Online Revenue Maximization in NFV-Enabled SDNs

what is online revenue for NFV?

Yu Ma†, Weifa Liang†, and Zichuan Xu‡

† The Australian National University, Canberra, ACT 2601, Australia
‡ Dalian University of Technology, Dalian, 116024, China

***Abstract*—Traditional networks employ expensive dedicated hardware devices as middleboxes to implement Service Chains (SC) of user requests by steering data traffic along the middleboxes in the service chains. Network Function Virtualization (NFV) is a promising virtualization technique by implementing network functions as pieces of software in servers or data centers. By leveraging the technique of Software Defined Networking (SDN), NFV can be further enabled in a flexible and dynamic way. In this paper, we consider dynamic admissions of delay-aware NFV-enabled requests in SDNs by leveraging pre-installed NFV instances in data centers. We first formulate a novel revenue maximization problem, and show that the problem is NP-hard. We then propose an online heuristic for the problem. We also devise an online algorithm with a provable competitive ratio for a special case of the problem where the end-to-end delay requirement of requests can be neglected. We finally evaluate the performance of the proposed algorithms through experimental simulations. The simulation results demonstrate that the proposed algorithms are promising.**

I. Introduction

Provisioning network services in traditional networks takes a great deal of time and effort, because dedicated hardware devices in the networks need to be acquired and configured manually in an inefficient manner. Also, the management of network services provided by the dedicated hardware devices is usually error-prone, since each service requires a dedicated hardware device that needs to be individually configured with its own command syntax [14]. Network Function Virtualization (NFV) and Software Defined Networking (SDN) have been envisaged as the next-generation networking paradigm to enable fast service deployment, inexpensive and error-free service provisioning in future communication networks [2]. Specifically, SDN introduces the concept of separation between the data plane and the control plane, thus provides more flexibility in steering network flow, while NFV enables highly optimized packet-processing network functions to run on commodity servers. Typically, each flow goes through *a sequence of network functions* in a specified order to perform necessary packet processing, such a sequence is termed as *the service chain* (SC) of the flow [12]. The integration of SDN and NFV enables flexible allocations of service chains and optimizes resource utilization for user request routing [14].

The adoption of SDN and NFV technologies poses several fundamental challenges for network service providers to offer NFV-enabled services in their SDNs. The first challenge is how to maximize their revenues by admitting as many requests as possible considering the constraints of computing and bandwidth resources of the network. This can be achieved by pre-installing instances of service chains in data centers and allowing user requests share the installed instances among NFVs of future user requests. User requests not only have different amounts of resource demands but also have different service chains and delay requirements. How to meet such diverse resource demands and delay requirements such that the revenue of a network service provider is maximized is challenging. The second challenge is how to find routing paths that include data centers to process the traffic of user requests. The last challenge is how to dynamically admit user requests, assuming that the requests arrive one by one without the knowledge of future arrivals. Tackling these challenges requires novel frameworks, algorithms, and technologies, as conventional routing algorithms, protocols designed for traditional networks are not applicable to NFV-enabled SDNs.

There have been several studies on NFV-enabled user request routing [7], [9], [10], [11], [13], [15]. For example, Martins *et al.* [11] introduced a virtualization platform to improve network performance, by extending existing virtualization technologies to support the deployment of modular virtual middleboxes on lightweight VMs. Lukovszki and Schmid [10] considered NFV-enabled request admissions under an ideal assumption that each server in a network can only accommodate one VNF and the VNFs of different network functions consume the same amount of computing resource. They devised an online algorithm with competitive ratio $O(\log l)$ where l is the maximum number of VNFs in any NFV-enabled request. Li *et al.* [9] investigated QoS-aware NFV-enabled user request routing problem by implementing the VNFs of the service chain of each request in a single data center. However, most of these studies considered only the computing capacity of nodes [10], and neglected the delay requirement of requests. Huang *et al.* [4], [5] considered maximizing the network throughput problem by consolidating all VNFs of a service chain into a server, through seeking non-trivial tradeoff between the accuracy of a solution and the running time needed to obtain the solution. Xu *et al.* [15] considered a throughput maximization problem by making use of existing VNF instances, assuming that all requests are given in advance. In addition, most studies focus on delivering exact solutions by Integer Linear Programming (ILP) or heuristic solutions. These solutions are neither scalable nor any performance guarantees. To the best of our knowledge, we are the first to study the revenue maximization problem of dynamically admitting NFV-enabled requests in SDNs, by making use of pre-installed VNF instances, and we devise very first online algorithms with performance guarantees for the problem.

The main contributions of this paper are as follows. We first formulate the online revenue maximization problem of dynamically admitting delay-aware NFV-enabled requests in SDNs, assuming that each request is admitted on a pay-as-you-go basis, subject to various network resource capacity constraints, and show the problem is NP-hard. We then propose

978-1-5386-3180-5/18/$31.00 ©2018 IEEE

an online heuristic for the problem. We also devise an online algorithm with a provable competitive ratio for a special case of the problem where the end-to-end delay requirement of each request can be neglected. We finally conduct empirical evaluation on the performance of the proposed algorithms through experimental simulations.

The rest of the paper is organized as follows. Section II introduces notions, notations, and the problem definition. Section III formulates an ILP solution to an offline version of the problem. Section IV develops a heuristic for the problem, and Section V devises an online algorithm with a provable competitive ratio for a special case of the problem where the end-to-end delay requirement of each request can be neglected. Section VI evaluates the performance of the proposed algorithms empirically, and Section VII concludes the paper.

II. Preliminaries

We consider a Software Defined Network (SDN) as a directed graph $G = (\mathcal{SW} \cup V, E)$, where $\mathcal{SW}$ is a set of SDN-enabled *switch nodes* and V is a set of *data centers*, and E is a set of *links* between SDN-enabled switches and between SDN-enabled switches and data centers. The implementation of virtualized network functions (VNFs) in data centers consumes computing resource. Denote by C_v the computing capacity of data center $v \in V$. Also, different VNFs may have different processing delays. Furthermore, the data traffic routing of each request consumes link bandwidth and incurs data transmission delays along its routing path. Denote by B_e the bandwidth capacity of a link $e \in E$.

