Routing Cost Minimization and Throughput Maximization of NFV-Enabled Unicasting in Software-Defined Networks

Mike Jia, Weifa Liang, *Senior Member, IEEE,* Meitain Huang, Zichuan Xu, *Member, IEEE*, and Yu Ma

***Abstract*—Data transfer in contemporary networks usually is associated with strict policy enforcement for data transfer security and system performance purposes. Such a policy is represented by a service chain consisting of a sequence of network functions such as firewalls, intrusion detection systems, transcoders, etc. Due to the high cost and inflexibility of managing hardware-based network functions, network function virtualization (NFV) has emerged as a promising technology to meet the stringent requirement imposed on the service chain of each data transfer request in a low-cost and flexible way. In this paper, we study policy-aware unicast request admissions with and without end-to-end delay constraints in a software defined network. We aim to minimize the operational cost of admitting a single request in terms of both computing resource consumption for implementing the NFVs in the service chain and bandwidth resource consumption for routing its data traffic, or maximize the network throughput for a sequence of requests without the knowledge of future request arrivals. We first formulate four novel optimization problems and provide a generic optimization framework for the problems. We then develop efficient algorithms for the admission of a single NFV-enabled request with and without the end-to-end delay constraint, where NFV-enabled requests are defined as the requests with policy enforcement requirements. We also devise online algorithms with a guaranteed performance for dynamic admissions of requests without the knowledge of future arrivals. In particular, we provide the very first online algorithm with a provable competitive ratio for the problem without the end-to-end delay requirement. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results show that the proposed algorithms are promising and outperform existing heuristics.**

***Index Terms*—Policy-aware unicasting, operational cost minimization, online algorithms, service chains, software defined networks, network function virtualization, algorithm design and analysis.**

I. Introduction

TO ENSURE data transfer security, system performance, and data integrity, modern communication networks rely on a wide spectrum of hardware middleboxes such as firewalls, intrusion detection systems, and traffic compression. Data transfer in such networks often have associated policy enforcement requirements, which are represented as a specified sequence of network functions such as firewalls, intrusion detection systems (IDSs), deep packet inspection (DPI), and so on. Network Function Virtualization (NFV) that implements network functions as software running in a virtual machine has emerged as a promising technique to meet network policy enforcement requirements introduces a new dimension for cost savings on hardware, and enables flexible, faster deployments of new network functions. As network functions can be agilely and dynamically initiated and reused by different users, this allows network operators to meet users requirements in a cost-effective way. Along with the NFV technology, Software-Defined Networking (SDN) has also been envisioned as another disruptive technology for network virtualization, which separates the network control plane from the data plane. SDN offers easy network management and simplified network configurations. In particular, data transfers can be performed efficiently and effectively through automatic route rearrangements, allowing network operators to dynamically adjust the route of traffic. Considering that unicasting is a primitive functionality of networks, in this paper we study the admissions of unicast requests with policy-enforcement in an SDN, by implementing the policies through virtual machines in data centers, and we refer to such a request as *an NFV-enabled request*.

Admitting NFV-enabled requests in an SDN poses great challenges. First, for each NFV-enabled request, we must determine not only a routing path for its data traffic but also which data centers to be included in the routing path. Second, since NFV-enabled requests arrive into the system dynamically and unpredictably, the response to each incoming request by either admitting or rejecting it is crucial in order to maximize the network throughput. If a request is admitted, a routing path and a set of data centers on the path should be found for the request immediately. The dynamic nature of resource allocation in SDNs and unpredictability of future request arrivals further increases the difficulty in tackling this dynamic request admission problem.

Several efforts on implementing NFV-enabled requests in SDNs have been taken recently [14], [15], [21]–[23]. Most of these studies however either considered computing resource at nodes [23], assuming that only one VM is associated to a

Manuscript received September 24, 2017; revised December 26, 2017; accepted February 25, 2018. Date of publication March 1, 2018; date of current version June 8, 2018. The associate editor coordinating the review of this paper and approving it for publication was G. Casale. *(Corresponding author: Weifa Liang.)*

M. Jia, W. Liang, M. Huang, and Y. Ma are with the Research School of Computer Science, Australian National University, Canberra, ACT 2601, Australia (e-mail: u5515287@anu.edu.au; wliang@cs.anu.edu.au; u4700480@anu.edu.au; u5108648@anu.edu.au).

Z. Xu is with the School of Software, Dalian University of Technology, Dalian 116024, China (e-mail: z.xu@dlut.edu.cn).

Digital Object Identifier 10.1109/TNSM.2018.2810817

service chain of each request [3], [14], [15], [22] by consolidating all network functions of the service chain into a single data center (server), or found a cost-inefficient routing path (rather than a routing walk) in which a data center (server) or link appears only once [20], [21].

We distinguish our work in this paper from the aforementioned work as follows. (i) We assume that the VMs for different network functions of the service chain of each NFV-enabled request can be implemented at different data centers (servers), rather in a single data center; and (ii) we not only make use of existing VMs but also create new VMs for implementing network functions in data centers, depending on which option leads to further cost savings. It must be mentioned that this is an extended version of a conference paper in [16]. The main difference between this extended version and its conference counterpart are listed as follows. (1) We provide a general optimization framework for NFV-enabled request optimization problems. (2) We consider the operational cost minimization problem for a single request admission by exploring the sharing of VNF instances of different network functions among different requests. (3) In addition to developing online algorithms for dynamic NFV-enabled request admissions in the conference version, we also conducted a detailed analysis on the competitive ratio of the proposed online algorithm for the problem without the end-to-end delay constraint. (4) In the performance evaluation, we conducted extensive experiments to evaluate the performances of all proposed algorithms against the benchmarks. We also investigate the impact of important parameters on the performance of the proposed algorithms.

To the best of our knowledge, we provide the very first generic optimization framework for NFV-enabled unicasting with and without end-to-end delay constraints, by implementing the service chain of each NFV-enabled request in multiple data centers to either minimize the implementation cost of the request, or maximize the network throughput of a sequence of NFV-enabled requests without the knowledge of future request arrivals.

The main contributions of this paper are as follows.

- We first formulate novel optimization problems for NFV-enabled unicasting in SDNs, and propose a generic optimization framework for the admissions of NFV-enabled requests with and without end-to-end delay constraints.
- We then devise efficient algorithms for minimizing the implementation cost of each NFV-enabled request in terms of computing and bandwidth resource consumptions.
- We also deal with the admissions of dynamic NFV-enabled requests without the knowledge of future arrivals, and devise the very first online algorithm with a provable competitive ratio for the problem without the end-to-end delay constraint; otherwise, we develop a fast online algorithm.
- We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms are promising and outperform existing heuristics.

