Journal of Network and Systems Management (2020) 28:436–456
https://doi.org/10.1007/s10922-020-09520-5

Dynamic VNF Placement to Manage User Traffic Flow in Software-Defined Wireless Networks

Tahira Mahboob[1] · Young Rok Jung[2] · Min Young Chung[1]

Received: 12 July 2019 / Revised: 13 December 2019 / Accepted: 28 February 2020 /
Published online: 13 March 2020

Abstract

In a Software-Defined Wireless Network (SDWN), Network Function Virtualization (NFV) technology enables implementation of network services using software. These softwarized network services running on NFV nodes, i.e., commercial servers with NFV capability, as virtual machines are called Virtual Network Functions (VNFs). To provide services to users several different VNFs can be configured into one logical chain referred to as a Service Function Chain (SFC). While receiving services from a specific VNF located at an NFV node, a mobile user may change its location. This user may continue to receive service from an associated VNF by routing flows through a new NFV node that is closest to its current location. This may introduce an inefficient routing path which may degrade the network performance. Therefore, it is feasible to relocate the VNFs associated with the service chain of the user to other NFV nodes. To relocate VNFs optimally, we need a new optimal routing path. However, if some NFV nodes on this new path are overloaded, placing these VNFs on overloaded NFV nodes affects the performance of the service chain. To solve this problem, this paper proposes an efficient method for dynamically relocating VNFs by considering changes of a user's location and the resources currently available at the NFV nodes. The performance of the proposed scheme is evaluated using simulations and an experimental testbed for multiple scenarios under three different network topologies. Results indicate that the proposed scheme balances the load on NFV nodes, reduces SFC blocking rates, and improves the network throughput.

Keywords Software-Defined Networking (SDN) · Network Function Virtualization (NFV) · Service Function Chaining (SFC) · Virtual Network Function (VNF) · VNF relocation

✉ Min Young Chung
mychung@skku.edu

Extended author information available on the last page of the article

1 Introduction

Massive influx in end user devices has led to growing demands for internet usage anywhere and anytime. Technologies such as Software Defined Networking (SDN), Content Delivery Network (CDN), and Cloud Computing (CC) have been proposed to facilitate the rapidly increasing data traffic demands of users [1, 2]. In this context, service providers have actively studied Network Function Virtualization (NFV) [3, 4]. In traditional networking, proprietary hardware and software are co-located in network devices. NFV technology can assist in solving this problem by separating software from hardware. This enables the replacement of high-end dedicated hardware devices, such as routers, firewalls, and load balancers, with software-based applications running as virtual machines on industry standard servers. These virtualized network services/applications are referred to as Virtual Network Functions (VNFs). Service Function Chaining (SFC) can be used to configure number of VNFs into one logical chain that provides network services according to the class of the users [5]. Also, network traffic in an SFC is sequentially processed and transmitted through an ordered set of VNFs [6, 7].

An efficient service chain can be constructed by recognizing the Network Service Header (NSH) [8], by distributing traffic flows in the service chain based on reinforcement learning [9], or by relocating VNFs when the users request the service chains [10]. Existing studies have not been able to cope with increased network traffic due to inefficient routing paths. These inefficient routing paths may be created as a result of user mobility. It is important to efficiently manage traffic flows in a dynamic networking environment. In this context, the dynamic placement of VNFs on the nodes in the network is a key challenge. Network resources must be taken into account before the deployment of VNFs on NFV nodes. To efficiently utilize the network resources, two main factors have a major impact; placement of VNFs on physical nodes and the resources allocated to a VNF in terms of computational capability of the physical host. The decisions for VNFs placement and resource allocations are eventually mapped to routing of flows in the network.

A mobile user receiving service from an SFC may change its location. It is feasible to relocate the VNFs associated with the SFC of the user to other NFV nodes. However, in some cases relocating the VNFs to nodes on an optimal path may lead to inefficiencies. This is because some of the nodes on the new optimal path may be overloaded which may degrade service performance. Therefore, in this paper we propose an efficient method for the deployment of VNFs on the NFV nodes. This method accounts for the location of the user and considers the current physical resources at the nodes such as CPU and memory capacities. The main idea is to find a near optimal routing path for relocating the VNFs if the NFV nodes on the optimal path are overloaded. We base our scheme on the fact that server resources have to be split among the VNFs placed on the same physical nodes. Consequently, efficient handling of server resources improves the network performance.

The remainder of this paper is organized as follows. Section 2 reviews related works and highlights the novelty of our contribution. Section 3 introduces the proposed scheme and problem formulation. Section 4 presents performance evaluation results and Sect. 5 concludes the paper.

2 Related Works

In general, there are three different approaches adopted by researchers for efficient management of resources in SDWNs; the network-centered approach, the user-centered approach, and the service-provider approach.

Network-centered approach Several researchers have adopted a network-centered approach to address the VNF embedding problem [11–13]. Carpio et al. have studied the VNF placement problem with replications to balance load in the network considering links efficiencies [11]. However, replications of VNFs may require additional costs. In Kuo et al. [12], have proposed a dynamic programming-based solution to optimize both the lengths of the routing path and the number of reused VNFs for SFC requests under the given traffic demands and network topologies. However, network performance may change with dynamic traffic demands. Another solution was proposed by Hirwe et al. for minimizing load on network nodes leveraging joint VNF placement and routing path optimization [13]. Since it minimizes the length of the path traversed by traffic flow, its performance may severely degrade due to the VNF deployed on an overloaded node on the shortest path.

User-centered approach The user-centered approach aims to optimize user experience. To minimize service delay, Agarwal et al. have proposed a decoupled solution for joint VNF placement and CPU allocation in 5G network [14]. In the first phase they deployed VNFs on the nodes in the network and in the second phase they allocated CPU to these VNFs. Computing resources were flexibly assigned to each VNF to reduce the service time of SFCs. If other constraints are considered such as memory size, link bandwidth, etc., the performance of the proposed scheme may vary. Wang et al. suggested an SFC latency estimation model for the service-scheduling problem [15]. To balance load on servers of varying capacities, they used stochastic latency bound as a metric. However, the proposed model was evaluated on a small network. To minimize inter-cloud traffic and response times, Bhamare et al. proposed an ILP model [16]. They used an affinity-based greedy algorithm to deploy VNFs in a geographically distributed cloud. For small-scale data centers located at the network edge, Martini et al. proposed a resource constrained shortest path optimization problem on an auxiliary layered graph [17]. VNFs were placed in data centers over the path that minimized network latency. In Cziva et al. [18], allocated VNFs to a distributed edge infrastructure and then minimized end-to-end latency of all users with their associated VNFs in the edge network. Although an optimal solution for VNFs placement was presented, the cost and complexity of the proposed model may be significantly high in practice. To our best knowledge, these works considered static SFC requests without user mobility and dynamic aspect of traffic requests.

Service provider approach Service providers want networks to support multiple services requiring different, yet overlapping sets of VNFs. They aim for efficient network resource utilization to reduce deployment/operational costs. To reduce operational and network traffic costs, a Markov Approximation (MA) scheme for VNF placement decisions was proposed by Pham et al. [19]. Using MA combined with matching theory, they first found a random subset of nodes to deploy VNFs and then placed the selected VNFs with the objective to minimize the total incurred cost. Selecting random subset of nodes from available nodes may not be reasonable for location-based VNF deployment. Cohen et al. proposed a near approximation algorithm for optimal placement of VNFs in the network considering operational cost [20]. They aimed to minimize both the distance costs between the user and the associated VNFs and the costs incurred due to deployment of VNFs. However, order of the VNFs in the SFC requests were not considered. Similarly, Wang et al. proposed a joint resource allocation and VNF embedding model [21]. The authors suggested a cost model leveraging trade-off between service performance and network costs. Moens et al. proposed an ILP model for placement of NFs in a hybrid environment constituted with both Physical Network Functions (PNFs) and VNFs [22]. The proposed method reduced service deployment costs and increased resource utilization. However, dynamic placement of VNFs was not considered. In [23], Savi et al. studied trade-off between upscaling of an existing VNF and hosting multiple VNFs within the same physical node. Upscaling of VNFs introduced additional costs, while the latter introduced context switching cost. Since the model was evaluated on a small scale network, performance may vary when the model is evaluated on a larger network.