Consider a *user request* $r_j = (s_j, t_j; SC^{(k)}, b_j, \rho_j, D_j)$ that its data traffic will transfer from source s_j to destination t_j with a given packet rate $\rho_j > 0$, and the data traffic must pass through the instances of VNFs in its service chain $SC^{(k)}$. We assume that the service chains among all requests can be classified into K types. We also assume that each instance of a VNF can process a basic packet rate [15], and a request r_j with ρ_j packet rate implies it needs ρ_j instances of its required type of service chain and the amount b_j ($= \rho_j \cdot b^{(k)}$) of bandwidth resource, where $b^{(k)}$ is the bandwidth requirement of a basic data packet rate of the request with $SC^{(k)}$. We assume that some service chains have been instantiated in data centers already. We further assume that the data traffic of each request will be processed in a single data center. Since different requests may have different types of service chains, establishing a routing path for each request needs steering its data traffic flow to pass through the specified VNFs in its service chain. An ordered sequence of VNFs is defined as a *service chain* (SC). Denote by $SC^{(k)}$ the service chain of type k and $C(SC^{(k)})$ the computing resource consumption of an instance of the service chain of type k with $1 \leq k \leq K$. Assume that there are $n_v^{(k)} \geq 0$ instances of type-k service chain pre-installed in data center $v \in V$. Sometimes, instances of a specified service chain in a data center may run out. If there is sufficient available computing resource in the data center, new instances for that type of service chain can be instantiated.

Each request r_j has an end-to-end delay requirement D_j that specifies the maximum duration of per data packet of the request from its source to its destination, which includes both the processing delay and the transmission delay of the packet. Denote by $d_j(SC_v^{(k)})$ the processing delay of a packet at data center v. Furthermore, traffic route for request r_j along a link $e \in E$ incurs a data transmission delay d_e. The data transmission delay along the first segment $P(s_j, v)$ of the routing path for the data traffic of request r_j from node s_j to data center v is $d_j(P(s_j, v)) = \sum_{e' \in P(s_j,v)} d_{e'}$. Similarly, the data transmission delay along the second segment $P(v, t_j)$ of the routing path is $d_j(P(v, t_j)) = \sum_{e' \in P(v,t_j)} d_{e'}$. The end-to-end delay of data transmission of request r_j along the routing path via data center v thus is $d(r_j) = d_j(P(s_j, v)) + d_j(SC_v^{(k)}) + d_j(P(v, t_j))$. To meet the end-to-end delay requirement of r_j, we have $d(r_j) \leq D_j$.

A network service provider typically charges its users by admitting their requests on a pay-as-you-go basis through adopting a common revenue collection model [13]. The revenue collected R_j by admitting a request r_j is proportional to its computing and bandwidth resource demands, where $R_j = \lambda_1 \cdot \rho_j \cdot C(SC^{(k)}) + \lambda_2 \cdot b_j$, where λ_1 and λ_2 are constant weights for computing and bandwidth selling prices. The total revenue collected by admitting a set $\mathcal{S}$ of requests is $RV_{\mathcal{S}} = \sum_{r_j \in \mathcal{S}} R_j$.

Given an SDN $G = (\mathcal{SW} \cup V, E)$, the number of instances $n_v^{(k)}$ of $SC^{(k)}$ for type k service chain pre-installed at each data center $v \in V$, let $r_1, r_2, \ldots r_j$ be the sequence of requests arrive one by one without the knowledge of future arrivals, the *online revenue maximization problem* in G is to maximize the total revenue $RV_{\mathcal{S}}$ collected by admitting a subset $\mathcal{S}$ of requests in the sequence while meeting the end-to-end delay requirement of each admitted request, subject to resource capacity constraints on both data centers and links.

Following a reduction from a well known NP-hard problem - the knapsack problem, the revenue maximization problem in G is NP-Hard. Detailed reduction is omitted due to space limitation.

III. ILP for the offline problem

In this section, we consider a revenue maximization problem as follows. Given a set of NFV-enabled requests $\mathcal{A}$, and some instances of service chains in each data center in G, the problem is to admit as many requests as possible such that the revenue collected by the network service provider is maximized, subject to network resource constraints and request end-to-end delay requirements.

We formulate an ILP solution to this offline revenue maximization problem, which will serve as the benchmark to the online revenue maximization problem. For each request $r_j \in \mathcal{A}$, $a_j^k = 1$ if r_j is a type k request; $a_j^k = 0$, otherwise. x_j is a decision variable with value 1 if request r_j is admitted and 0 otherwise, and R_j is the revenue collected if request r_j is admitted. Each request can be either assigned to pre-installed or newly-created service chain instances. We thus introduce decision variables $y_{j,v}$ and $y'_{j,v}$ for request r_j and data center $v \in V$, with $y_{j,v} = 1$ or $y'_{j,v} = 1$ if it is assigned to pre-installed or newly-created service chain instances in data center v, respectively. For clarity, denote by $\psi^+(u)$ and $\psi^-(u)$ the sets of out-going and incoming edges of a node $u \in \mathcal{SW} \cup V$, respectively. As an edge $e \in E$ can be used in both the first segment $P(s_j, v)$ and the second segment $P(v, t_j)$ of the routing path for request r_j via v, two decision variables $z_j^{sv}(e)$ and $z_j^{vt}(e)$ are adopted to distinguish between these two segments. $z_j^{sv}(e)$ is with value 1 if and only if link e belongs to the first segment $P(s_j, v)$ and $z_j^{vt}(e)$ is with value

1 if link e is within the second segment $P(v, t_j)$; otherwise, their values are 0. The proposed ILP algorithm is described as follows.

The optimization objective of this problem is:

$$maximize \quad \sum_{r_j \in \mathcal{A}} R_j \cdot x_j,$$

subject to:

$$\sum_{v \in V} y_{j,v} + y'_{j,v} = x_j, \qquad \forall r_j \in \mathcal{A} \qquad (1)$$

$$\sum_{e \in \psi^-(u)} z_j^{sv}(e) - \sum_{e \in \psi^+(u)} z_j^{sv}(e) = 0, \forall u \in \mathcal{SW} \setminus \{s_j\}, r_j \in \mathcal{A} \qquad (2)$$

$$\sum_{e \in \psi^+(u)} z_j^{vt}(e) - \sum_{e \in \psi^-(u)} z_j^{vt}(e) = 0, \forall u \in \mathcal{SW} \setminus \{t_j\}, r_j \in \mathcal{A} \qquad (3)$$

$$\sum_{e \in \psi^-(v)} z_j^{sv}(e) - \sum_{e \in \psi^+(v)} z_j^{sv}(e) = y_{j,v} + y'_{j,v}, \forall v \in V, r_j \in \mathcal{A} \qquad (4)$$