The rest of the paper is organized as follows. Section II will review existing studies. Section III will introduce the system model and problem definitions. Section IV will propose a novel optimization framework for NFV-enabled unicasting. Section V will devise efficient algorithms that admit a single NFV-enabled request with and without the end-to-end delay constraint. Section VI will develop online algorithms for dynamic admissions of a sequence of NFV-enabled requests without the knowledge of future arrivals. Section VII will provide experimental results on the performance of the proposed algorithms, and Section VIII concludes the paper.

II. Related Work

There are recent studies on implementing NFV-enabled unicasting in SDNs. For example, under an ideal model with the assumption that each data center (server) can accommodate only one VM and different VMs for different network functions have the same amount of computing resource, Lukovszki and Schmid [23] showed that for a request with a service chain of length 3, finding a minimum cost routing walk for the request in an SDN is NP-hard, and there is no approximate solution with a constant approximation ratio for it, i.e., the problem is APX-hard, where *a walk* may not be a simple path, in which a node and/or an edge can appear multiple times. Furthermore, they provided an $O(\log l)$-competitive online algorithm under the assumption that each node capacity are at least logarithmic in l, where l is the maximum length of any NFV-enabled request. However, they only considered the computing cost at servers without taking the bandwidth demand of each request into consideration when identifying a routing walk for the request. Lukovszki *et al.* [22] recently considered middlebox placements in a n-node network so that each source-destination pair in a set has a path of length at most L with one middlebox in it. And each middlebox can be used by at most k pairs. They devised an approximation algorithm with an approximation ratio of $O(\log \min\{n, k\})$ for the problem, assuming that only one VM or one network function associated with each request and each server can accommodate no more than k VMs. This assumption however is over simplified as the length of a service chain of a request may be far larger than one. Huang *et al.* [14], [15] considered consolidating all NFVs of a service chain into a single VM and placed the VM into one of the data centers in the network. Li *et al.* [21] considered requests from a single tenant with the end-to-end delay constraint in a data center, for which they considered routing the packets from the tenant by partitioning the service chain into different server racks in a data center, and provided a solution by adopting the dynamic programming strategy. Kuo *et al.* [20] studied the NFV-enabled request routing problem in an SDN by fully utilizing existing VMs for network functions at different servers attached to SDN-enabled switches. They developed a heuristic algorithm by incorporating the bandwidth requirement into consideration. They also made use of the dynamic programming strategy. However, there is an important assumption in [20] and [21]. That is, they find a routing path, rather than a routing walk,

for each request. This implies that a server and a link in the routing path can only appear once, which may not be cost-effective in practice if a server contains the VMs of multiple NFVs that are not consecutive in the service chain of the request. Clearly, significant costs could potentially be saved by visiting the server multiple times. In fact, realizing an NFV-enabled request in an SDN with the minimum implementation cost is to strive for a fine tradeoff between the computing resource usage and the bandwidth resource usage in the SDN. Sometimes, it is very costly to make use of an existing VM of an NFV that is far from the source and the destination of a request. Instead, creating a new VM for the NFV in a server nearby the source and/or the destination of the request can be much cheaper, as far less bandwidth will be consumed on its routing path. Xu *et al.* [29] recently generalized the online NFV-enabled unicasting problem to the online NFV-enabled multicasting problem in SDNs. They proposed a novel online algorithm with performance guarantee for online NFV-enabled multicasting. Eramo *et al.* [6] studied the offline and online throughput maximization problems in NFV-enabled networks, by proposing efficient heuristics. Bari *et al.* [3] investigated the virtualized network function orchestration problem, by determining the required number and placement of VNFs that optimizes network operational costs and utilization, without violating service level agreements. An integer linear programming (ILP) based solution is proposed for small problem sizes, and a dynamic programming-based heuristic is used for large network sizes.

There are studies on virtual network embedding (VNE) that aim to embed a set of virtual networks with computing resource demand on nodes and network bandwidth demand on links to a substrate network with node and link resource capacities [8], [30], [31], this problem is termed as *the Virtual Network Embedding problem* (or VNE for short). At first sight, the problem in this paper is similar to the VNE problem. However, the problem of NFV-enabled unicasting is essentially different from the VNE problem, due to the following reasons. First, the VNE problem deals with a virtual network denoted as a graph, whereas the service chain of each request is a sequence of network functions denoted as a chain. Second, in the VNE problem, resources allocated to a user are exclusive to that user whereas allocated resources in the admission of NFV-enabled requests can be shared by multiple users. Specifically, different virtual networks do not share computing and bandwidth resources with the other virtual networks embedded in the same substrate network, while the VNF placement of different network functions service chains is to explore the VNF instance sharing among different requests or creating new VNFs for some requests. Third, there are no node ordering requirements in the VNE problem; the traffic of a service chain however is required to pass through the network functions in the specified order of the service chain.

III. Preliminaries

In this section, we first introduce the system model and notations, and then define the problems precisely.

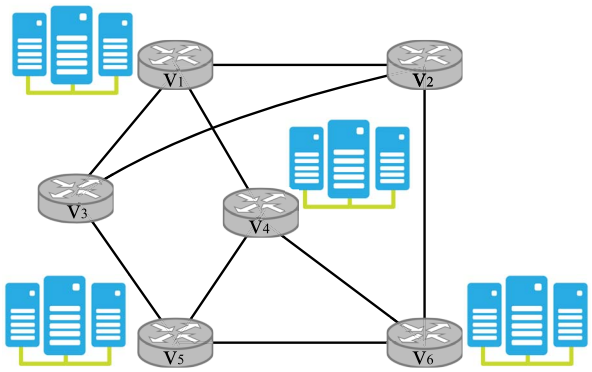


Fig. 1. A software-defined network G with a set $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ of SDN-enabled switch nodes and a subset $V_S = \{v_1, v_4, v_5, v_6\}$ ($V_S \subseteq V$) of switch nodes attached with data centers.

A. System Model

We consider a software-defined network $G = (V, E)$ with a set V of SDN-enabled switch nodes and a set E of links between SDN-enabled switch nodes. To implement various network functions as Virtual Machines (VMs), some of the switch nodes are attached with data centers that host the VMs for the network functions. Without loss of generality, we assume that implementing network functions at data centers incur costs and delays. Similarly, data transfers at each link $e \in E$ incur communication costs and transmission delays. We further assume that the communication delay and the cost between a switch node and the data center attached to it are negligible in comparison with the communication delay and the cost of other nodes in the network, as they are connected by a high-speed optical fiber. We thus denote by V_S ($\subseteq V$) the subset of switch nodes attached with data centers. Notice that each node $v \in V_S$ is treated like a switch node without an attached data center if its data center will not be used for implementing network functions of an NFV-enabled request. Otherwise, the processing cost and delay of implementing a VM at the data center attached to v must be taken into account. Following the same model as [2] and [26] and the observation that additional computing and storage resources can be added to the network by purchasing and installing new hardware in data centers, whereas adding additional network bandwidth resources via building communication cables that connect data centers in different cities or continents is beyond the capacity of most network service providers, we primarily deal with bandwidth capacity of links in this paper. Denote by B_e the bandwidth capacity of a link $e \in E$ in G. In this paper, we assume that each data center in V_S has rich computing and storage resources for implementing network functions. There is an SDN controller in G that controls both the computing and bandwidth resources of G and performs resource allocations to meet the resource demands of each admitted NFV-enabled request, where an NFV-enabled request is admitted if its demanded resources can be met. Figure 1 is an example of an SDN, where switch nodes v_1, v_4, v_5, and v_6 are attached with data centers, while the rest of its nodes are not.