From the review of literature, we analyzed that VNFs placement decisions were constrained by resources available at physical nodes. This implied that either a node had enough spare capacity to offer a VNF, or it did not. Also, individual resource requirements of different service classes were ignored before deployment of SFCs. Moreover, dynamic relocation of VNFs was not considered within the SFC embedding solutions. To solve these problems, this paper proposes a model for flexible allocation of server resources such as CPU and memory to VNFs. Our proposed scheme efficiently relocates the VNFs considering the current location of the user and the available resources on the NFV nodes hosting these VNFs. If the nodes on the optimal routing path are overloaded, the VNFs are relocated on a sub-optimal routing path.

3 Proposed Scheme

When a mobile user receiving service from a specific VNF located on an NFV node changes its location, the VNFs associated with the service chain of the user have to be relocated to other nodes. The relocation of VNFs results in a new optimal path for the service chain of the user. If nodes on the optimal routing path are overloaded, relocating VNFs on these overloaded nodes may degrade service performance. To solve this problem, we propose a dynamic placement method for VNFs that considers the location of the user and the current load on NFV nodes. The main idea is to relocate VNFs associated with the SFC of the user on near optimal routing path, if the nodes on the optimal routing path are overloaded.

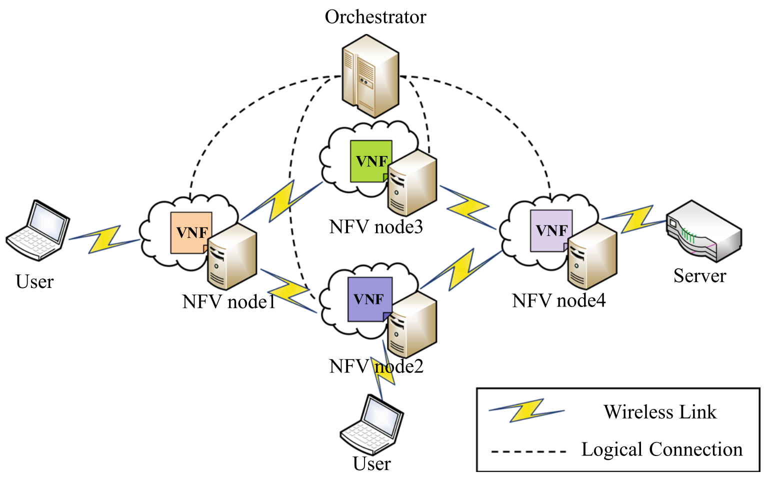


Fig. 1 An example of an SDWN

3.1 Software-Defined Wireless Network (SDWN)

We consider SDWN consisting of NFV nodes, an orchestrator, users and a server as presented in Fig. 1. The orchestrator manages the NFV nodes connected to it. The wireless NFV node has two Wi-Fi interfaces; an AP-mode interface that provides wireless LAN services to its connected users and a client-mode interface that connects NFV nodes to the internet and other NFV nodes. Users can connect to this network using mobile devices such as mobile phones and laptops.

NFV defines the standards of computing, storage and networking resources that are used to build VNFs. To provide network services to users, VNFs execute on a wireless NFV nodes. The latter transmits three types of information to the orchestrator; resource information, VNF-information, and traffic information of connected users. The orchestrator stores information received from NFV nodes in the databases. The network topology database contains three types of information; resource information, topology information of NFV nodes, and information of currently connected users of the NFV nodes. The NFV management database contains two types of information; the service chain rule specific to the class of the user and the VNF-information used in the service chain. Several different types of VNFs are configured into SFC depending on the class of the user. Service chain rules are created according to rules set by the network administrator. The orchestrator transmits the VNF placement information and the service chain rules to the NFV nodes. Using the service chain rules, the routing table on the NFV node forwards the network traffic. The NFV node transmits the *"VNF-Update"* message to the orchestrator including information of currently executing VNFs.

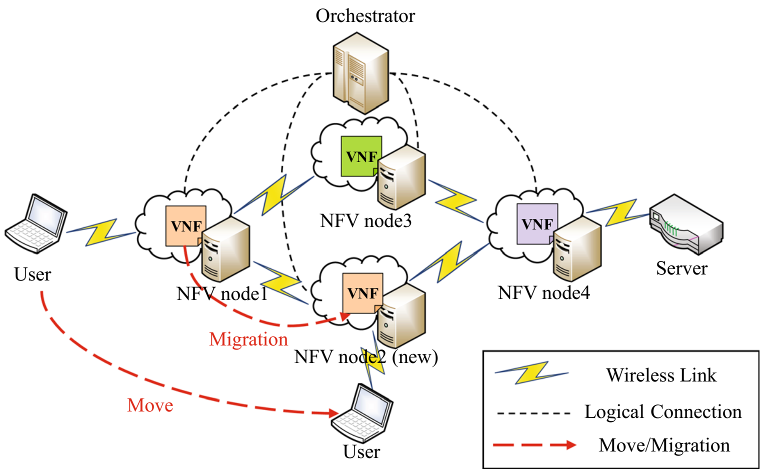


Fig. 2 Proposed scheme for dynamic user traffic flow management

3.2 Managing Traffic Flows of Mobile Users

Mobile users may change location while receiving service from a specific VNF executing on an NFV node. Users may continue to receive service from corresponding VNF via new NFV node. An example of this is shown in Fig. 2. As depicted, a user previously connected to NFV node 1 moves to a new location and receives service from NFV node 2. The current NFV node 2 records this updated connection information and transmits *"User-Update"* message to the orchestrator. This *"User-Update"* message contains Medium Access Control (MAC) address of the connected user. The orchestrator receives *"User-Update"* message and estimates change in the location of the user. This is done by comparing previously stored location of the user, i.e., location of the user at the first connection establishment to NFV node 1, present in the network topology database. The orchestrator transmits *"VNF-information"* of NFV node 1 to NFV node 2, so the associated VNF of the user can be relocated/migrated.