$$\sum_{e \in \psi^+(v)} z_j^{vt}(e) - \sum_{e \in \psi^-(v)} z_j^{vt}(e) = y_{j,v} + y'_{j,v}, \forall v \in V, r_j \in \mathcal{A} \qquad (5)$$

$$\sum_{e \in \psi^-(s_j)} z_j^{sv}(e) = 0, \qquad \forall r_j \in \mathcal{A} \qquad (6)$$

$$\sum_{e \in \psi^+(t_j)} z_j^{vt}(e) = 0, \qquad \forall r_j \in \mathcal{A} \qquad (7)$$

$$\sum_{e \in \psi^+(s_j)} z_j^{sv}(e) = x_j, \qquad \forall r_j \in \mathcal{A} \qquad (8)$$

$$\sum_{e \in \psi^-(t_j)} z_j^{vt}(e) = x_j, \qquad \forall r_j \in \mathcal{A} \qquad (9)$$

$$\sum_{e \in E} d_e(z_j^{sv}(e) + z_j^{vt}(e)) + \sum_{v \in V} d_j(SC_v^{(k)}) \cdot (y_{j,v} + y'_{j,v}) \le D_j, \qquad \forall r_j \in \mathcal{A} \qquad (10)$$

$$\sum_{r_j \in \mathcal{A}} b_j \cdot (z_j^{sv}(e) + z_j^{vt}(e)) \le B_e, \qquad \forall e \in E \qquad (11)$$

$$\sum_{1 \le k \le K} n_v^{(k)} C(SC^{(k)}) + \sum_{r_j \in \mathcal{A}} \sum_{1 \le k \le K} y'_{j,v} a_j^k \rho_j C(SC^{(k)}) \le C_v, \qquad \forall v \in V \qquad (12)$$

$$\sum_{r_j \in \mathcal{A}} \rho_j \cdot a_j^k \cdot y_{j,v} \le n_v^{(k)}, \qquad \forall v \in V, 1 \le k \le K \qquad (13)$$

$$z_j^{sv}(e), z_j^{vt}(e) \in \{0, 1\}, \qquad \forall e \in E, r_j \in \mathcal{A} \qquad (14)$$

$$x_j, y_{j,v}, y'_{j,v} \in \{0, 1\}, \qquad \forall v \in V, r_j \in \mathcal{A} \qquad (15)$$

Constraints (2) and (3) ensure flow conservation at any switch node except the source and sink of the flow. Constraints (4) and (5) specify whether the service chain of r_j is implemented in data center v. Constraints (6) and (7) ensure that there is no flow that has not been processed by its service chains enters the source node s_j and there is no flow that has been processed by its service chains leaves the destination node t_j. Constraints (8) and (9) enforce data flow entering and leaving the network in accordance with the admission of r_j. Constraint (10) enforces the end-to-end delay requirement. Constraints (11), (12) and (13) impose bandwidth and computing resource capacity constraints for all links and data centers in the network, respectively.

IV. A Heuristic

In this section, we deal with the online revenue maximization problem. We first consider the delay-constrained shortest path problem via a specified node, which will be used as a subroutine for the online revenue maximization problem. We then propose an efficient heuristic for the problem, and we finally devise an online algorithm with a provable competitive ratio for a special case of the problem where the end-to-end delay requirement of each request can be neglected.

A. Delay-constrained shortest path via a specified node

Given a graph $G = (\mathcal{SW} \cup V, E)$, a request r_j from a source s_j to a destination t_j with a delay D_j, and a specified node $v \in V$, the *delay-constrained shortest path via a specified node problem* in G is to find a minimum cost path (or route) from s_j to t_j that passes through the specified node v, while the route delay is no greater than D_j. This problem is NP-hard as its special case the delay-constrained shortest path problem is NP-hard [6].

Assume that the implementation of the service chain of a request r_j is assigned to the instances in data center v, and the processing delay at v is $d_j(SC_v^{(k)})$. To meet its end-to-end delay requirement D_j, its routing delay $d_j(P(s_j, v)) + d_j(P(v, t_j))$ from its source s_j to its destination t_j must be no greater than $D_j - d_j(SC_v^{(k)})$. However, finding such a minimum cost routing path for request r_j from s_j to t_j via v is challenging, and existing algorithms for the delay-constrained shortest path problem is not applicable. We instead develop a novel algorithm to solve it. Specifically, for request r_j and data center $v \in V$, we construct an auxiliary graph $G'_{j,v}$ from G, we reduce the problem in G to the delay-constrained shortest path problem in $G'_{j,v} = (V', E')$, where $V' = \{u', u \mid \forall u \in \mathcal{SW} \cup V\}$, and $E' = \{\langle u', w' \rangle, \langle u, w \rangle \mid \forall \langle u, w \rangle \in E\}$. We then merge the specified data center node v and its copy v' into a single node v, i.e., $V' = V' \setminus \{v'\}$, $E' = E' \setminus (\bigcup_{\langle u', v' \rangle \in E'} \{\langle u', v' \rangle\} \cup \bigcup_{\langle v', u' \rangle \in E'} \{\langle v', u' \rangle\}) \cup (\bigcup_{\langle u', v' \rangle \in E'} \{\langle u', v \rangle\} \cup \bigcup_{\langle v', u' \rangle \in E'} \{\langle v, u' \rangle\})$ for any $u' \in V'$.

A delay-constrained shortest path $P'(s_j, t'_j)$ in $G'_{j,v}$ from node s_j to t'_j with a delay no greater than $D_j - d_j(SC_v^{(k)})$ can then be found, by applying the approximation algorithm due to Juttner *et al.* [6]. A pseudo-routing path (route) $P(s_j, t_j; v)$ in G for request r_j via data center v can then be derived from $P'(s_j, t'_j)$, where a pseudo path (route) is a path in which nodes and links can appear multiple times.

The detailed algorithm description for finding a delay-constrained shortest path via a specified node is given in `Algorithm` 1.

Lemma 1. *(i) If there is a directed path in $G'_{j,v}$ from s_j to t'_j, it must pass through node v, i.e., v is an articulation point in $G'_{j,v}$. (ii) Any delay-constrained shortest path $P'(s_j, t'_j)$ in $G'_{j,v}$ from s_j to t'_j cannot pass through node v twice. (iii) Each edge or node in G appears at most twice in the route $P(s_j, t_j; v)$ in G which is derived from $P'(s_j, t'_j)$.*

Proof: We first show claim (i) by contradiction. Assume that there is a directed path from s_j to t'_j that does not pass

Algorithm 1 Finding a delay-constrained shortest path via node v for request r_j in G

Input: $G = (\mathcal{SW} \cup V, E)$, a data center $v \in V$, a request $r_j = (s_j, t_j; SC^{(k)}, b_j, \rho_j, D_j)$.