We here consider *an NFV-enabled request* that transfers its data from a source to a destination such that the traffic passes through a sequence of network functions in a specified order. Such a request requires not only bandwidth resource demands to transfer data traffic but also computing resource demands to implement its sequence of network functions in VMs.

Consider the kth request $\rho_k = (s_k, t_k; SC_k, b_k)$ with bandwidth requirement b_k and *the service chain* SC_k, where $SC_k = \langle SC_{k,1}, SC_{k,2}, \ldots, SC_{k,l} \rangle$ is a sequence of network functions for which each packet from its source s_k to its destination t_k must pass through one by one in order. Recall that some switches are attached with data centers, while others are not. If a switch is not attached to any data center, it only serves as a routing switch; otherwise, the switch can serve two roles: one serves as a routing switch; another serves not only as a routing switch but also as its attached data center for implementing some (or all) of the network functions in SC_k. If a switch has this latter role, the switch and its attached data center can be interchangeably used in the rest of this paper. Furthermore, if there is a VM (or an instance) for implementing a specific network function in a data center, then the VM will be used; otherwise (if no such VM exists), a new VM can be created in the data center if needed. Some users may impose stringent end-to-end delay constraints D_k on their requests ρ_k while others do not. We will consider both cases in this paper.

B. Problem Definitions

Given a software-defined network $G = (V, E)$, a subset $V_S \subset V$ of switches attached with data centers each of which has unlimited computing resource, and a request $\rho_k = (s_k, t_k; b_k, SC_k)$, we assume that G has sufficient computing resource to meet the computing resource demands of any NFV-enabled request while the bandwidth capacity at each link is limited. We further assume that the network operator of G charges each admitted request on a pay-as-you-go basis, and it focuses on minimizing its *operational cost* which is the sum of computing and bandwidth resource consumption costs for implementing the requests. Assuming that c_e and c_v represent the costs of using one unit of bandwidth and computing resources at link $e \in E$ and data center $v \in V_S$, respectively, we define the following four optimization problems in the SDN G.

Definition 1: The NFV-enabled unicasting problem in an SDN $G = (V, E)$ with a set V_S of data centers for an NFV-enabled request ρ_k $(= (s_k, t_k; SC_k, b_k))$ is to find a routing walk for ρ_k such that its implementation cost, in terms of both computing and bandwidth resource consumption, is minimized, subject to the bandwidth capacity constraint at each link in G, assuming that computing resource at data centers are unlimited.

Definition 2: The delay-aware NFV-enabled unicasting problem in an SDN $G = (V, E)$ with a set V_S of data centers for an NFV-enabled request ρ_k $(= (s_k, t_k; SC_k, b_k, D_k))$ is to find a routing walk for ρ_k such that its implementation cost is minimized while the end-to-end delay of the walk is no greater than a given end-to-end delay constraint D_k, subject to the bandwidth capacity constraint at each link in G, assuming that the computing resource at data centers are unlimited.

Definition 3: The online NFV-enabled unicasting problem in an SDN $G = (V, E)$ with a set V_S of data centers is to admit as many NFV-enabled requests as possible without the knowledge of future request arrivals, subject to the bandwidth capacity constraint at each link in G, assuming that the computing resource at data centers are unlimited.

Definition 4: The *online delay-aware NFV-enabled unicasting problem* in an SDN $G = (V, E)$ with a set V_S of data centers is to admit as many NFV-enabled requests as possible without the knowledge of future request arrivals, while meeting the end-to-end delay requirement of each admitted request, subject to the bandwidth capacity constraint at each link in G, assuming that the computing resource at data centers are unlimited.

Notice that both the delay-aware NFV-enabled unicasting problem and the online delay-aware NFV-enabled unicasting problem are both NP-hard, as the well known delay-constrained shortest path problem is NP-hard [17], which is a special case of these two problems where each node has an attached data center. Even if G is a directed acyclic graph, both the problems are still NP-hard [28].

IV. A Generic Optimization Framework

In this section, we propose a generic framework for solving the four mentioned optimization problems.

A. Overview

There are two important issues for solving the problems. One is the resource availability. Given an NFV-enabled request ρ_k, whether it can be admitted or rejected will be determined by the availability of its demanded resources in G. Another is which data centers should be used to implement its service chain, considering that the network functions in the service chain can be implemented in different data centers. These two issues are critical for delivering efficient and high-quality solutions to the problems, since careless admissions of requests can significantly increase their implementation costs, violate their end-to-end delay requirements, and heavily affect the admissions of future requests. We address these issues by proposing a novel optimization framework that efficiently reduces each of the four problems into a problem of finding a (delay-constrained) shortest path in an auxiliary acyclic graph $H_k = (N_k, A_k)$ for each incoming request ρ_k. Specifically, if there is no such path in H_k, this implies there are insufficient resources in G to meet the demands of the request, and the request should be rejected. Otherwise, a routing walk (sometimes a path) for ρ_k then will be derived from the found (delay-constrained) shortest path H_k. The construction of H_k is as follows.