To improve network throughput and reduce latency, it is reasonable to relocate VNFs associated with the service chain of the user to other nodes on the new optimal routing path. However, this is not practical since multiple users may be receiving service from the particular VNF. Also, the available resources at the NFV node should be considered before deploying a VNF. Furthermore, it is not reasonable to relocate the VNF for every change in the user's location. Therefore, it is not relocated immediately, the traffic flow is routed to the user via the previous node while it is connected to the current node. It is to be noted that latency metric is considered for making relocation decisions of a VNF. The delays experienced by all the users receiving service from a particular VNF are transmitted as *"Latency-Information"* to the orchestrator after a pre-defined time interval. The VNF must be relocated to another node if $d_x \geq d_{th}$, where d_x and d_{th} are the delay experienced by a user receiving service from the corresponding VNF and the pre-defined threshold delay for the relevant VNF, respectively. The

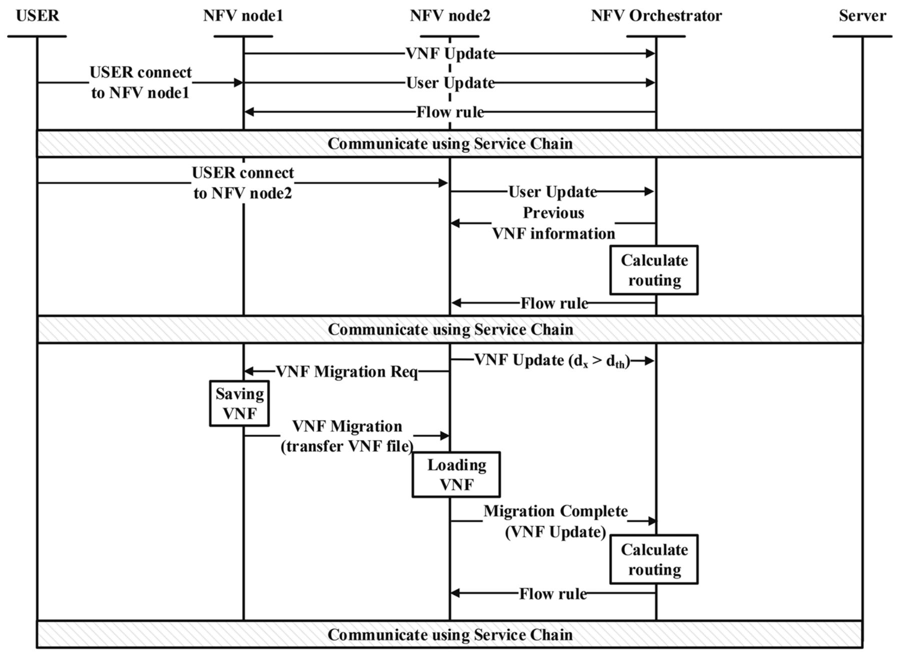

USER
NFV node1
NFV node2
NFV Orchestrator
Server
VNF Update
USER connect to NFV node1
User Update
Flow rule
Communicate using Service Chain
USER connect to NFV node2
User Update
Previous VNF information
Calculate routing
Flow rule
Communicate using Service Chain
VNF Update ($d_x > d_{th}$)
VNF Migration Req
Saving VNF
VNF Migration (transfer VNF file)
Loading VNF
Migration Complete (VNF Update)
Calculate routing
Flow rule
Communicate using Service Chain

Fig. 3 Signaling flow of VNF relocation for the proposed scheme

orchestrator estimates new optimal routing path for relocating the VNFs. Signaling flow information for the proposed VNF relocation method is presented in Fig. 3.

When a VNF relocation request is received at the orchestrator, the next step is to estimate its optimal placement in the network. The dynamic placement of VNFs considering available resources at the NFV nodes is discussed in the next sub-section.

3.3 Dynamic Placement of VNFs Considering Load of NFV Nodes

We consider an SDWN that contains several NFV nodes. Several different types of VNFs have to be relocated on nodes satisfying CPU and memory requirements. This network provides services to the users via several different VNFs configured into various SFCs depending on the class of the user.

We represent the network architecture with a set of NFV nodes $\mathbb{S} = \left\{s_1, s_2, \dots, s_M\right\}$ and a set of VNFs $\mathbb{V} = \left\{v_1, v_2, \dots, v_N\right\}$, where M and N are the number of NFV nodes in the network and the number of different types of VNFs, respectively. The indicator function for placement of VNF on an NFV node is defined as

$$x_{ij} = \begin{cases} 1, & \text{if } v_i \text{ is executing on } s_j \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The load on an NFV node if v_i is placed on s_j can be calculated as

$$L_{ij} = \alpha \cdot U_{CPU_{ij}} + \ (1 - \alpha) \cdot U_{MEM_{ij}}, \tag{2}$$

where α is a real number in $(0, 1)$, and $U_{CPU_{ij}}$ and $U_{MEM_{ij}}$ are the CPU utilization and memory utilization, respectively. Here, $U_{CPU_{ij}}$ can be defined as $U_{CPU_{ij}} = \frac{c_i}{C_{max_j}}$, where c_i is the required CPU allocation (processor rate) for v_i and C_{max_j} is the CPU capacity of s_j expressed in hertz (Hz). $U_{MEM_{ij}}$ can be defined as $U_{MEM_{ij}} = \frac{m_i}{M_{max_j}}$. Here m_i is the RAM requirement of v_i and M_{max_j} is maximum size of RAM on s_j expressed in MB. If load on the NFV node is not managed, this may lead to inefficient utilization of resources. VNFs placement considering load on NFV nodes can be handled by assigning appropriate weight values to α. For example, if a transcoder VNF requires high computational power, α can be assigned a higher value.

The total load on an NFV node j can be calculated as

$$L_j = \sum_{i=1}^{n} L_{ij} \cdot x_{ij} \ \text{ for } 1 \ \leq \ n \ \leq N \ \text{ and all } \ j \ \text{ in } \ \{1, 2, \cdots, M\}. \tag{3}$$

VNF placement considering load on NFV nodes can be expressed using Integer Linear Programming (ILP). Therefore, we formulate the ILP model as

$$\text{Minimize} \sum_{j=1}^{M} L_j = \sum_{j=1}^{M} \left\{ \sum_{i=1}^{N} \left[\alpha \cdot U_{CPU_{ij}} + \ (1 - \alpha) \cdot U_{MEM_{ij}} \right] \cdot x_{ij} \right\} \tag{4}$$

subject to:

$$\sum_{i=1}^{N} x_{ij} \cdot U_{CPU_{ij}} \leq C_{max_j}, \quad \text{for all } j \text{ in } \ \{1, 2, \cdots, M\}, \tag{5}$$

$$\sum_{i=1}^{N} x_{ij} \cdot U_{MEM_{ij}} \leq M_{max_j}, \quad \text{for all } j \text{ in } \ \{1, 2, \cdots, M\}, \tag{6}$$

$$L_j \leq L_{th_j}, \qquad \text{for all } j \text{ in } \ \{1, 2, \cdots, M\}. \tag{7}$$

The objective of the formulated ILP problem is to minimize load on the nodes. Eq. (5) implies that sum of CPU utilization of all VNFs executing on an NFV node should be less than its CPU capacity. Similarly, in Eq. (6), the sum of RAM utilization of all VNFs executing on an NFV node should be less than its RAM size. From Eq. (7), load on s_j is constrained by a predefined load threshold value. NFV node capabilities and capacities may vary depending on hardware such as CPUs, GPUs, FPGAs, and other system specifications [24]. Assigning tasks appropriately to heterogeneous hardware is a challenging task for administrators. Performances are dependent on computing resource specifications such as the number of CPUs, GPUs, and special hardware like FPGAs. The availability of various types of hardware requires careful study to assign appropriate load threshold limits.

Therefore, the load threshold, L_{th_j}, largely depends on the hardware specifications of the node.

3.4 Heuristic Algorithm

The proposed ILP model for the dynamic placement of the VNFs considering the available load on the NFV nodes under stated constraints is an NP-hard problem. Therefore, to solve this problem we adopt the heuristic-based approach described in Algorithm 1.

We assume that a user is receiving service via a particular VNF executing on an NFV node. This user changes its location and continues to receive service from a corresponding VNF via a new NFV node. This is done by routing flows to the previous NFV node from the current NFV node. The current NFV node generates a VNF relocation request if $d_x \geq d_{th}$, where d_x and d_{th} are delay experienced by the user receiving service from corresponding v_i and the pre-defined threshold delay, respectively. The new optimal routing path for relocating VNFs associated with the SFC of the user is calculated using the shortest-path algorithm, i.e., Dijkstra's algorithm [25]. Some of the nodes on this new optimal routing path may be overloaded. Hence, nodes adjacent to overloaded NFV nodes on the sub-optimal routing path are considered for relocating the VNF. For this, we utilize the proposed heuristic algorithm to relocate VNFs on the near optimal routing path.