Output: A delay-constrained shortest path via data center v; if it does not exist, data center v cannot be used for implementing service chain for r_j without violating its delay requirement.

1: Construct the auxiliary graph $G'_{j,v}$ from G;
2: Find a delay-constrained shortest path $P'(s_j, t'_j)$ in $G'_{j,v}$, by applying the algorithm due to Juttner *et al.* [6];
3: **if** $P'(s_j, t'_j)$ in $G'_{j,v}$ exists **then**
4: $P(s_j, t_j; v)$ in G is derived from $P'(s_j, t'_j)$;
5: **else**
6: Request r_j cannot make use of data center v.

through node v. We remove v and its incident edges from $G'_{j,v}$, this will not impact the directed path since v is not on it. However, the removal of v from $G'_{j,v}$ will lead to the resulting graph disconnected and nodes s_j and $t_{j'}$ will not be in the same connected component. Then, there is not any directed path from s_j to t'_j, which forms a contradiction.

We then show claim (ii). Assume that there is such a delay-constrained shortest path $P'(s_j, t'_j)$ from s_j to t'_j that passes through v twice. Then, $P'(s_j, t'_j)$ must contain a directed cycle with v in it. If we break the cycle by removing some of its edges and nodes except v while keeping t'_j reachable from s_j. The length of the resulting path from s_j to t'_j thus is shorter than that of $P'(s_j, t'_j)$, which contradicts the fact that $P'(s_j, t'_j)$ is a shortest delay-constrained path from s_j to t'_j.

We finally show claim (iii). It is easy to see that $P'(s_j, t'_j)$ is a simple path in $G'_{j,v}$. Each edge $\langle u, w\rangle \in E$ is contained in the route $P(s_j, t_j; v)$ in G derived from $P'(s_j, t'_j)$ if $\langle u, w\rangle \in P'(s_j, t'_j)$ or $\langle u', w'\rangle \in P'(s_j, t'_j)$. Thus, each edge or node in G appears at most twice in $P(s_j, t_j; v)$. ■

Lemma 2. *Given a $G = (\mathcal{SW} \cup V, E)$, a request r_j, and a specified node $v \in V$, there is an approximation algorithm with an approximation ratio of $(1+\epsilon)$ for the delay-constrained shortest path via a specified node problem, which takes $O(|E|^2 \log^4 |E|)$ time, where ϵ is a constant with $0 < \epsilon \leq 1$.*

Proof: The approximation ratio and time complexity of `Algorithm 1` can be inferred from Juttner *et al.*[6]. Detailed proof is omitted, due to space limitation. ■

B. Online algorithm

We here propose an online algorithm for the online revenue maximization problem. We start by introducing a cost model to measure the network resource consumption, where if a specific resource has been highly utilized, it should be less used in the future, otherwise it will result in a higher cost. On the other hand, if the resource has rarely been used, it should be encouraged to use by assigning it a lower cost. Thus, when request r_j arrives, for each data center node $v \in V$, we have

$$c_v(j) = C_v(\alpha^{1-\frac{C_v(j)}{C_v}} - 1), \qquad \forall v \in V, \tag{16}$$

where α is a tuning parameter to be decided later, with $\alpha > 1$; $C_v(j)$ is the residual computing resource at data center v when request r_j arrives, with $C_v(j) = C_v(j-1) - \rho_j \cdot C(SC^{(k)})$ if r_j admitted and $C_v(0) = C_v$; $c_v(j)$ thus is the cost of using the computing resource at data center v by request r_j.

Similarly, for each link $e \in E$, the value of $c_e(j)$ is the cost of using the bandwidth resource at link e by request r_j,

$$c_e(j) = B_e(\beta^{1-\frac{B_e(j)}{B_e}} - 1), \qquad \forall e \in E, \tag{17}$$

where β is another tuning parameter, with $\beta > 1$; $B_e(j)$ is the residual bandwidth at link e when request r_j arrives, with $B_e(j) = B_e(j-1) - b_j$ if r_j admitted and $B_e(0) = B_e$.

We then define the weights of each data center v and each link e in G as *the normalized usage costs* as follows. $\omega_v(j) = c_v(j)/C_v = \alpha^{1-\frac{C_v(j)}{C_v}} - 1$, and $\omega_e(j) = c_e(j)/B_e = \beta^{1-\frac{B_e(j)}{B_e}} - 1$.

For the admission of a coming request r_j, denote by $G_j = (\mathcal{SW} \cup V, E; \omega_v(j), \omega_e(j))$ the network of G in which its data center nodes and links are assigned with the defined weights by removing those nodes and links that do not have enough residual resources to accommodate r_j. Then, for each data center $v \in V$, a delay-constrained shortest path $P(s_j, t_j; v)$ in G_j from s_j to t_j via v is found by applying `Procedure 1`. One shortest path via a data center v' finally is identified if $P(s_j, t_j; v') = \min_{v \in V}\{P(s_j, t_j; v)\}$.

To avoid admitting request r_j that consumes too much resources (or high costs), thereby undermining the performance of the SDN, the following *admission control policy* is adopted. (i) if $\omega_{v'}(j) > \sigma_1$ for the chosen data center $v' \in V$, r_j will be rejected; or (ii) if $\sum_{e' \in P(s_j, t_j; v')} \omega_{e'}(j) > \sigma_2$, r_j will be rejected, where σ_1 and σ_2 are admission control thresholds of computing and bandwidth resource, respectively, and we set $\sigma_1 = \sigma_2 = n - 1$, $n = |\mathcal{SW} \cup V|$. The algorithm is given in `Algorithm 2`.

Algorithm 2 Maximizing the total revenue by admitting a sequence of arriving NFV-enabled requests one by one

Input: An SDN $G = (\mathcal{SW} \cup V, E)$ with a set V of data centers, a sequence of requests $\mathcal{A} = \{r_j = (s_j, t_j; SC^{(k)}, b_j, \rho_j, D_j)\}$.