B. The Construction of the Auxiliary Directed Acyclic Graph H_k for Request ρ_k

Recall that there may have multiple candidate data centers that have the VMs for the jth network function $SC_{k,j}$ of the service chain SC_k or the VM that can be created from the data centers. For clarity, let V_j be the set of data centers that can implement $SC_{k,j}$. The node set N_k of H_k consists of all such

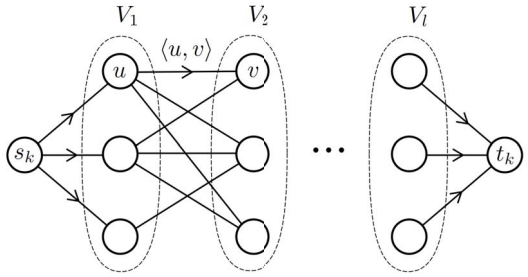


Fig. 2. A constructed auxiliary graph H_k, where $V_1, \ldots, V_l$ represent the sets of candidate nodes for each service layer in the service chain.

sets of data centers and the source, destination of request ρ_k, i.e., $N_k = \cup_{j=1}^{l} V_j \cup \{s_k, t_k\}$. To guarantee network functions of SC_k are traversed in its specified order, we connect nodes in N_k according to the specified order. Specifically, we first add a directed edge from s_k to each node $v \in V_1$ and the weight of this edge is the cost of the shortest path in G between s_k and v if such a path exists. We then add a directed edge from a node $v \in V_l$ to t_k and assign its weight as the cost of the shortest path in G between v and t_k if the path exists. We thirdly add a directed edge from a node $u \in V_j$ to a node in $u \in V_{j+1}$ and assign its weight to be the cost of the shortest path between them in G if the shortest path exists. Notice that, if nodes u and v for implementing different network functions are in the same data center, the weight of the edge is zero. Thus, $A_k = \{\langle s_k, v\rangle \mid v \in V_1\} \cup \{\langle v, t_k\rangle \mid v \in V_l\} \cup_{j=1}^{l-1} \{\langle u, v\rangle \mid u \in V_j \ \& \ v \in V_{j+1}\}$. Figure 2 gives an example of the auxiliary graph.

C. Operational Cost and Transmission Delay Models

Realizing NFV-enabled requests in an SDN consumes its computing and bandwidth resources which incur the operational cost. To minimize the implementation cost of each request ρ_k, we strive for the fine tradeoff between the computing resource consumption at data centers for implementing the service chain of the request and the bandwidth resource consumption for routing the packets of the request along a routing path (or walk) from nodes s_k to t_k. If such a request also has an end-to-end delay requirement, we consider not only the processing delay of each VM in data centers but also the data transmission delays at links in G. Specifically, consider implementing the network function $SC_{k,j}$ of service chain SC_k in a switch node v ($\in V_S$) with an attached data center, the implementation cost $c_v(k, j)$ of $SC_{k,j}$ at v is the cost of using a VM resource for $SC_{k,j}$ if the VM already exists in v; otherwise, the implementation cost will consist of the setup cost of the VM for $SC_{k,j}$ and the cost of using the VM in v. That is, for each node $v \in V_j \subset N_k$, the cost $c_v(k, j)$ of implementing the network function $SC_{k,j}$ of service chain SC_k for all j with $1 \le j \le l$ is

$$c_v(k,j) = \begin{cases} c_v(SC_{k,j}) & \text{if there is a VM of } SC_{k,j} \text{ in } v, \\ c_v(SC_{k,j}) + In_v(SC_{k,j}) & \text{otherwise,} \end{cases} \tag{1}$$

where $In_v(SC_{k,j})$ is the setup cost of a VM at v for $SC_{k,j}$.

The processing delay $d_v(k, j)$ by running the VM for $SC_{k,j}$ at a data center $v \in V_j \subset N_k$ can be defined similarly. That is, for all j with $1 \le j \le l$,

$$d_v(k,j) = \begin{cases} d_v(SC_{k,j}) & \text{running duration of the VM of } SC_{k,j} \text{ at } v, \\ d_v(SC_{k,j}) + d_v^{in} & \text{otherwise,} \end{cases} \tag{2}$$

where d_v^{in} is the VM setup delay for $SC_{k,j}$ at node v. Notice that there usually is a queuing delay for processing each admitted request at a data center. As we assume that each incoming request must be responded by either rejecting or admitting it immediately, and each data center has enough resource to meet the request computing demands, such a queuing delay in the data center assigned to it can be ignored, compared with its processing delay in the data center.

For each directed edge $e = \langle u, v\rangle \in A_k$, the cost of admitting an NFV-enabled request ρ_k is proportional to the accumulated amount of bandwidth consumed at links by the request, i.e.,

$$c_e(k) = c_{u,v}(k) = \sum_{e' \in P_{u,v}} c_{e'} \cdot b_k, \tag{3}$$

where $P_{u,v}$ is the shortest path in G from u to v, and $c_{e'}$ is the cost of unit bandwidth at link $e' \in P_{u,v}$.

Similarly, the transmission delay $d_e(k)$ on link $e \in A_k$ is

$$d_e(k) = d_{u,v}(k) = \sum_{e' \in P_{u,v}} d_{e'} \tag{4}$$

where $P_{u,v}$ is the shortest path in G from u to v, and $d_{e'}$ is the delay on link $e' \in P_{u,v}$.

V. Algorithms for Delay-Aware NFV-Enabled Unicasting Problem

In this section we devise efficient algorithms for the NFV-enabled unicasting problem with and without the end-to-end delay constraint. We start with the problem without the end-to-end delay constraint. We then consider the delay-aware NFV-enabled unicasting problem.

A. Optimal Algorithm Without the End-to-End Delay Constraint

The basic idea behind the proposed algorithm is that the routing walk in G for a request ρ_k must pass through l VMs on the walk with each VM corresponding to a network function in SC_k in its specified order. The VM for a specific network function may already exist in a data center, or can be dynamically created in that data center or the other data centers. Thus, any data center (or a subset of data centers if there are restrictions on the use of other data centers) can be a candidate data center for the implementation of that network function.

To this end, we first construct an auxiliary directed acyclic graph $H_k = (N_k, A_k; c_v(k), c_e(k))$ for each request ρ_k, as detailed in the previous section, where costs $c_v(k, j)$ and $c_e(k, j)$ are the usage costs of implementing $SC_{k,j}$ at node $v \in V_j$ and an edge $\langle u, v\rangle \in (V_i \times V_{i+1}) \cap A_k$, which have been defined in Eqs. (1) and (3), respectively.

We then find a shortest path P' in H_k from node s_k to node t_k. If P' exists, a routing walk P in G from s_k to t_k can then

Algorithm 1 Finding a Minimum-Cost Routing Path for an NFV-Enabled Request ρ_k

Input: An SDN $G = (V, E)$ with a set V_S of data centers and installed VMs implementing network functions, a request $\rho_k = (s_k, t_k; SC_k, b_k)$ with source node s_k, destination node t_k, bandwidth resource demand b_k and the service chain $SC_k = \langle SC_{k,1}, SC_{k,2} \ldots, SC_{k,l} \rangle$.

Output: Admit or reject request ρ_k. If it is admitted, find a minimum cost routing walk in G for it, in which some existing VMs will be utilized while the VMs for other network functions in SC_k can be created at data centers on the walk.

