%	76.33%	0.77%	76.38%	80.05%	87.3%
Node optimization rate (δ_{node})	5.56%	X	29.37%	24.6%	2.31%	99.8%	X	99.81%	61.7%	43.2%
Node load variance (σ^2)	0.0847	X	0.0353	0.33	0.52	0.095	X	0.096	0.12	0.38
Link optimization rate (δ_{band})	X	2.97%	77.19%	50.9%	55.2%	X	5.59%	58.95%	48.1%	52.78%

Therefore, we mainly compare NLAVMM with other methods.

2) PERFORMANCE COMPARISON

In this section, the proposed NLAVMM is compared with an Optimized VNF Migration Mechanism (OVMM) [14] and a Real-time VNF Migration Mechanism (RVMM) [18], where OVMM focuses on minimizing the migration cost by trying to reduce the traffic rates, while RVMM focuses on minimizing the network latency by using a local optimal search method.

Due to the fact that four network optimization indicators under two specific cases are calculated by NLAVMM, we implement OVMM and RVMM respectively and first run them using the parameters of `case 1` and `case 2` in ITALYNET and CERNET2. The corresponding results are also summarized in Tables 2 and 3. Via analyzing the data in Tables 2 and 3, we can discover that the average performance of NLAVMM is better than that of OVMM and RVMM. However, the performance of NAVMM or LAVMM is worse than that of OVMM and RVMM. For example, in ITALYNET `case 1`, the SFC optimization rates achieved by different methods can be ranked in the way: NLAVMM(92.68%) $>$ RVMM(90.02%) $>$ OVMM(84.15%) $>$ NAVMM(74.07%) $>$ LAVMM(6.12%).

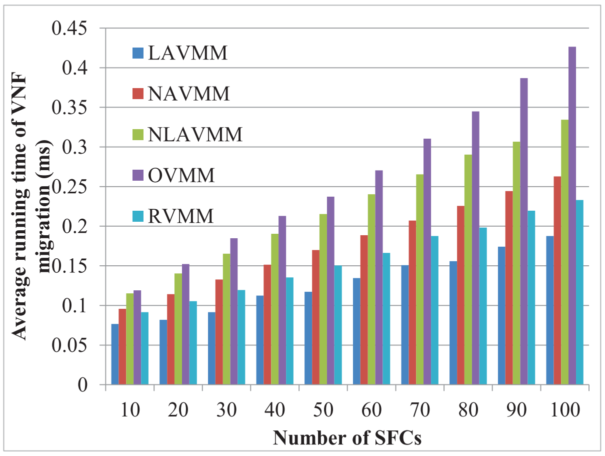


FIGURE 12. Average time spent on VNF migration and decision.

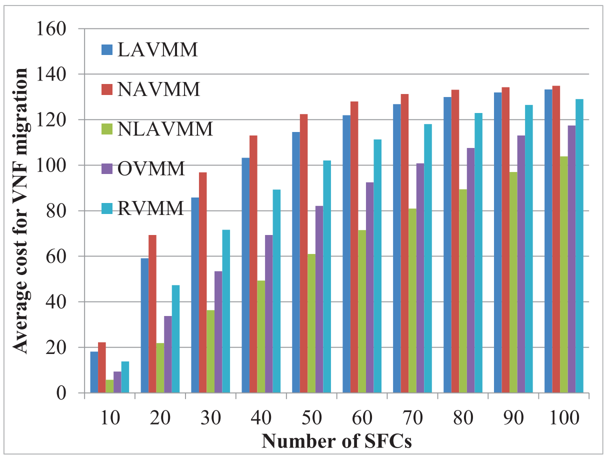


FIGURE 13. Average cost spent on VNF migration.

In some certain points, NLAVMM does not show the best performance. For instance, in ITALYNET `case 2`, the SFC optimization rate of RVMM is 96% which is higher than that of NLAVMM (i.e., 94.81%). Despite this, there is not much difference in their SFC performance. Furthermore, we can still observe that the network stability achieved by NLAVMM is higher than that achieved by OVMM and RVMM, because the load variance of NLAVMM is the smallest in 1)either ITALYNET or CERNET2; 2)either `case 1` or `case 2`.