Output: A solution to maximize the revenue. If a request r_j is admitted, a data center v' to implement its service chain and its routing path from s_j to t_j via v' will be delivered.

1: For each request r_j, construct G_j;
2: Find the delay-constrained shortest path $P(s_j, t_j; v')$ via data center v' among all shortest paths via different data centers in V, by invoking **Algorithm** 1 for r_j via each data center $v \in V$;
3: Determine whether r_j should be accepted or not by the admission control policy.

Theorem 1. *Given an SDN $G = (\mathcal{SW} \cup V, E)$ with a set V of data centers, there is an online algorithm, `Algorithm 2`, for the online revenue maximization problem, which delivers a feasible solution for each request in $O(|E|^2 \log^4 |E| \cdot |V|)$ time.*

The time complexity analysis of `Algorithm 2` is trivial, thus is omitted.

V. Online algorithm with a competitive ratio

In this section, we consider a special case of the online revenue maximization problem where the end-to-end delay requirement can be neglected, for which an online algorithm with a provable competitive ratio will be devised through performing modifications to `Algorithm 2`. Specifically, a shortest path $P(s_j, t_j; v')$ in G_j, instead of a delay-constrained shortest path will be found. This modified algorithm is termed

as Algorithm 3, and its detailed description is omitted due to space limitation.

The rest is to analyze the performance of Algorithm 3 as follows. We first show the upper bound on the total cost of admitted requests when request r_j arrives. We then provide a lower bound on the cost of a rejected request by Algorithm 3 but admitted by the optimal offline algorithm. We finally derive the competitive ratio of Algorithm 3. Let $\gamma = b_{max}$ be the maximum bandwidth demand by any request, i.e., $\gamma = b_{max} = \max\{b_{j'} \mid 1 \le j' \le j\}$, and let δ be the maximum computing resource demand by any request, i.e., $\delta = \max\{\rho_{j'} \cdot C(SC^{(k')}) \mid 1 \le j' \le j\}$.

The upper bound on the total cost of all admitted requests at the arrival of request r_j is given by the following lemma.

Lemma 3. *Given an SDN $G = (\mathcal{SW} \cup V, E)$ with a set V of data centers, denote by $\mathcal{S}(j)$ the set of requests admitted by the online algorithm,* Algorithm *3, until the arrival of request r_j. Then, the cost sum of data center nodes and links in G are $\sum_{v \in V} c_v(j) \le \mathbb{C}(j)(\sigma_1 + n - 1)\log\alpha$, $\sum_{e \in E} c_e(j) \le \mathbb{B}(j)(\sigma_2 + n - 1)\log\beta$, respectively, provided that $\rho_{j'} \cdot C(SC^{(k')}) \le \frac{\min_{v \in V}\{C_v\}}{\log\alpha}$ and $b_{j'} \le \frac{\min_{e \in E}\{B_e\}}{\log\beta}$, with $1 \le j' \le j$, where $\mathbb{C}(j)$ and $\mathbb{B}(j)$ are the accumulated computing and bandwidth resources being occupied by the admitted requests in $\mathcal{S}(j)$ respectively, i.e., $\mathbb{C}(j) = \sum_{r_{j'} \in \mathcal{S}(j)} \rho_{j'} \cdot C(SC^{(k')})$, $\mathbb{B}(j) = \sum_{r_{j'} \in \mathcal{S}(j)} b_{j'}$.*

The detailed proof of Lemma 3 is omitted, due to limited space.

The lower bound on the cost of a rejected request by Algorithm 3 but admitted by an optimal offline algorithm is given by the following lemma.

Lemma 4. *Let $\mathcal{T}(j)$ be the set of requests that are rejected by* Algorithm *3 but admitted by an optimal algorithm OPT prior to the arrival of request r_j, and let $P_{OPT}(s_{j'}, t_{j'}; v')$ be the routing path in $G_{j'}$ found by the algorithm OPT for a request $r_{j'} \in \mathcal{T}(j)$ with $1 \le j' \le j$, and v' the data center to implement the service chain of $r_{j'}$. Then, for any request $r_{j'} \in \mathcal{T}(j)$, we have $\sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v')} \omega_e(j') + \omega_v(j') \ge \min\{\sigma_1, \sigma_2\} = n - 1$.*

The detailed proof of Lemma 4 is omitted, due to space limitation.

We finally analyze the competitive ratio of Algorithm 3.

Theorem 2. *Given an SDN $G = (\mathcal{SW} \cup V, E)$ with a set of data centers V, there is an online algorithm,* Algorithm *3, with a competitive ratio of $O(\log n)$ for the special case online revenue maximization problem where the end-to-end delay requirement of each request can be neglected, when $\alpha = 2n$ and $\beta = 2n$. The algorithm takes $O(|\mathcal{SW} \cup V|^2 \cdot |V|)$ time to admit each request, where $n = |V| + |\mathcal{SW}|$.*

Proof: Let $\mathbb{C}_{OPT}(j)$ and $\mathbb{B}_{OPT}(j)$ be the total computing resource and bandwidth occupied by the requests admitted by the algorithm OPT when request r_j arrives, and let $P_{OPT}(s_{j'}, t_{j'}; v^*)$ be the optimal routing path by the algorithm OPT for request $r_{j'} \in \mathcal{T}(j)$ and $v^* \in V$ be the data center to implement the service chain of $r_{j'}$. We have

$$
\begin{aligned}
&(n-1)(\mathbb{B}_{OPT}(j) - \mathbb{B}(j)) \\
&\le (n-1) \sum_{r_{j'} \in \mathcal{T}(j)} b_{j'} = \sum_{r_{j'} \in \mathcal{T}(j)} b_{j'}(n-1) \\
&\le \sum_{r_{j'} \in \mathcal{T}(j)} b_{j'} \Big(\sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} \beta^{1 - \frac{B_e(j')}{B_e}} + \alpha^{1 - \frac{C_{v^*}(j')}{C_{v^*}}} - 2 \Big) \\
&= \sum_{r_{j'} \in \mathcal{T}(j)} b_{j'} \Big(\sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} \frac{c_e(j')}{B_e} + \frac{c_{v^*}(j')}{C_{v^*}} \Big) \\
&\le \sum_{r_{j'} \in \mathcal{T}(j)} b_{j'} \Big(\sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} \frac{c_e(j)}{B_e} + \frac{c_{v^*}(j)}{C_{v^*}} \Big) \qquad (18) \\
&= \sum_{r_{j'} \in \mathcal{T}(j)} \Big(\sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} \frac{c_e(j) \cdot b_{j'}}{B_e} + \frac{c_{v^*}(j) \cdot b_{j'}}{C_{v^*}} \Big) \\
&= \sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} c_e(j) \frac{\sum_{r_{j'} \in \mathcal{T}(j)} b_{j'}}{B_e} + c_{v^*}(j) \frac{\sum_{r_{j'} \in \mathcal{T}(j)} b_{j'}}{C_{v^*}} \\
&\le \sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} c_e(j) + \gamma c_{v^*}(j), \text{where } \gamma = b_{max}, \qquad (19) \\
&\le \sum_{e \in E} c_e(j) + \sum_{v \in V} c_v(j) \cdot \gamma \\
&\le 2(n-1)(\gamma \cdot \mathbb{C}(j)\log\alpha + \mathbb{B}(j)\log\beta). \qquad (20)
\end{aligned}
$$