1: Denote by $G = (V, E)$ the resulting graph after the removal of links with residual bandwidth less than $\ell \cdot b_k$ from G, assign each link in G a cost of using the amount b_k of bandwidth, where $\ell = |SC_k|$;
2: Compute all pairs shortest paths in $G(V, E)$;
3: Construct the auxiliary directed acyclic graph $H_k = (N_k, A_k; c_v(k), c_e(k))$ and assign a weight to each of its nodes and edges by Eqs.(1) and (3);
4: Find a shortest path P' in H_k from s_k to t_k in terms of the weighted sum of edges and nodes in the path;
5: If P' exists, a routing walk (or a pseudo-routing path) P in G for ρ_k is derived by replacing each edge in P' with its corresponding shortest path in G; otherwise, reject request ρ_k.

be derived, by expanding each edge in the shortest path in H_k to the set of edges in the corresponding shortest path of G. Notice that the derived walk P in G may not be a simple path, because a node and/or an edge in G can appear multiple times in P. Thus, for each incoming request ρ_k, it is either admitted or rejected, depending on the resource availabilities. If it is admitted, a routing walk P in G with the minimum operational cost will be delivered.

The detailed algorithm for NFV-enabled unicasting problem without the end-to-end constraint is given in `Algorithm` 1.

The rest is to show the correctness and analyze the time complexity of the proposed algorithm, `Algorithm` 1. We start with the following lemma.

Lemma 1: Given an SDN $G = (V, E)$, a set V_S of data centers, an NFV-enabled request ρ_k can be admitted by the system if the destination node t_k is reachable in H_k from the source node s_k.

Proof: We show the claim by contradiction. Following the construction of H_k, H_k is a layered directed acyclic graph, node s_k is at layer zero, each node $v \in V_1$ is at layer one, and node t_k at layer $\ell + 1$ if $|SC_k| = \ell$.

Let $v \in V_j$ be a reachable node in H_k from s_k and the layer number j is the maximum value. Clearly, $1 \leq j \leq l$. Otherwise, there is a directed edge in H_k from v to t_k, and t_k is reachable from s_k. As there is no directed edge in H_k starting from $v \in V_j$, this implies that there is insufficient bandwidth resource in any path in G from v to t_k, and the request ρ_k will be rejected. ■

Lemma 2: Given an SDN $G = (V, E)$ and an NFV-enabled request ρ_k with the bandwidth demand b_k, and a service chain SC with $|SC_k| = \ell$, to ensure that there is a feasible solution by `Algorithm` 1, for any admitted request ρ_k, the amount of available bandwidth at each link in G when implementing ρ_k is no less than $\ell \cdot b_k$.

Proof: The amount of available bandwidth $l \cdot b_k$ at each link for the admission of ρ_k is the most conservative estimation. To show this claim, we consider an extreme case where the SDN consists of nodes v_1, v_2, $\ldots$, v_n, and there is an edge between v_i and v_{i+1} with $1 \leq i < n$. We assume that $v_1 = s_k$ and $v_n = t_k$, and only two nodes v_i and v_{i+1} in the SDN are attached with data centers. We further assume that v_i contains the VMs of network functions $SC_{k,1}, SC_{k,3}, SC_{k,5}, \ldots, SC_{k,\ell-1}$ while node v_{i+1} contains the VMs for network functions $SC_{k,2}, SC_{k,4}, SC_{k,6}, \ldots, SC_{k,\ell}$ of the service chain SC_k of request ρ_k, and l is even. We assume that the cost of creating a new VM or migrating an existing VM between v_i and v_{i+1} is much higher than the cost of forwarding a packet to the data centers. In order to implement the service chain SC_k of ρ_k, the routing walk for request ρ_k must traverse link (v_i, v_{i+1}) at least $\ell/2$ times with each direction, and each traversal on the link consumes the amount b_k of bandwidth. Thus, the amount of bandwidth consumed for implementing SC_k in link (v_i, v_{i+1}) is at least $\ell \cdot b_k$ if the link is undirected. The lemma thus follows. ■

Theorem 1: Given an SDN $G = (V, E)$ with a set V_S of data centers ($V_S \subseteq V$), there is an algorithm, `Algorithm` 1, for the NFV-enabled unicasting problem, which takes $O(|V|^3)$ time and delivers an optimal solution for the problem.

Proof: Following Lemma 1, if s_k is reachable in H_k from s_k, then request ρ_k is admissible.

We first show the solution obtained by `Algorithm` 1 is feasible. Let P'_{s_k,t_k} be a shortest path in H_k from s_k to t_k. It can be seen that there is a corresponding routing walk in G from s_k to t_k that passes through the data centers implementing network function $S_{k,j}$ in service chain SC_k in order due to the fact that H_k is a layered directed acyclic graph, $1 \leq j \leq l$. Since P'_{s_k,t_k} is the shortest path, the cost of realizing ρ_k is the minimum one.

We then analyze the time complexity of `Algorithm` 1. The construction of $H_k = (N_k, A_k)$ takes $O|V|^3)$ time, as $|N_k| = \sum_{i=1}^{\ell} |V_i| + 2 \leq |V|^2$ and $|A_k| = O(\sum_{i=1}^{\ell}(|V_i| \times |V_{i+1}|)) = O(|V|^3)$. The edge and node assignments in H_k can be implemented through computing all pairs of shortest paths in G that take $O(|V|^3)$ time. Since H_k is a directed acyclic graph, it takes a linear time to find a shortest path in it [5]. Thus, finding a shortest path P'_{s_k,t_k} in H_k from s_k to t_k takes $O(|N_k|+|A_k|) = O(|V|^2)$ time. The routing walk for request ρ_k in G then can be derived from P'_{s_k,t_k} in $O(|V|+|SC_k|) = O(|V|)$ time. `Algorithm` 1 thus takes $O(|V|^3)$ time. ■

B. Heuristic Algorithm With the End-to-End Delay Constraint

We then deal with the delay-aware NFV-enabled unicasting problem that is NP-hard, as the well-known NP-complete problem - the delay-constrained shortest path problem [9], is

one of its special cases. In the following we focus on devising a heuristic algorithm for the problem.

The technique adopted is similar to the one for the problem without the end-to-end constraint. The only difference lies in finding a delay-constrained shortest path, instead of a shortest path, in H_k. Specifically, we first construct a directed acyclic graph $H_k = (N_k, A_k; c_v(k), c_e(k), d_v(k), d_e(k))$ as defined in the previous section, and now associated with each edge $e \in A_k$, in addition to its cost $c_e(k)$, a delay $d_e(k)$ is assigned too, which is the delay sum of all edges in the shortest path of G from one endpoint to another endpoint of e. Similarly, associated with each data center v for implementing $SC_{k,j}$, the running time delay $d_v(k, j)$ is assigned in addition to the processing cost $c_v(k, j)$.