In order to present a dynamic simulation process, the arrival of SFC request is now set to follow the Poisson distribution with an average rate of 5 requests in every 100 time units. The life-cycle of each SFC follows an exponential distribution with 1000 time units in average. Besides, the simulation topology is set to be CERNET2. Then, according to the above setting, the running time and the migration cost are measured for comparison against the number of SFCs.

In particular, the running time is calculated by the time spent on VNF migration and decision. The corresponding results are shown in Fig. 12, where NAVMM, LAVMM, NLAVMM, OVMM and RVMM are all evaluated. Apparently, we can notice that the running time of NLAVMM is higher than that of NAVMM and LAVMM, because NLAVMM is the combination of them. On one hand, the worst time complexity of NAVMM spent on VNF migration and decision is $O(|N| * |vnfs|)$. On the other hand, the worst time complexity of LAVMM is $O(|C| * |vnfs|)$. In this way, the worst time complexity of NLAVMM is $O(|N| * |vnfs|) + O(|C| * |vnfs|) = O((|N| + |C|) * |vnfs|)$, where N is the number of physical nodes, $|C|$ is the number of SFCs and $|vnfs$ is the number of VNFs on each physical node. Considering the fact that NLAVMM is the core mechanism of this paper, we mainly compare NLAVMM with the other two benchmarks. The worst time complexity of RVMM is $O(|N| * |vnfs|)$, while that of OVMM is $O(|N| * |vnfs| * (1 + |vnfs|))$. Thus, we have that RVMM($O(|N| * |vnfs|)$) < NLAVMM($O((|N|+|C|)*|vnfs|)$) < OVMM($O(|N|*|vnfs|*(1 + |vnfs|))$). On one hand, the relationship between RVMM and NLAVMM is obvious. On the other hand, the number of nodes is generally larger than the number of services in the network, namely, $|N| > |C|$, so that the relationship between NLAVMM and OVMM is illustrated as follows:

$$\begin{aligned}
&O(|N| * |vnfs| * (1 + |vnfs|)) \\
&\quad = O(|N| * |vnfs| + |N| * |vnfs|^2) \\
&\quad \leq O(|N| * |vnfs| + |C| * |vnfs|^2) \\
&\quad \leq O(|N| * |vnfs| + |C| * |vnfs|) \\
&\quad = O((|N| + |C|) * |vnfs|)
\end{aligned}$$

Based on the above relationship among them and the results in Fig. 12, we can discover that the running time of RVMM is smaller than that of NLAVMM and OVMM, while the running time of NLAVMM is smaller than that of OVMM, which accords with the above time complexity analysis. In addition, the running times of all approaches increase with the number of SFCs, which is reasonable, because the more number of SFCs, the more time would be required for VNF decision and migration.

Finally, the migration cost is also measured. However, it is aware that RVMM does not considers the migration cost, while OVMM only regards the aggregate traffic as the migration cost. On the contrary, we consider not only the aggregate traffic, but also the migration hops and the network impact. In order to present a fair comparison, the comprehensive migration cost indicator is used, and the corresponding results are shown in Fig. 13. One common phenomenon in Fig. 13 is that the migration costs of the three methods increase with the increasing of the number of SFCs. That is reasonable, because the more SFCs, the more VNFs to be migrated, and the more cost will be spent on VNF migration. Besides, it is also easy to note that the increasing trend becomes stabilized when the number of SFC exceeds 80. Generally, the number of SFCs is proportional to the network load. Thus, when the load exceeds the processing capacity of network itself, the VNF migration cannot be carried out, which in turn prevents the migration cost from growing.