We consider an SDWN as a graph ($\mathscr{G}$) with several input parameters, including a set of NFV nodes ($\mathbb{S}$), a set of bi-directional edges ($\mathbb{E}$), a set of VNFs ($\mathbb{V}$), CPU (c_i) and RAM requirements (m_i) for each v_i , and the load threshold (L_j) value for each NFV node s_j. Also, M_o^* and M_{conn}^* are the number of over-loaded NFV nodes on the optimal routing path and number of NFV nodes adjacent to the overloaded NFV node on the sub-optimal routing path, respectively. As stated earlier, when a VNF relocation request is sent to the orchestrator, the shortest path for relocating VNFs associated with the service chain of the user is calculated. If an NFV node on the optimal routing path is overloaded, the proposed heuristic approach is initiated by the orchestrator. The orchestrator sets the hop count value. For example, if $M_{hop} = 1$, nodes at a one-hop distance are considered for relocating the VNFs. Also, if $M_{hop} = 2$, nodes at a two-hop distance are considered for relocating the VNFs.

For CPU and RAM utilization, the available CPU and RAM for the NFV nodes are calculated in steps 13 to 16. In steps 17 to 22, the resource load is calculated by assigning α.[1] If load on an NFV node is less than a pre-defined threshold value, v_i is placed on s_j and available CPU and RAM at s_j are updated. However, if load calculated in step 23 is greater than the load threshold value, v_i is placed on the next s_j, as estimated in step 23 to step 25. Furthermore, in step 26, each s_j is constrained by the CPU capacity (C_{max_j}) and RAM size (M_{max_j}). In step 32, if no suitable node is found for placing v_i, the value of M_{hop} is increased. The algorithm terminates when all the VNFs have been placed on the NFV nodes.

[1] Weighted values of the CPU and the RAM.

Table 1 Resource specifications for VNFs

OS	VNF name	VNF type	RAM (MB)	CPU (MHz)	Weight value (α)
TinyCore [26]	VNF 1	Authentication/ftp	16	1(i486DX), 15–50	[0.4–0.6]
SliTaz [27]	VNF 2	Billing	200	1 (×86), 3500	[0.4–0.6]
OpenWrt [28]	VNF 3	Firewall	250	1 (×86), 3500	[0.4–0.6]
Xubuntu	VNF 4	(OpenCV) [29]	8192	1–8 (×86), 3500	[0.6–0.8]
Xubuntu	VNF 5	(FFMpeg) [30]	3860	1–4 (×86), 3500	[0.6–0.8]

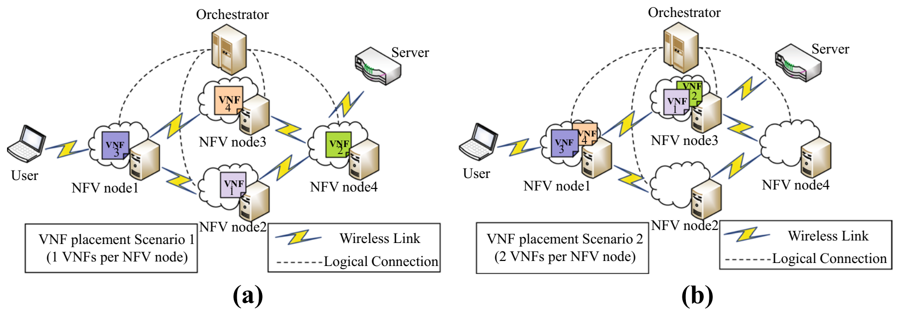


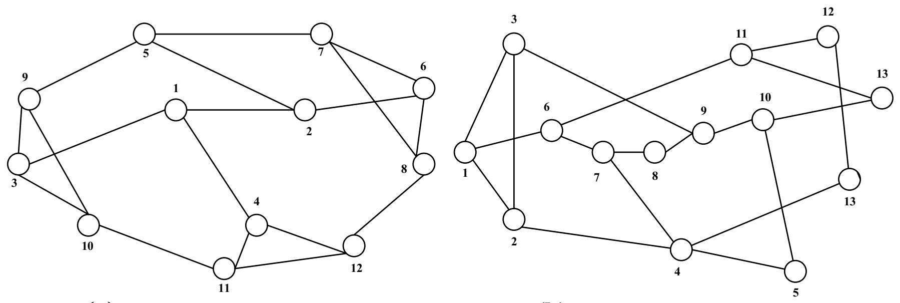
Fig. 4 An example of VNF placement for a topology with four nodes

4 Performance Evaluation

4.1 Simulation Results

To evaluate our proposed scheme, we simulate sample use case scenarios for multiple network topologies. These use case scenarios leverage real-world services. The proposed scheme is evaluated using 3.50GHz quad core computer with 16GB of RAM size. The network has to support two types of services; file transfer and video streaming. Service classes $k \in K$ can be represented as {file transfer, video streaming}, each corresponding to a VNF graph, i.e., (1) file transfer: firewall—(authentication and ftp)—billing and (2) video streaming: firewall—encoder—billing. Each VNF has a pre-defined CPU allocation and memory requirement. VNFs can be implemented using Linux based Operating System (OS) distribution and libraries. Sample flavors of leveraged VNFs are presented in Table 1.

Utilizing these VNFs resource requirements, we study two use case scenarios shown in Fig. 4 to simulate the proposed scheme. We compare the results of the proposed scheme to the scheme without Load Balancing (LB). The scheme without load balancing named *"Scheme w/o LB"* is implemented based on Dijkstra's algorithm. So its complexity is similar to the Dijkstra's algorithm. We implement three network topologies to obtain the simulation results. The first network topology consists of four nodes as shown in Fig. 4. In scenario 1, one VNF is placed on an NFV node. In scenario 2, two VNFs can be placed on an NFV node. Note that the test bed for experimental results is similar to this four node topology. The size of this



(a) **Custom topology (12 nodes)** **(b)** **NSFNET topology (14 nodes)**

Fig. 5 Sample topologies for the simulation environment

network is very small, therefore we simulate two additional topologies to evaluate the proposed scheme on a larger network.

Figure 5a presents a randomly deployed topology having 12 nodes with 18 bi-directional links and Fig. 5b shows the NSFNET [31] topology having 14 nodes with 42 bi-directional links. We set propagation delay of 10 ms for each link in the network. Additionally, all nodes are equi-distant, i.e., 10 km. To receive network services, users associate/connect to nodes in the network. Mobile users may change their association points from one node to another.

To obtain simulation results, we use {file transfer} and {video streaming} SFCs (with an SFC length = 3). VNF resource requirements such as CPU and RAM requirements and weight values of α are extracted from Table 1. SFC requests are generated by users and the VNFs associated with the service chain of a user are placed on the NFV nodes. We simulate 10,000 times with different seed values and then compute the average with a Confidence Interval (CI) of 95%.

Figure 6 shows comparison between average loads on NFV nodes for the proposed scheme and that of the scheme without LB. The topologies with four nodes and 12 nodes, and NSFNET topology with 14 nodes show 4%, 29%, and 7% lower average load values compared to those of the scheme without LB, respectively. For scenario 2, a decrease of 56%, 27%, and 36% in the average load is observed for topologies with four nodes and 12 nodes, and NSFNET topology with 14 nodes compared to those of the scheme without LB, respectively.