We then find a delay-constrained shortest path in H_k from s_k to t_k in terms of the usage costs of both computing and bandwidth resources while meeting the end-to-end delay of request ρ_k. In other words, the problem is to find a delay-constrained shortest path P' in $H_k(N_k, A_k; w_e(k), d_v(k), d_e(k))$ from s_k to t_k in terms of its edge cost while $d(P) = \sum_{v \in V(P)} d_v(k) + \sum_{e \in P} d_e \leq D_k$, where D_k is the given end-to-end delay bound for request ρ_k. We find the delay-constrained shortest path P' in H_k, by adopting the algorithm due to Juttner *et al.* [17] whose description is as follows.

Given a directed, connected graph $G = (V, E)$, a non-negative cost c_e and a non-negative delay d_e for each link $e \in E$, a source node s, a destination node t, and a positive delay constraint Δ_{delay}, the algorithm by Juttner *et al.* [17] finds an approximate solution to a delay-constrained shortest path problem with an objective to $\min_{P' \in \mathcal{P}'(s,t)} \sum_{e \in P'} c_e$, where $\mathcal{P}'(s, t)$ is the set of paths in G from s to t while the end-to-end-delay of each path is bounded by Δ_{delay}. The details are described in `Algorithm` 2.

We finally find a routing walk P in G for request ρ_k, which is derived from the shortest path P' in H_k. The heuristic for the delay-aware NFV-enabled unicasting problem is proposed in `Algorithm` 3.

We now show that `Algorithm` 3 delivers a feasible solution to the delay-aware NFV-enabled unicasting problem by the following theorem.

Theorem 2: Given an SDN $G = (V, E)$ with a set V_S of data centers, there is an algorithm for the delay-aware NFV-enabled unicasting problem, `Algorithm` 3, which delivers a feasible solution in $O(|V|^3)$ time.

Proof: The solution delivered by `Algorithm` 3 does not violate bandwidth capacity constraints on links, because the links without sufficient residual bandwidths to accommodate request ρ_k will not be contained in the resulting graph G. In addition, the found path P' in H_k meets the end-to-end delay requirement D_k of ρ_k, by the algorithm due to Juttner *et al.* [17], since the end-to-end delay of the derived routing walk P in G from P' equals the end-to-end delay of P'.

The dominant time of `Algorithm` 3 is the time of finding a delay-constrained shortest path in H_k, while the construction of H_k takes $O(|V|^3)$ time. In addition, the running time of `Algorithm` 2 in $H_k(N_k, A_k)$ is $O(|N_k|^2 \log^4 |N_k|) = O(|V_S|^2 \log^4 |V_S|) = O(|V|^2 \log^4 |V|)$. As a result, the running time of `Algorithm` 3 is $O(|V|^3)$. ■

Algorithm 2 Finding a Delay-Constrained Shortest Path in H_k [17]

Input: $H_k = (N_k, A_k; c_v(k), c_e(k), d_v(k), d_e(k))$, a request ρ_k.
Output: Admit or reject request ρ_k. If it is admitted, find an approximate delay-constrained shortest path P' while $d(P') \leq D_k$.

Find a shortest path P_c in H_k from s_k to t_k by applying Dijkstra's algorithm, using only the edge weights w_e for all $e \in A_k$ in H_k;
if $\sum_{e \in P_c} d_e(k) \leq D_k$ **then**
  return path P_c
else
  Find a shortest path P_d in H_k from s_k to t_k by applying Dijkstra's algorithm, using only the edge delays d_e in H_k;
  if $\sum_{e \in P_d} d_e(k) > D_k$ **then**
    return no solution, reject ρ_k.
  else
    boolean $\leftarrow$ *false*;
    while *boolean* $\neq$ *true* **do**
      $\lambda \leftarrow \frac{c(P_c) - c(P_d)}{d(P_d) - d(P_c)}$, where $c(P)$ and $d(P)$ are the sums of costs and delays on the edges in P, respectively;
      Find a shortest path P_r in H_k from s_k to t_k by applying Dijkstra's algorithm, using a modified edge weights $c_\lambda(e) = c_e + \lambda \cdot d_e$;
      if $c_\lambda(P_r) = c_\lambda(P_c)$ **then**
        boolean $\leftarrow$ *true*;
      else
        If $d(P_r) \leq D_k$, $P_d \leftarrow P_r$; otherwise $P_c \leftarrow P_r$;
      end if
    end while
    return path P_d;
  end if
end if

VI. Online Algorithm for Dynamic Admissions of NFV-Enabled Requests Routing

In this section, we deal with dynamic admissions of NFV-enabled requests with and without end-to-end delay requirements, assuming that requests arrive one by one without the knowledge of future request arrivals. Each arrived request must be responded by either admitting or rejecting it immediately, depending on the availabilities of its demanded resources at that moment. The objective is to maximize the network throughput, i.e., to maximize the number of requests admitted for a given monitoring period.

The general strategy for dynamic admissions of NFV-enabled requests still makes use of the proposed framework in Section IV. However, the cost (or weight) assigned to each node and edge in G will be different from the ones in Eqs. (1) and (3) but we will incorporate

Algorithm 3 Finding a Delay-Constrained Routing Path for Request ρ_k

Input: $H_k = (N_k, A_k; c_v(k), c_e(k), d_v(k), d_e(k))$ with a set V_S of nodes attached data centers and installed VMs implementing network functions, a request $\rho_k = (s_k, t_k; SC_k, b_k, D_k)$ with source node s_k, destination node t_k, bandwidth resource demand b_k and the service chain $SC_k = \langle SC_{k,1}, SC_{k,2} \ldots, SC_{k,l} \rangle$, and the end-to-end delay constraint D_k.

Output: Admit or reject request ρ_k. If it is admitted, find a routing walk in G for the request while meeting its end-to-end delay constraint.

1: Denote by $G = (V, E)$ the resulting graph after the removal of edges with residual bandwidth less than $\ell \cdot b_k$ from G, assign each edge in G a cost of using the amount b_k of bandwidth, where ℓ is the number of network functions in SC_k;
2: Compute all pairs shortest paths and the delays of the shortest paths in G;
3: Construct an auxiliary graph $H_k = (N_k, A_k; c_v(k), c_e(k), d_v(k), d_e(k))$ and assign a weight to each of its nodes and edges by Eqs.(1), (2), (3), and (4);
4: Find an approximate delay-constrained shortest path P' in H_k from s_k to t_k, in terms of the weighted sum of the cost of edges and nodes while meeting the end-to-end delay of ρ_k, by invoking `Algorithm 2`;
5: If P' exists, a routing walk P in G for ρ_k is derived, by replacing each edge in P' with its corresponding shortest path in G; otherwise, reject request ρ_k.

the resources demands of future requests into consideration when we assign weights (cost/delay) to links and nodes in G.