Ineq. (18) holds since the utilization ratio of each resource does not decrease and thus the weight of each node or link in G_j does not decrease with more request admissions. Ineq. (19) holds because for any data center $v \in V$, the accumulated bandwidth of all requests that are scheduled to v is no more than $C_v \cdot b_{max}$, i.e., $\sum_{r_{j'} \in \mathcal{T}(j)} \sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} b_{j'} \le C_{v^*} \cdot b_{max} \le \gamma \cdot C_{v^*}$. Meanwhile, all algorithms, including the algorithm OPT for the problem of concern, the accumulated usage of bandwidth in any link is no greater than its capacity, i.e., $\sum_{r_{j'} \in \mathcal{T}(j)} \sum_{e \in P_{OPT}(s_{j'}, t_{j'}; v^*)} b_{j'} \le B_e$. By Ineq. (20), we have $\mathbb{B}_{OPT}(j) - \mathbb{B}(j) \le 2(\gamma \cdot \mathbb{C}(j)\log\alpha + \mathbb{B}(j)\log\beta)$, and similarly, we have $\mathbb{C}_{OPT}(j) - \mathbb{C}(j) \le 2(\mathbb{C}(j)\log\alpha + \delta \cdot \mathbb{B}(j)\log\beta)$.

Recall that $\mathcal{S}(j)$ is the set of requests admitted by algorithm Algorithm 3, and $\mathcal{T}(j)$ is the set of requests rejected by Algorithm 3 but accepted by the algorithm OPT, until the arrival of request r_j. Let $RV_{\mathcal{S}(j)}$ and $RV_{\mathcal{T}(j)}$ be the total revenues for the sets of requests $\mathcal{S}(j)$ and $\mathcal{T}(j)$, respectively, i.e., $RV_{\mathcal{S}(j)} = \sum_{r_{j'} \in \mathcal{S}(j)} R_{j'}$, $RV_{\mathcal{T}(j)} = \sum_{r_{j'} \in \mathcal{T}(j)} R_{j'}$. We here abuse the notation OPT to denote the optimal online algorithm OPT and the revenue collected by it. The competitive ratio of Algorithm 3 then is

$$
\begin{aligned}
\frac{RV_{\mathcal{S}(j)}}{OPT} &\ge \frac{RV_{\mathcal{S}(j)}}{RV_{\mathcal{S}(j)} + RV_{\mathcal{T}(j)}} = \frac{1}{\frac{RV_{\mathcal{T}(j)}}{RV_{\mathcal{S}(j)}} + 1} \\
&\ge \frac{1}{\frac{2\lambda_1(\mathbb{C}(j)\log\alpha + \delta\mathbb{B}(j)\log\beta) + 2\lambda_2(\gamma\mathbb{C}(j)\log\alpha + \mathbb{B}(j)\log\beta)}{\lambda_1\mathbb{C}(j) + \lambda_2\mathbb{B}(j)} + 1} \\
&= \frac{1}{2\log 2n(1 + \frac{\lambda_1\delta + \lambda_2\gamma x}{\lambda_2 + \lambda_1 x}) + 1},
\end{aligned}
$$

where $x = \frac{\mathbb{C}(j)}{\mathbb{B}(j)}$ and when $\alpha = \beta = 2n$. The rest is to estimate the upper bound of denominator term $2\log 2n(1 + \frac{\lambda_1\delta + \lambda_2\gamma x}{\lambda_2 + \lambda_1 x}) + 1$ as follows.

Let $f(x) = \frac{\lambda_1\delta + \lambda_2\gamma x}{\lambda_2 + \lambda_1 x}, x \in (0, +\infty)$. The first derivative of $f(x)$ is $f'(x) = (\frac{\lambda_1\delta + \lambda_2\gamma x}{\lambda_2 + \lambda_1 x})' = \frac{{\lambda_2}^2\gamma - {\lambda_1}^2\delta}{(\lambda_2 + \lambda_1 x)^2}$. The term $\frac{1}{(\lambda_2 + \lambda_1 x)^2}$ always is positive. We show that function $f(x)$ is upper bounded by distinguishing into three cases.

Case 1. If ${\lambda_2}^2\gamma - {\lambda_1}^2\delta > 0$, then $f'(x) > 0$ and approaches 0 when $x \to +\infty$. Function $f(x)$ thus is an increasing function on $x \in (0, +\infty)$ and $\lim_{x\to+\infty} f(x) = \frac{\lambda_2\gamma}{\lambda_1}$.

Case 2. If ${\lambda_2}^2\gamma - {\lambda_1}^2\delta < 0$, then $f'(x) < 0$ and approaches 0 when $x \to +\infty$. Thus, $f(x)$ is a decreasing function on $x \in (0, +\infty)$ and $\lim_{x\to 0} f(x) = \frac{\lambda_1\delta}{\lambda_2}$.

Case 3. If ${\lambda_2}^2\gamma - {\lambda_1}^2\delta = 0$, $f'(x) = 0$, then $f(x) = \frac{\lambda_2\gamma}{\lambda_1} = \frac{\lambda_1\delta}{\lambda_2}$.

In summary, the upper bound of $f(x)$ is $\frac{\lambda_2\gamma}{\lambda_1}$ if ${\lambda_2}^2\gamma - {\lambda_1}^2\delta \geq 0$; or $\frac{\lambda_1\delta}{\lambda_2}$ otherwise. Therefore, $\frac{RV_{\mathcal{S}(j)}}{OPT} \geq \frac{1}{2\log 2n(1+\frac{\lambda_2\gamma}{\lambda_1})+1}$ if ${\lambda_2}^2\gamma - {\lambda_1}^2\delta \geq 0$; otherwise, $\frac{RV_{\mathcal{S}(j)}}{OPT} \geq \frac{1}{2\log 2n(1+\frac{\lambda_1\delta}{\lambda_2})+1}$. Notice that λ_i and δ are constants with $i = 1, 2$.