For the proposed scheme, topology with four nodes shows a 4.6% increase in average load from scenario 1 to scenario 2, whereas a 44% increase is observed for the scheme without LB. A limited number of nodes and fewer network paths may increase load on the nodes. Therefore, it is necessary to evaluate the proposed scheme on a larger network.

Under 12 nodes topology, from scenario 1 to 2, there is 21% increase in average load for the proposed scheme. For the same case, there is 29% increase in average load for the scheme without LB. A similar pattern is observed for the NSFNET topology. From scenario 1 to 2, average load increases by 20% for the proposed scheme. Compared to this, the scheme without LB increases average load by 48%. Furthermore, for both scenarios, the proposed scheme maintains nearly consistent

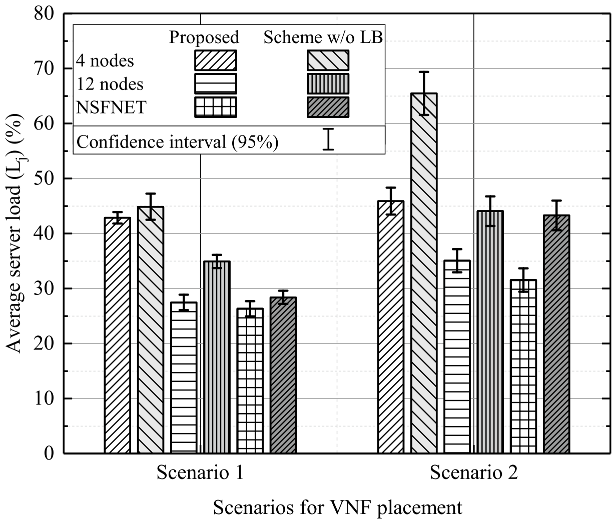


Fig. 6 Average NFV node load (L_j) for two scenarios under three different topologies

average load on the NFV nodes. On the contrary, the scheme without LB exhibits a very large increase in the average load when the number of VNFs deployed on an NFV is increased.

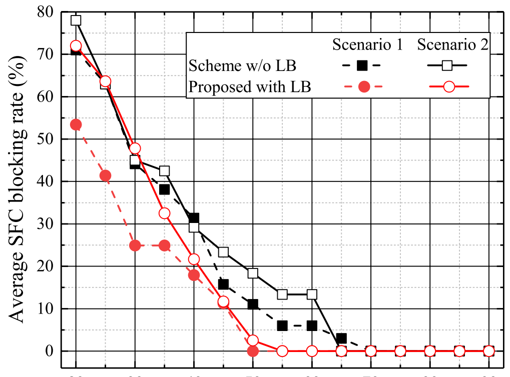
SFC request is blocked if a VNF cannot be placed on an NFV node. This occurs due to the limited physical resources on the nodes. As shown in Fig. 7a, at a load threshold of 20%, for the proposed scheme, SFC blocking rates for topology with four nodes are 71% and 54% for scenarios 1 and 2, respectively. All SFC requests are accepted at a load threshold of 50%. The scheme without LB exhibits higher SFC blocking rates, i.e., 70% for scenario 1 and 78% for scenario 2. All SFC requests are accepted at load threshold values of 70% and 65% for scenario 1 and scenario 2, respectively. Intuitively, if an NFV node is overloaded, the number of SFC requests that can be accepted by it are reduced.

As shown in Fig. 7b, in scenario 1 for 12 nodes topology, SFC blocking rate of 55% and 60% is observed at load threshold value of 20% for the proposed scheme and the scheme without LB, respectively. For scenario 2, the proposed scheme exhibits a considerably lower blocking rate compared to the scheme without LB.

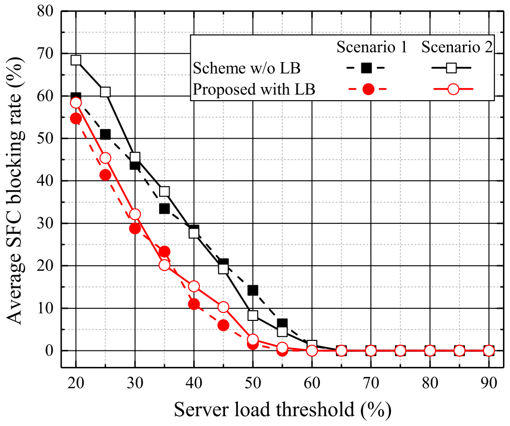
A similar pattern is observed for the NSFNET topology as shown in Fig.7c. At load threshold of 20%, the proposed scheme exhibits SFC blocking rate of 40% and 50% for scenarios 1 and 2, respectively. For the same load threshold, the scheme without LB blocks 60% and 70% of SFC requests for scenario 1 and scenario 2, respectively. In the proposed scheme, individual resource requirements of VNFs such as CPU and memory have been accounted for before placing them on the nodes. Therefore, efficient utilization of node resources enables a greater number of SFC requests that can be serviced.

To compare the time complexity of the proposed heuristic algorithm and that of the scheme without load balancing, we consider the complexity using Big $\mathcal{O}$

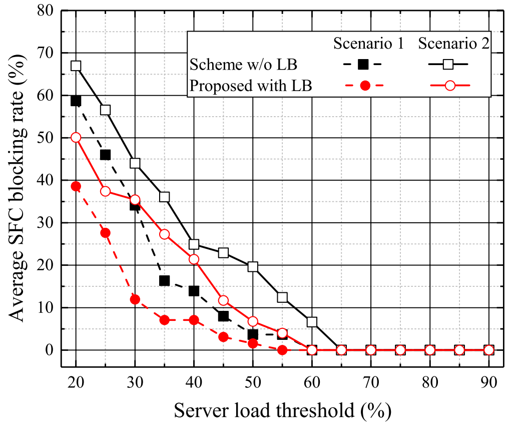
Fig. 7 Average SFC blocking rate for load threshold (L_{th_j}) values for two scenarios under three different topologies



(a) Topology with four nodes



(b) Topology with 12 nodes



(c) NSFNET topology

Table 2 Comparison of simulation execution time between the proposed scheme and the scheme w/o LB

Topology[a]	Execution time (ms)		Increment in execution time (%)
	Proposed Scheme	Scheme w/o LB	
4 nodes	2.0645	1.9616	5.25
12 nodes	2.4738	2.3321	6.08
NSFNET (14 nodes)	2.9651	2.7526	7.72

[a]Refer to the Figs. 4 and 5

notation and execution time. The scheme without load balancing performs based on Dijkstra's algorithm and has the time complexity of $\mathcal{O}(M^2)$, where M is the total number of nodes in the network. Our proposed scheme basically performs based on the Dijkstra's algorithm. In addition to this, in our proposed heuristic algorithm, an overloaded node on the optimal routing path should be replaced by an alternative node on the sub-optimal routing path. It requires the time complexity of $\mathcal{O}(M_o^*)$ to examine the alternative node, where M_o^* is the number of overloaded nodes. The algorithm examines an alternative node satisfying with the resource requirements from M_{conn}^* nodes on the sub-optimal path within M_{hop} distance from the overloaded node. Therefore, the time complexity to examine all the alternative nodes is $\mathcal{O}$ (M_{conn}^*). Our proposed scheme additionally requires the time complexity of $\mathcal{O}(M_o^*$ $M_{conn}^*)$ to determine the alternative node.

Moreover, we compare simulation execution times of our proposed scheme and those of the scheme without load balancing. In Table 2, the results show that simulation execution times of our proposed scheme are approximately 5.25, 6.08, and 7.72% larger than those of the scheme without load balancing, respectively. Even though the execution time of our proposed scheme is larger than the scheme without load balancing, we think it is reasonable.