Moreover, we can see that NLAVMM achieves the smallest VNF migration cost among all the methods. There are two reasons. Firstly, NLAVMM tries to cut down the hops of the migration path, such that the aggregate traffic along this path would be reduced. Secondly, since the migration process may affect the other services using this migrated VNF, NLAVMM tries to make a balance between all the SFCs when determining the VNFs to be migrated and the migration destinations, thus to reach a global optimum situation. The VNF migration costs of NAVMM and LAVMM are relatively high. However, the core proposed method of this paper is NLAVMM. Hence, we mainly compare the performance among NLAVMM, RVMM and OVMM. Obviously, NLAVMM has the smallest migration cost, while RVMM has the highest migration cost. The performance of OVMM is in between that of NLAVMM and RVMM. This phenomenon is reasonable. On one hand, for RVMM, it targets on minimizing the network latency. However, it searches the local optimum solution for the currently arriving service request, which can reduce the network latency in some certain points. In the long run, some follow-up services may require a long path before reaching the destination, such that the aggregate traffic and the number of hops passed become large, which directly increases the migration cost. On the other hand, for OVMM, it tries to minimize the cost spent on traffic aggregation by searching the best match between the source and the destination, that is, the destination with the maximum available resource will selected. However, the following requests with large bandwidth requirements will have to make a far detour before reaching destinations, such that the routing path becomes long and the migration cost becomes large.

VII. CONCLUSION

In this work, we propose a network-aware VNF migration mechanism which includes two parts, that is, the node-aware and link-aware VNF migrations, to address the problems caused by node and link overload situations respectively. To implement them, we first build a VNF migration friendly model according to the characteristics of dynamic traffic in NFV. Then, based on the formulated model, two VNF migration strategies are designed, during which the migration impact and cost are both taken into consideration. Finally, we implement the system modules and carry out experiments over ITALYNET and CERNET2 topologies. The results indicate that the proposed mechanisms are effective and can improve the performance of both network and SFC. The future work include 1) sharing VNFs among different SFCs to increase the resource utilization; 2) calculating the migration cost based on more metrics such as the VNF importance and the service interruption.

REFERENCES

[1] S. Deering and R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, document RFC 2460, 2017.

[2] J.-J. Lin, K.-C. Wang, S.-M. Cheng, and Y.-C. Liu, "On exploiting SDN to facilitate IPv4/IPv6 coexistence and transition," in *Proc. IEEE Conf. Dependable Secure Comput.*, Aug. 2017, pp. 473–474.

[3] D. Gu, Y. Xue, D. Wang, Z. Luo, and B. Yan, "Improving IPv6 transition management with IPv6 network virtualization," in *Proc. 9th Int. Conf. Adv. Infocomm Technol. (ICAIT)*, Nov. 2017, pp. 95–104.

[4] (2014). *NFV White Paper*. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf

[5] U. Ashraf and C. Yuen, "Capacity-aware topology resilience in software-defined networks," *IEEE Syst. J.*, vol. 12, no. 4, pp. 3737–3746, Dec. 2018.

[6] J. Halpern and C. Pignataro, *Service Function Chaining (SFC) Architecture*, document 7665, 2015.

[7] X. Wang, X. Chen, C. Yuen, W. Wu, M. Zhang, and C. Zhan, "Delay-cost tradeoff for virtual machine migration in cloud data centers," *J. Netw. Comput. Appl.*, vol. 78, pp. 62–72, Jan. 2017.

[8] D. Kapil, E. S. Pilli, and R. C. Joshi, "Live virtual machine migration techniques: Survey and research challenges," in *Proc. 3rd IEEE Int. Adv. Comput. Conf. (IACC)*, Feb. 2013, pp. 963–969.

[9] H. Woesner and D. Verbeiren, "SDN and NFV in telecommunication network migration," in *Proc. 4th Eur. Workshop Softw. Defined Netw.*, Sep. 2015, pp. 125–126.

[10] V. Medina and J. M. García, "A survey of migration mechanisms of virtual machines," *ACM Comput. Surv.*, vol. 46, no. 3, pp. 1–33, Jan. 2014.

[11] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," *IEEE/ACM Trans. Netw.*, vol. 25, no. 4, pp. 2008–2025, Aug. 2017.

[12] J. Zhang, D. Zeng, L. Gu, H. Yao, and M. Xiong, "Joint optimization of virtual function migration and rule update in software defined NFV networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–5.