The time complexity analysis of `Algorithm` 3 is trivial, thus is omitted. ■

VI. Performance Evaluation

In this section, we evaluate the performance of the proposed algorithms through experimental simulations.

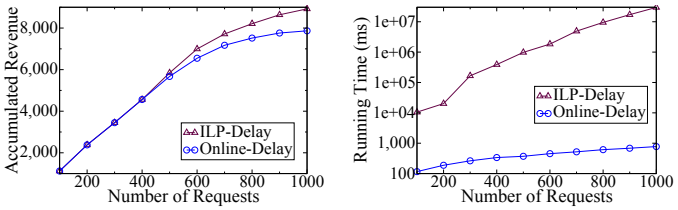
A. Experiment settings

We consider an SDN consisting of from 10 to 250 nodes. We adopt a commonly used tool GT-ITM to generate network topologies. The number of data centers in each network is set to 10% of network size, and the computing capacity of each data center varies from 3,000 GHz to 6,000 GHz [15]. The bandwidth capacity of each link varies between $2,000$ Mbps and $20,000$ Mbps [8], and the transmission delay of a link varies between $2ms$ and $5ms$ [8]. The revenue accumulation weights λ_1 and λ_2 are set as two constants drawn within $[0.05, 0.12]$ and $[0.15, 0.22]$, following typical charges in Amazon EC2 [1]. Six different types of network functions: IDS, Proxy, Load Balancer, Firewall, Gateway, and NAT are considered. The processing delay of a packet for each network function is randomly drawn from $0.045ms$ to $0.3ms$ [11]. The processing delay of a service chain instance is the sum of processing delays of its network functions. The number of service chain types K is set as 10. The bandwidth requirement of each type of service chain instance is set between 10 Mbps and 20 Mbps [3], and their computing demand is set from 1 GHz to 3 GHz [3], [11]. The number of instances of each type of service chain in a data center is randomly drawn from 10 to 50. The packet rate ρ_j of each request r_j is randomly drawn from 1 to 10 packets/second [9]. The delay D_j varies from $10ms$ to $100ms$ [11], and its type of service chain $SC^{(k)}$ is randomly picked from the K types. The running time of an algorithm is obtained based on a machine with 3.4 GHz Intel i7 Quad-core CPU and 16GB RAM.

B. Performance evaluation of online algorithms

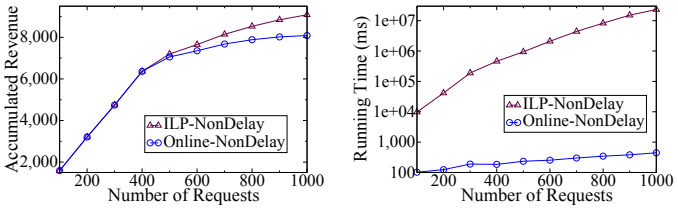
We refer to online algorithms `Algorithm` 2 and `Algorithm` 3 as `Online-Delay` and `Online-NonDelay`, respectively. We first evaluate the performance of online algorithms, `Online-Delay` and `Online-NonDelay`, against the ILP solutions for the offline revenue maximization problem with and without delay requirements that are termed as `ILP-Delay` and `ILP-NonDelay`, respectively. We then evaluate the online algorithms against two baseline heuristics: `Linear-Delay` and `Linear-NonDelay`. For each request r_j, algorithms `Linear-Delay` and `Linear-NonDelay` first remove all nodes and links from G that do not have enough residual resources to admit r_j, and then assign each data center node and each link with a weight of one. They finally find a shortest path with the minimum number of edges from s_j to t_j via a data center $v \in V$ with and without the delay requirement of the request.

We first evaluate the performance of algorithms `Online-Delay` and `Online-NonDelay` against ILP solutions by varying the number of requests from 100 to 1,000, and fixing the number of nodes in G to 50. Fig. 1 depicts the total revenues and running times of different algorithms. From Fig. 1 (a), we can see that algorithm `Online-Delay` achieves a nearly optimal revenue with at least 88.2% of the optimal one by `ILP-Delay`. For example, algorithm `Online-Delay` can obtain as much revenue as `ILP-Delay` do when there are less than 400 requests, and it achieves 88.2% revenue of that by `ILP-Delay` when there are 1,000 requests. From Fig. 1 (b), it can be seen that `ILP-Delay` takes a much longer time to deliver an optimal solution, while algorithm `Online-Delay` delivers a nearly optimal solution in much shorter time, and the running time of `ILP-Delay` is prohibitively high with the increase of the number of requests, and no solution can be achievable when the number of requests reaches 1,000. Similar results can be obtained for algorithm `Online-NonDelay` against `ILP-NonDelay` as well, which can be seen from Fig. 2 (a) and (b).



(a) The accumulated revenue (b) The running time

Fig. 1. Performance of different algorithms with network size 50 and with delay requirements.

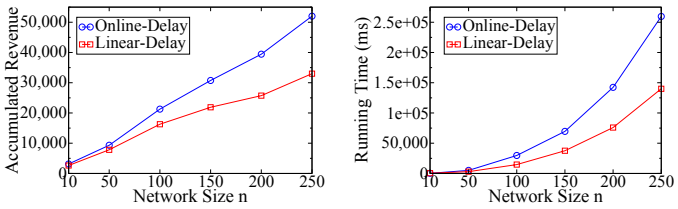


(a) The accumulated revenue (b) The running time

Fig. 2. Performance of different algorithms with network size 50 and without delay requirements.

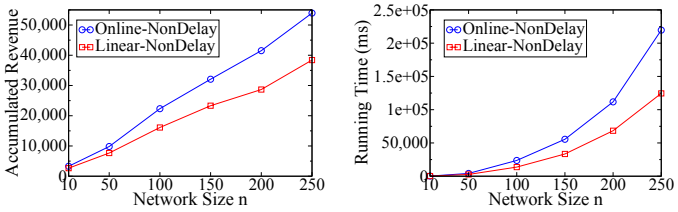
We then evaluate the performance of algorithms `Online-Delay` and `Online-NonDelay` against heuristics `Linear-Delay` and `Linear-NonDelay` by varying the number of nodes from 10 to 250 for a set of 10,000 requests, while keeping other parameters fixed, i.e., $\alpha = \beta = 2n$ and $\sigma_1 = \sigma_2 = n - 1$. Fig. 3 demonstrates the performance behaviors of different algorithms. From Fig. 3 (a) and 4 (a), we can see that both algorithms `Online-Delay` and `Online-NonDelay` outperform algorithms `Linear-Delay` and `Linear-NonDelay`.