4.2 Experimental Evaluation

To evaluate the performance of our proposed method with real-world services, we construct a real testbed. Virtual machines can be configured easily and can reduce costs in terms of required physical devices. However, virtual machines cannot use CPU virtualization or Kernel-based Virtual Machine (KVM) functions available in physical devices. To apply NFV, KVM functions must be available. Therefore, we implemented a test bed with physical devices. As shown in Fig. 8, test bed consists of one desktop and four laptops with Ubuntu OS (18.04 LTS 64 bits). The desktop computer works as an orchestrator for managing the wireless network. The orchestrator has an open source Open Network Operating System (ONOS) version 12.2 [32] running on it.

The database consists of three *hashtables* that are maintained by the orchestrator. The first is a *"VNF table"* that stores information of VNFs such as the IP addresses of NFV nodes and information of VNFs running on them. The second is *"User table"* that stores information about users such as MAC address of NFV

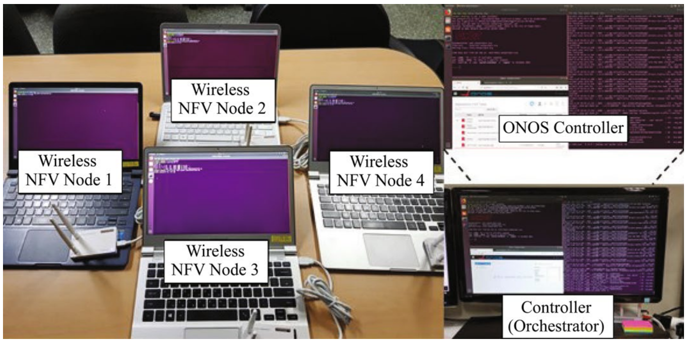


Fig. 8 Test bed environment

node they are connected to, IP address of the NFV node, and delay statistics. The third is an *"SFC table"* that stores SFC for the different users.

In ONOS, a JAVA-based application referred to as *"VNF Migration Module"* is installed. It is responsible for managing VNF relocation process. A desktop computer is connected to a WLAN router using a wired connection and indirectly connected with NFV nodes to communicate with the orchestrator. The laptops working as wireless NFV nodes are responsible for configuring, routing, and forwarding traffic. Each laptop has a Python-based application referred to as *"VNF Migration Application"* running on it. This application is responsible for configuring VNFs and it sends information of connected users to the orchestrator. In addition to this, it coordinates with the orchestrator using socket programming to migrate the requested VNF. The logical view of the testbed is presented in Fig. 9.

Laptops, i.e., NFV nodes have two wireless Wi-Fi interfaces. The first is a WLAN AP (Intel AC-7260 built-in Wi-Fi interface [33]) that provides users and servers with IEEE 802.11g wireless LAN service. The second is a WLAN interface (USB adapter with ipTIME N600UA Wi-Fi interface chipsets [34]) that connects the NFV nodes to the wireless router and other NFV nodes. The Quick Emulator (QEMU) [35] is running on the NFV nodes. This emulator provides virtualization platform for VNFs, i.e., VNF executes on it. The user is an Android 8.0-based smart-phone with TCP client application running on it. The server is a Windows 10 OS-based laptop with TCP server application running on it.

The TCP transmission rate between the user, i.e., TCP client, and the server, i.e., TCP server, is fixed at $5 Mbps$. This traffic is generated using the *iPerf*3 [36] application. Furthermore, it is also used to measure the average data rate achieved between the user and the server. The TCP packets sequence numbers and packet arrival times between the user and the server are measured using the Wireshark application. The orchestrator, wireless NFV nodes, users, and the server are located in the same local network.

The average load on NFV nodes is depicted in Fig. 10. The proposed scheme minimizes the average load on the nodes compared to that of the scheme without LB for both scenarios. Scenario 1 has one VNF and scenario 2 has two VNFs

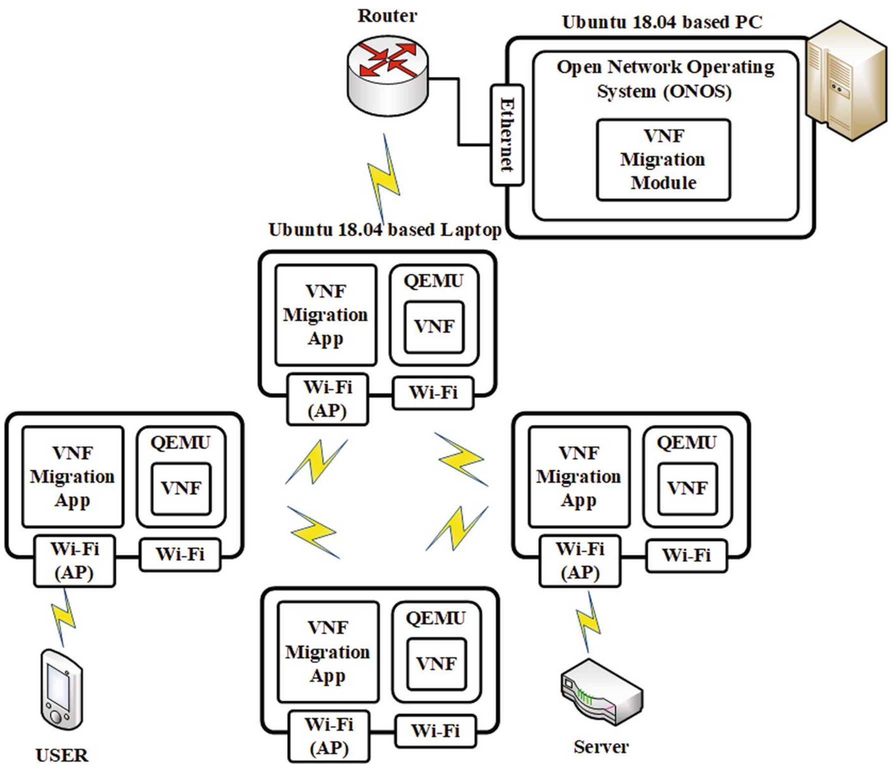


Fig. 9 Logical view of the considered NFV environment

running on the nodes. Furthermore, average load on the nodes in the testbed experiments are comparable to those obtained by simulations and only a slight variation is observed.

In the experiments, we consider that a user receives service from wireless NFV node 1 for 20 s. This user then relocates and connects to wireless NFV node 2 and is then served for 20 s. The results are obtained by repeating this experiment 30 times.

Figure 11a shows consistent throughput, i.e., approximately 16 Mbps upto 20 s. At 20 s, the user relocates and associates to NFV node 2. Throughput between the two nodes goes to zero. For the scheme without LB, only the routing rule is calculated and applied to the network. Communication for the user is restored after 1 s. When user is associated to NFV node 2 at 20 s, throughput is reduced on the link between wireless NFV nodes 1 and 2. For the proposed scheme, at 20 s, the associated VNF is relocated from wireless NFV node 1 to node 2 and the new routing rule is calculated. The communication is restored after a delay of two seconds. In the case of proposed scheme, throughput increases after the VNF relocation. This is because the service path is shortened and traffic between the client and the NFV node is decreased.