A. Online Algorithm With a Guaranteed Competitive Ratio for the Online NFV-Enabled Unicasting Problem

The construction of the auxiliary directed acyclic graph H_k for each request ρ_k is identical to the one in the previous section. The cost (or weight) assignment to each node or edge will be jointly determined by its current workload (or the utilization ratio) and capacity of the resource dynamically. Specifically, let $H_k = (N_k, A_k)$ be the auxiliary graph for the kth NFV-enabled request ρ_k. We define *the potential cost* of each node or edge in H_k as follows.

For each node $v \in V_j \subset N_k$ where V_j is the set of nodes at layer j of H_k, if the VM for $SC_{k,j}$ is implemented at node v, then the cost assigned to node v is

$$c_v(k, j) = C(v)\left(\alpha^{1-\frac{C_v(k)}{C(v)}} - 1\right) \text{ if } v \text{ is a data center,} \quad (5)$$

where α is a constant with $\alpha > 1$, $C_v(k) = C_v(k-1) - c(VM_j)$ is the amount of available computing resource at data center v, and VM_j is the VM for the specific network function $SC_{k,j}$ implemented in data center v, and $C_v(0) = C(v)$ is the computing resource capacity at node $v \in V_S$. As we assume that the capacity $C(v)$ at each data center $v \in V_S$ is unlimited, its resource consumption (or the cost) for $SC_{k,j}$ can be ignored. In the rest of the discussions, we only focus on the bandwidth constraint.

For each edge $e = \langle u, v \rangle \in E$ in G, the cost assigned to it is

$$c_{u,v}(k) = B(u, v)\left(\beta^{1-\frac{B_{u,v}(k)}{B(u,v)}} - 1\right), \quad (6)$$

where β is a constant with $\alpha > 1$, $B_e(k) = B_e(k-1) - b_k$ is the residual amount of bandwidth at link e when ρ_k arrives, and $B_e(0) = B(e)$ is the bandwidth resource capacity at link $e \in E$.

We now define the weight of each edge $e' = \langle u', v' \rangle \in A_k$ of H_k as *the normalized cost* of the shortest path in G from u' to v'. Let $P_{u',v'}$ be a shortest path in G from u' to v' in terms of the cost defined in Eq. (6). Then, the normalized cost of edge e' is defined as

$$w_{e'}(k) = \sum_{e \in P_{u',v'}} c_e(k)/B_e. \quad (7)$$

Similarly, we take into account the end-to-end delay of edge $e' \in A_k$, then the delay weight of edge $e' \in A_k$ is

$$d_{e'}(k) = \sum_{e \in P_{u',v'}} d_e. \quad (8)$$

To ensure that there is a competitive ratio of an online algorithm for the online NFV-enabled unicasting problem, we introduce an admission policy as follows. If the length (cost) of a shortest walk P in G derived from a shortest path P' in H_k from s_k to t_k is no greater than a given threshold $\sigma > 0$, the request will be admitted; otherwise, it will be rejected, where both the values of σ and β will be determined later in the analysis of the competitive ratio of our proposed algorithm.

The algorithm for the online NFV-enabled unicasting problem thus is given in `Algorithm 4`.

The rest is dedicated to the analysis of the competitive ratio and time complexity of `Algorithm 4`.

Denote by $\mathcal{S}(k)$ and *OPT* the sets of admitted requests by `Algorithm 4` and an optimal offline algorithm when request ρ_k arrives, respectively. Meanwhile, let $\mathcal{R}(k)$ be the set of requests that are rejected by `Algorithm 4` while admitted by the optimal offline algorithm. Denote by $\mathbb{B}(k)$ the accumulative amount of bandwidth being occupied by the admitted requests in $\mathcal{S}(k)$, respectively.

We first show an upper bound on the accumulative bandwidth being occupied by the requests in $\mathcal{S}(k)$ in the following lemma.

Lemma 3: Given an SDN $G = (V, E)$ with link bandwidth capacity B_e for each link $e \in E$, When request ρ_k arrives, the cost sum of all links in E is

$$\sum_{e \in E} c_e(k) \leq 2 \cdot \mathbb{B}(k) \cdot \log \beta \cdot \ell_{\max}^2 \cdot (|V| - 1), \quad (9)$$

if $b_{k'} \leq \frac{\min_{e \in E} B_e}{\ell_{\max} \cdot \log \beta}$ for all $k' < k$, and the threshold is $\sigma = \ell_{\max} \cdot (|V| - 1)$, where $\ell_{\max} = \max\{ |SC_{k'}| \mid 1 \leq k' \leq k\}$.

Proof: Consider any request $\rho_{k'} \in \mathcal{S}(k)$ admitted by the online algorithm, its data traffic is routed via a routing walk p_{s_k,t_k} in G.

Algorithm 4 Finding a Scheduling to Maximize the Network Throughput by Admitting or Rejecting Each Incoming NFV-Enabled Request, Assuming Requests Arrive One by One

Input: An SDN $G = (V, E)$ with a set V_S of data centers for implementing network functions, a sequence of requests with the kth request represented by $\rho_k = (s_k, t_k; SC_k, b_k)$.

Output: a solution to maximize network throughput, by admitting or rejecting each incoming request immediately. If a request ρ_k is admitted, a routing walk for it will be found.

1: Let G be the subgraph of $G = (V, E)$ after the removal of its edges with residual bandwidth less than $b_k \cdot \ell$;
2: Assign each edge in G with a normalized weight (or a normalized cost) $w_e(k)$ which is defined in Eq. (6) divided by its bandwidth capacity;
3: Compute all pairs shortest paths in G;
4: Construct $H_k = (N_k, A_k; w_e(k), d_v(k), d_e(k))$, by assigning each edge with the normalized length of the shortest path in G;
5: Find a shortest path P'_{s_k,t_k} in H_k from s_k to t_k if it exists, by an algorithm due to Juttner *et al.* [17]; otherwise reject ρ_k;
6: A routing walk P_{s_k,t_k} in G is then derived from P'_{s_k,t_k}, determine whether P_{s_k,t_k} should be accepted by the admission policy σ, if not, request ρ_k is rejected.