Specifically, algorithm Online-Delay outperforms algorithm Linear-Delay, and the performance gap between them becomes larger and larger with the increase in network size. The similar performance behaviors can be observed by algorithm Online-Delay against algorithm Linear-NonDelay too, see Fig. 4 (b).



(a) The accumulated revenue (b) The running time

Fig. 3. Performance of different online algorithms by varying network size from 10 to 250 and with delay requirements.

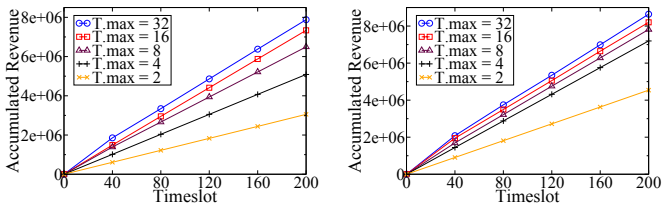


(a) The accumulated revenue (b) The running time

Fig. 4. Performance of different online algorithms by varying network size from 10 to 250 without delay requirements.

The rationale behind is that both algorithms Online-Delay and Online-NonDelay assign higher costs to over-utilized resources while assigning lower costs to under-utilized resources, thus the resources in the network can be maximally allocated to meet demands of user requests, thereby achieve higher revenue gains. However, algorithms Linear-Delay and Linear-NonDelay do not take into account the utilization of resources, thus the resource usages on some data centers and links are overloaded. Fig. 3 (b) depicts the running time curves of algorithms Online-Delay and Linear-Delay, where algorithm Linear-Delay takes less time than algorithm Online-Delay in all network sizes. The similar results on running time of algorithms Online-NonDelay and Linear-NonDelay can be seen in Fig. 4 (b).

C. Impact of parameters on the performance of algorithms



(a) The accumulated revenue by Online-Delay (b) The accumulated revenue by Online-NonDelay

Fig. 5. The accumulated revenue of online algorithms Online-Delay, and Online-NonDelay with different maximum durations of requests when the network size is 200.

We evaluate the impact of the maximum duration T_{max} of request implementations on the performance of algorithms Online-Delay and Online-NonDelay in a network of 200 nodes, by varying T_{max} from 2 to 32. Assume that the finite time horizon is slotted into equal time slots. The request admissions and resource releases proceed in the beginning of each time slot. The duration of each request implementation is randomly drawn in $[1, T_{max}]$. The revenue collected by admitting a request r_j is the duration of request implementation in a data center times its revenue R_j in a single time slot. We consider a time horizon consisting of 200 time slots with up to 1,000 requests per time slot. From Fig. 5 (a), it can be seen that the longer the maximum duration, the more revenue collected by algorithm Online-Delay. For each request, the longer its implementation in a data center, the more revenue collected by admitting it. The similar result can be obtained for algorithm Online-NonDelay from Fig. 5 (b) as well.

VII. Conclusion

In this paper, we investigated the online revenue maximization problem in SDNs by leveraging pre-installed VNF instances. We first proposed an ILP solution for offline version of the problem. We then develop an efficient heuristic for the problem. We also devised an online algorithm with a provable competitive ratio for a special case of the problem. We finally evaluated the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising.

Acknowledgments

The research by Zichuan Xu is partially supported by the Fundamental Research Funds for the Central Universities under Grant No.: DUT17RC(3)061.

References

[1] Amazon Web Services, Inc. Amazon ec2 instance configuration. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-ug.pdf.

[2] N. Chowdhury, and R. Boutaba. Network virtualization: state of the art and research challenges. *IEEE Commu. Magazine*, pp. 20–26, 2009.

[3] M. Ghaznavi, N. Shahriar, R. Ahmed, and R. Boutaba. Service Function Chaining Simplified. *CoRR*, vol. abs/1601.00751, Feb. 2016.

[4] M. Huang, W. Liang, Z. Xu, M. Jia, and S. Guo. Throughput maximization in software-defined networks with consolidated middleboxes. *Proc. of LCN*, IEEE, 2016.

[5] M. Huang *et al.* Efficient algorithms for throughput maximization in software-defined networks with consolidated middleboxes. To appear in *IEEE Transactions on Network and Service Management*, accepted on July 6, 2017.

[6] A. Juttner *et al.* Lagrange relaxation based method for the QoS routing problem. *Proc. of INFOCOM*, IEEE, 2001.

[7] T-W. Kuo *et al.* Deploying chains of virtual network functions: on the relation between link and server usage. *Proc. of INFOCOM*, IEEE, 2016.

[8] S. Knight *et al.* The internet topology zoo. *Journal of Selected Areas in Communications*, vol. 29, pp. 1765–1775, IEEE, 2011.

[9] Y. Li, L. T. X. Phan, and B. T. Loo. Network functions virtualization with soft real-time guarantees. *Proc. of INFOCOM*, IEEE, 2016.

[10] T. Lukovszki and S. Schmid. Online admission control and embedding of service chains, *Proc. of International Colloquium on Structural Information and Communication Complexity*, Springer, 2014.

[11] J. Martins et. al. ClickOS and the art of network function virtualization. *Proc. of NSDI*, USENIX, 2014.

[12] P. Quinn, and T. Nadeau. Service Function Chaining Problem Statement. *IETF Secretariat*, 05 Apr. 2014.

[13] W. Racheg *et al.* Profit-driven resource provisioning in NFV-based environments. *Proc. of ICC*, IEEE, May, 2017.

[14] S. V. Rossem *et al.* Deploying elastic routing capability in an SDN/NFV-enabled environment. *2015 IEEE Conference on NFV-SDN*, 2015.

[15] Z. Xu *et al.* Throughput maximization and resource optimization in NFV-enabled networks. *Proc. of ICC*, IEEE, 2017.

[16] Z. Xu *et al.* Approximation and online algorithms for NFV-enabled multicasting in SDNs. *Proc. of ICDCS*, IEEE, 2017.