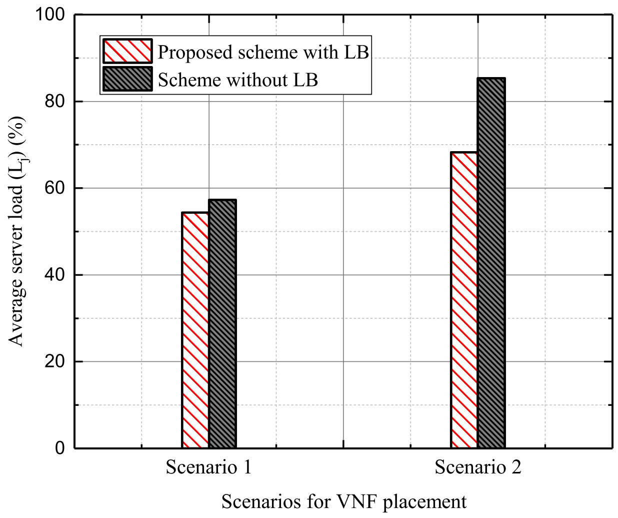


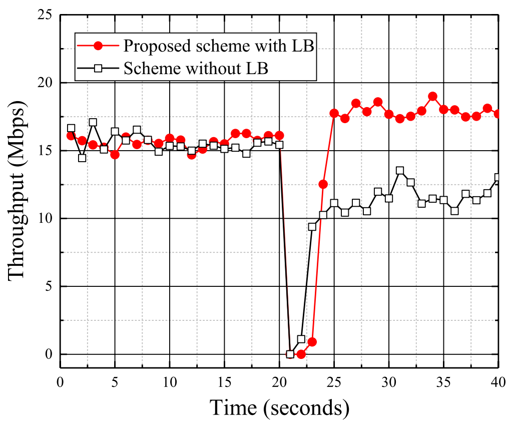
Fig. 10 Average NFV node load (L_j) for testbed

Moreover, we observe the links between wireless NFV nodes 1, 2, and 4. In Fig. 11b, before the user changes its location, the traffic volume between NFV nodes 1 and 2 is constant, i.e., approximately 2100 kB for both schemes. For the proposed scheme, after VNF relocation is complete and communication is restored, the link between wireless NFV nodes 1 and 2 is no longer used. Therefore, the traffic volume on this link goes to zero. For the scheme without LB, VNF is not migrated and the link between wireless NFV nodes 1 and 2 must be used twice. Therefore, traffic volume nearly doubles to 3500 kB on this link.

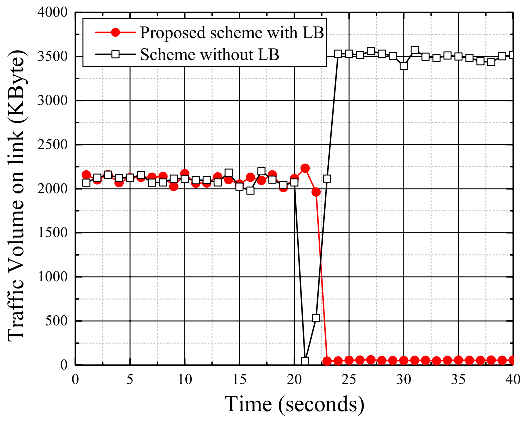
Figure11c shows the traffic volume between wireless NFV nodes 2 and 4. Proposed method takes approximately two seconds to restore communication due to VNF relocation, while the scheme without LB takes 1 s. As communication is restored, both schemes use the link between wireless NFV nodes 2 and 4 only once. Therefore, the traffic volume remains nearly constant on this link, i.e., approximately 2100 kB.

5 Conclusion

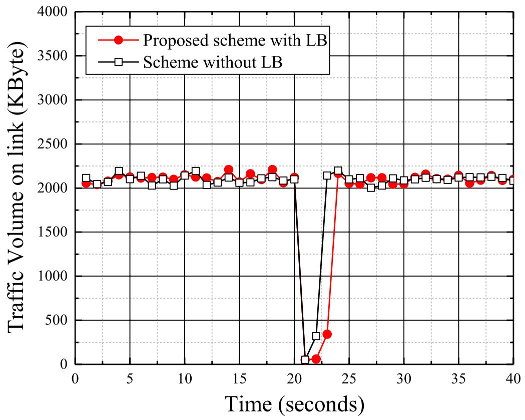
In this paper, we proposed an efficient model for dynamic placement of VNFs to manage user traffic flows in an SDWN. A mobile user may change its location while receiving service from a specific VNF located on an NFV node. The user receives service from a corresponding VNF via a new NFV node by routing flows to the previous NFV node. This may introduce an inefficient routing path, which may degrade service performance. It is feasible to relocate VNFs associated with the service chain of the user to other NFV nodes on new optimal routing path. However, some of the NFV nodes on this new optimal routing path may be overloaded. To overcome this problem, this paper proposes an efficient method for dynamic replacement



(a) Network throughput for the proposed scheme and scheme without load balancing



(b) Traffic volume on link between wireless NFV nodes 1 and 2



(c) Traffic volume on link between wireless NFV nodes 2 and 4

Fig. 11 Throughput and traffic volume for the testbed

of VNFs by considering the location of the user and the available resources at the NFV nodes, resulting in improved network performance. Existing studies mainly discuss resource-constrained VNF placement problems based on minimizing server utilization. However, they do not account for flexible resource allocation to VNFs or the location of the users. We present an efficient model for dynamic relocation of VNFs on near optimal routing path satisfying the resource requirements of CPU and memory if NFV nodes on optimal path are overloaded. The proposed scheme was evaluated using simulations for two different scenarios under three different topologies. Simulation results indicated that the topologies with four nodes and 12 nodes and NSFNET topology with 14 nodes showed 4%, 29%, and 7% decrease in average load for scenario 1 for the proposed scheme compared to that of the scheme without LB. For scenario 2, the proposed scheme decreased average load by 56%, 25%, and 36% for topologies with four nodes and 12 nodes and NSFNET topology with 14 nodes relative to that of the scheme without LB. Furthermore, the proposed scheme maintained nearly consistent load on the NFV nodes for both scenarios and was independent of network topology. Also, the SFC blocking rates were reduced considerably for the proposed scheme. Experimental results showed that the throughput of the network was approximately doubled compared to the scheme without LB. Simulation and experimental results are promising, therefore in the future, we intend to investigate the proposed scheme by considering other parameters as constraints such as bandwidth, latency, and link capacity, etc.

Acknowledgements This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (B0190-19-2013, Development of Access Technology Agnostic Next-Generation Networking Technology for Wired-Wireless Converged Networks). Prof. Min Young Chung is the corresponding author.

Author's contributions All authors contributed to the study conception and design. Material preparation, data collectionand analysis for simulations were performed by Tahira Mahboob and tested experiments wereimplemented by Young Rok Jung. The first draft of the manuscript was written by TahiraMahboob. Min Young Chung supervised the process to make the manuscript. All authorscommented on previous versions of the manuscript. All authors read and approved the finalmanuscript.