[13] J. Xia, D. Pang, Z. Cai, M. Xu, and G. Hu, "Reasonably migrating virtual machine in NFV-featured networks," in *Proc. IEEE Int. Conf. Comput. Inf. Technol. (CIT)*, Dec. 2016, pp. 361–366.

[14] J. Xia, Z. Cai, and M. Xu, "Optimized virtual network functions migration for NFV," in *Proc. IEEE 22nd Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2016, pp. 340–346.

[15] B. Zhang, J. Hwang, and T. Wood, "Toward online virtual network function placement in software defined networks," in *Proc. IEEE/ACM 24th Int. Symp. Qual. Service (IWQoS)*, Jun. 2016, pp. 1–6.

[16] J. Huang, Q. Wu, Y. Liu, B. Wang, and W. Wang, "Virtual network embedding by node-splitting," in *Proc. 15th IEEE Int. Conf. Commun. Technol.*, Nov. 2013, pp. 329–333.

[17] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1206–1243, 2nd Quart., 2018.

[18] D. Cho, J. Taheri, A. Y. Zomaya, and P. Bouvry, "Real-time virtual network function (VNF) migration toward low network latency in cloud environments," in *Proc. IEEE 10th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2017, pp. 798–801.

[19] W. Cerroni and F. Callegati, "Live migration of virtual network functions in cloud-based edge networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 2963–2968.

[20] Y. Jia, C. Wu, Z. Li, F. Le, and A. Liu, "Online scaling of NFV service chains across geo-distributed datacenters," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 699–710, Apr. 2018.

[21] B. Yi, X. Wang, K. Li, S. K. Das, and M. Huang, "A comprehensive survey of network function virtualization," *Comput. Netw.*, vol. 133, pp. 212–262, Mar. 2018.

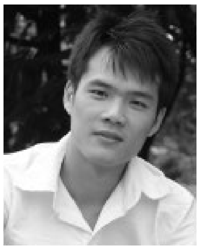
[22] K. Han, S. Li, S. Tang, H. Huang, S. Zhao, G. Fu, and Z. Zhu, "Application-driven end-to-end slicing: When wireless network virtualization orchestrates with NFV-based mobile edge computing," *IEEE Access*, vol. 6, pp. 26567–26577, 2018.

[23] R. Cziva and D. P. Pezaros, "Container network functions: Bringing NFV to the network edge," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 24–31, Jun. 2017.

[24] R. Cziva, S. Jouet, K. J. S. White, and D. P. Pezaros, "Container-based network function virtualization for software-defined networks," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2015, pp. 415–420.

[25] *Docker Platform*. Accessed: 2019. [Online]. Available: https://www.docker.com/

[26] *PyCharm*. Accessed: 2019. [Online]. Available: http://www.jetbrains.com/



BO YI received the B.S. and M.S. degrees in computer science from the South-Central University for Nationalities, Wuhan, China, in 2012 and 2015, respectively. He is currently pursuing the Ph.D. degree with Northeastern University, Shenyang, China. His research interests include routing and service function chain in SDN and NFV.

XINGWEI WANG received the B.S., M.S., and Ph.D. degrees in computer science from Northeastern University, Shenyang, China, in 1989, 1992, and 1998, respectively. He is currently a Professor with the College of Computer Science and Engineering, Northeastern University, Shenyang. He has published more than 100 journal articles and refereed conference papers and books and book chapters. His research interests include cloud computing and future Internet. He has received several best paper awards.

MIN HUANG received the B.S. degree in automatic instrument, the M.S. degree in systems engineering, and the Ph.D. degree in control theory from Northeastern University, Shenyang, China, in 1990, 1993, and 1999, respectively. She is currently a Professor with the College of Information Science and Engineering, Northeastern University, Shenyang. Her research interests include modeling and optimization for logistics and supply chain systems. She has published more than 100 journal articles, books, and refereed conference papers.

KEXIN LI received the B.S. degree in software engineering from the Harbin University of Science and Technology, Harbin, China, in 2015, and the M.S. degree in computer technology from Northeastern University, Shenyang, China, in 2016, where she is currently pursuing the Ph.D. degree. Her research interests include edge computing and artificial intelligence.