References

1. Nunes, B.A.A., Mendonca, M., Nguyen, X.-N., Obraczka, K., Turletti, T.: A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun. Surv. Tut. **16**(3), 1617–1634 (2014)
2. Jain, R., Paul, S.: Network virtualization and software defined networking for cloud computing: a survey. IEEE Commun. Mag. **51**(11), 24–31 (2013)
3. ISG, N.: Network functions virtualisation (NFV)–virtual network functions architecture. Tech. rep., ETSI (2013)
4. Haleplidis, E., Salim, J.H., Denazis, S., Koufopavlou, O.: Towards a network abstraction model for SDN. J. Netw. Syst. Manag. **23**(2), 309–327 (2015)
5. Davoli, G., Cerroni, W., Contoli, C., Foresta, F., Callegati, F.: Implementation of service function chaining control plane through openflow. In: 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV/SDN), IEEE, pp. 1–4 (2017)
6. Halpern, J., Pignataro, C.: Service function chaining (SFC architecture), RFC 7665. Tech. rep. (2015)

7. Quinn, P., Nadeau, T.: Problem statement for service function chaining, RFC 7498. Tech. rep. (2015)
8. Quinn, P., Elzur, U., Pignataro, C.: Network service header (NSH), RFC 8300. Tech. rep. (2018)
9. Chen, X., Li, Z., Zhang, Y., Long, R., Yu, H., Du, X., Guizani, M.: Reinforcement learning-based QoS/QoE-aware service function chaining in software-driven 5G slices. T. Emerg. Telecommun. **29**(11), e3477 (2018)
10. Carpio, F., Jukan, A., Pries, R.: Balancing the migration of virtual network functions with replications in data centers. In: Network Operations and Management Symposium (NOMS), IEEE, pp. 1–8 (2018)
11. Carpio, F., Dhahri, S., Jukan, A.: VNF placement with replication for load balancing in NFV networks. In: 2017 IEEE International Conference on Communications (ICC), IEEE, pp. 1–6 (2017)
12. Kuo, T.-W., Liou, B.-H., Lin, K.C.-J., Tsai, M.-J.: Deploying chains of virtual network functions: on the relation between link and server usage. IEEE ACM Netw. **26**(4), 1562–1576 (2018)
13. Hirwe, A., Kataoka, K.: Lightchain: A lightweight optimisation of VNF placement for service chaining in NFV. In: 2016 IEEE NetSoft Conference and Workshops (NetSoft), IEEE, pp. 33–37 (2016)
14. Agarwal, S., Malandrino, F., Chiasserini, C.-F., De, S.: Joint VNF placement and CPU allocation in 5G. In: IEEE 2018 Conference on Computer Communications (INFOCOM), IEEE, pp. 1943–1951 (2018)
15. Wang, H., Schmitt, J.: Load balancing towards balanced delay guarantees in NFV/SDN. In: 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV/SDN), IEEE, pp. 240–245 (2016)
16. Bhamare, D., Samaka, M., Erbad, A., Jain, R., Gupta, L., Chan, H.A.: Optimal virtual network function placement in multi-cloud service function chaining architecture. Comput. Commun. **102**, 1–16 (2017)
17. Martini, B., Paganelli, F., Cappanera, P., Turchi, S., Castoldi, P.: Latency–aware composition of virtual functions in 5G. In: Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), IEEE, pp. 1–6 (2015)
18. Cziva, R., Anagnostopoulos, C., Pezaros, D. P.: Dynamic, latency-optimal VNF placement at the network edge. In: 2018-IEEE Conference on Computer Communications (INFOCOM), IEEE, pp. 693–701 (2018)
19. Pham, C., Tran, N.H., Ren, S., Saad, W., Hong, C.S.: Traffic-aware and energy-efficient VNF placement for service chaining: Joint sampling and matching approach. IEEE T. SERV. COMPUT. (2017)
20. Cohen, R., Lewin-Eytan, L., Naor, J. S., Raz, D.: Near optimal placement of virtual network functions. In: 2015 IEEE Conference on Computer Communications (NFOCOM), IEEE, pp. 1346–1354 (2015)
21. Wang, L., Lu, Z., Wen, X., Knopp, R., Gupta, R.: Joint optimization of service function chaining and resource allocation in network function virtualization. IEEE Access. **4**, 8084–8094 (2016)
22. Moens, H., De Turck, F.: VNF-P: A model for efficient placement of virtualized network functions. In: 10th International Conference on Network and Service Management (CNSM) and Workshop, IEEE, pp. 418–423 (2014)
23. Savi, M., Tornatore, M., Verticale, G.: Impact of processing costs on service chain placement in network functions virtualization. In: 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV/SDN), IEEE, pp. 191–197 (2015)
24. Yamato, Y.: Server selection, configuration and reconfiguration technology for IaaS cloud with multiple server types. J. Netw. Syst. Manag. **26**(2), 339–360 (2018)
25. Liu, J., Lu, W., Zhou, F., Lu, P., Zhu, Z.: On dynamic service function chain deployment and readjustment. IEEE Netw. Serv. Man. **14**(3), 543–553 (2017)
26. TinyCore/Linux. http://www.tinycorelinux.com/. Accessed on 16 Feb 2019
27. GNU/Linux. http://www.slitaz.org/. Accessed on 11 Jan 2019
28. OpenWRT. https://openwrt.org/. Accessed on 20 Feb 2019
29. Kaehler, A., Bradski, G.: Learning OpenCV 3: Computer vision in C++with the OpenCV library. O'Reilly Media, Inc. (2016)
30. Bellard, F.: https://www.ffmpeg.org/. Accessed on 9 Jan 2019
31. Chinoy, B., Braun, H.W.: The National Science Foundation (NSF) network. Tech. rep., GA-A21029, SDSC (1992)

32. Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W., et al.: ONOS: Towards an open, distributed SDN OS. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, ACM, pp. 1–6 (2014)
33. Helling, S.: Home network security, Masters Thesis. Eindhoven University of Technology, Netherlands (2015)
34. Yoo, S., Jung, J., Chung, A. Y., Kim, K., Lee, J., Park, S., Lee, S. K., Lee, H. K., Kim, H.: Empowering drones' teamwork with airborne network. In: 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), IEEE, pp. 678–685 (2017)
35. QEMU. https://www.qemu.org/. Accessed on 9 Feb 2019
36. Mortimer, M.: iperf3 Documentation, release 0.1.10 (2018)

Tahira Mahboob received her B.S. degree in computer science and engineering from University of Engineering and Technology, Lahore, Pakistan, in 2007. She received her M.S., degree in computer engineering from University of Engineering and Technology, Taxila in 2012. From 2010 to 2018, she served as faculty member at Fatima Jinnah Women University, Pakistan. She is currently a Ph.D. student at the Department of Electrical and Computer Engineering at Sungkyunkwan University, Suwon, Korea. Her research interests include Software-Defined Networking (SDN), Network Function Virtualization (NFV), wireless communication networks, and deep learning.

Young Rok Jung received his B.S. degree in information and technology from Korea Nazarene University, Cheonan, Korea, in 2017. He received his M.S. degree in electrical and computer engineering from Sungkyunkwan University, Suwon, Korea in 2019. He joined MnSoft, Hyundai, Seoul, Korea, in December 2019. His research interests include Software-Defined Networking (SDN), Network Function Virtualization (NFV), and wireless communication networks.

Min Young Chung received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 1990, 1993, and 1999, respectively. From January 1999 to February 2002, he was a Senior Member of Technical Staff with the Electronics and Telecommunications Research Institute, where he was engaged in research on the development of multiprotocol label switching systems. In March 2002, he joined the Faculty of Sungkyunkwan University, Suwon, Korea, where he is currently a Professor with the Department of Electrical and Computer Engineering. His research interests include D2D Communications, Software-Defined Networking (SDN), 5G wireless communication networks, and wireless energy harvesting. He worked as an editor on the Journal of Communications and Networks from January 2005 to February 2011, and is a member of IEEE, IEICE, KICS, KIPS, and KISS.

Affiliations

Tahira Mahboob[1] · **Young Rok Jung**[2] · **Min Young Chung**[1]

Tahira Mahboob
tahira@skku.edu

Young Rok Jung
youngrok.jung@hyundai-mnsoft.com

[1] Department of Electrical and Computer Engineering, Sungkyunkwan University, 2066, Seobu-ro, Jangan-gu, Suwon-si, Gyeonggi-do 16419, Republic of Korea

[2] Hyundai MnSoft, 74 Wonhyoro, Seoul, Yongsan-gu 04365, Republic of Korea