If edge $e \in E$ is in the walk of G for ρ_k, then

$$\begin{aligned} c_e\big(k'+1\big) - c_e(k') &\le B_e\left(\beta^{1-\frac{B_e(k'+1)}{B_e}} - \beta^{1-\frac{B_e(k')}{B_e}}\right) \\ &= B_e\beta^{1-\frac{B_e(k')}{B_e}}\left(\beta^{\frac{B_e(k')-B_e(k'+1)}{B_e}} - 1\right) \\ &\le B_e\beta^{1-\frac{B_e(k')}{B_e}}\left(\beta^{\frac{\ell_{\max}\cdot b_{k'}}{B_e}} - 1\right) \quad (10) \\ &= B_e\beta^{1-\frac{B_e(k')}{B_e}}\left(2^{\frac{\ell_{\max}\cdot b_{k'}\cdot\log\beta}{B_e}} - 1\right) \\ &\le B_e\beta^{1-\frac{B_e(k')}{B_e}}\left(\frac{\ell_{\max}\cdot b_{k'}\cdot\log\beta}{B_e}\right) \quad (11) \\ &= \beta^{1-\frac{B_e(k')}{B_e}}\cdot\ell_{\max}\cdot b_{k'}\cdot\log\beta, \quad (12) \end{aligned}$$

where Inequality (10) holds, because link e is used for at most $\ell_{\max}$ times for the admission of request $\rho_{k'}$, and Inequality (11) holds because $2^x - 1 \le x$ for $0 \le x \le 1$. Notice that $c_e(0) = 0$ for every edge $e \in E$ initially.

If an NFV-enabled request $\rho_{k'}$ is admitted, then

$$w_e(k') = \beta^{1-\frac{B_e(k')}{B_e}} - 1 \le \sigma = \ell_{\max}\cdot(|V|-1). \quad (13)$$

We now calculate the cost sum of all edges in the walk when admitting request $\rho_{k'}$ ($\in \mathcal{S}(k')$). Notice that if an edge in E is not in $P_{s_{k'},t_{k'}}$ for $\rho_{k'}$, its cost does not change after the admission of request $\rho_{k'}$. We hence have

$$\begin{aligned} \sum_{e\in E} c_e(k'+1) - c_e(k') &= \sum_{e\in P_{s_{k'},t_{k'}}} c_e(k'+1) - c_e(k') \\ &\le \sum_{e\in P_{s_{k'},t_{k'}}} \beta^{1-\frac{B_e(k')}{B_e}}\cdot\ell_{\max}\cdot b_{k'}\cdot\log\ \beta \text{ due to inequality (12)}, \\ &= \ell_{\max}\cdot b_{k'}\cdot\log\beta\sum_{e\in P_{s_{k'},t_{k'}}}\left(\beta^{1-\frac{B_e(k')}{B_e}} - 1 + 1\right) \\ &= \ell_{\max}\cdot b_{k'}\cdot\log\beta\sum_{e\in P_{s_{k'},t_{k'}}}\left(\beta^{1-\frac{B_e(k')}{B_e}} - 1\right) \\ &\quad + \ell_{\max}\cdot b_{k'}\cdot\log\beta\sum_{e\in P_{s_{k'},t_{k'}}} 1 \quad (14) \\ &\le \ell_{\max}^2\cdot b_{k'}\cdot\log\beta(|V|-1) + \ell_{\max}^2\cdot b_{k'}\cdot\log\beta(|V|-1) \quad (15) \\ &\le 2\cdot\ell_{\max}^2\cdot b_{k'}\cdot\log\beta\cdot(|V|-1), \quad (16) \end{aligned}$$

where the derivation from Eq. (14) to inequality (15) follows from inequality (13) and the fact that the number of edges in a path $P_{s_{k'},t_{k'}}$ is no more than $\ell_{\max}\cdot(|V|-1)$. Thus,

$$\begin{aligned} \sum_{e\in E} c_e(k) &= \sum_{k'=1}^{k-1}\sum_{e\in E}\big(c_e(k'+1) - c_e(k')\big) \\ &\le \sum_{k'\in\mathcal{S}(k)} 2\cdot\ell_{\max}^2\cdot b_{k'}\cdot\log\beta \\ &\quad \times (|V|-1) \quad \text{due to inequality (16)}, \\ &= 2\cdot\ell_{\max}^2\cdot\log\beta\cdot(|V|-1)\sum_{k'\in\mathcal{S}(k)} b_{k'} \\ &= 2\cdot\ell_{\max}^2\cdot\mathbb{B}(k)\cdot\log\beta\cdot(|V|-1). \quad (17) \end{aligned}$$

■

We then show the lower bound on the implementation cost of a request that is rejected by `Algorithm 4` but admitted by an optimal offline algorithm.

Recall that $\mathcal{R}(k)$ is the set of requests that are rejected by `Algorithm 4` but accepted by the optimal offline algorithm until the arrival of request ρ_k.

Lemma 4: For each NFV-enabled request $\rho_{k'} \in \mathcal{R}(k)$, i.e., $\rho_{k'}$ is rejected by `Algorithm 4` yet admitted by an offline optimal algorithm, if $\beta = 2|V|$, then

$$w(P'_{k'}) \ge \ell_{\max}(|V|-1), \quad (18)$$

where $P'_{k'}$ is the path found by the optimal offline algorithm to route the traffic of $\rho_{k'}$ and $\ell_{\max} = \max\{\ |SC_{k'}|\ \mid 1 \le k' \le k\}$, assuming that

$$b_{k'} \le \frac{\min_{e\in E} B_e}{\log\beta\cdot\ell_{\max}} \quad \text{for all } k' < k. \quad (19)$$

Proof: A request $\rho_{k'}$ will be rejected by `Algorithm 4` in either of the following two cases: (1) there is insufficient bandwidth resource for routing the traffic of the request; (2) the weighted sum of the edges in the routing path for the request is too high.

Case 1: If request $\rho_{k'}$ is rejected, there exists an edge $e' \in P'_{k'}$ that does not have enough bandwidth resource to be included in the auxiliary graph. This implies that $B_{e'}(k') < b_{k'}\cdot\ell_{\max}$ for edge e'. We then have

$$\begin{aligned} 1 - \frac{B_{e'}(k')}{B_{e'}} &> 1 - \frac{b_{k'}\cdot\ell_{\max}}{B_{e'}} \\ &\ge 1 - \frac{1}{\log\beta} \quad \text{by inequality (19)}. \quad (20) \end{aligned}$$

Similarly,

$$w(P'_{k'}) \geq \sum_{e \in P'_{k'}} \left(\beta^{1-\frac{B_e(k')}{B_e}} - 1 \right) \geq \beta^{1-\frac{B_{e'}(k')}{B_{e'}}} - 1$$
$$\geq \beta^{1-\frac{1}{\log \beta}} - 1 = \frac{\beta}{2} - 1 = \ell_{\max} \cdot (|V| - 1). \quad (21)$$

Case 2: Let $P_{s_{k'},t_{k'}}$ be the routing path delivered by `Algorithm 4` for request $\rho_{k'}$. Since the proposed online algorithm rejected the request, we have $w(P_{s_{k'},t_{k'}}) > \sigma$. Meanwhile, each edge in $P_{s_{k'},t_{k'}}$ represents a shortest path, and $P_{s_{k'},t_{k'}}$ is a shortest path from $s_{k'}$ to $t_{k'}$ in the auxiliary graph H_k. As a result, the weight of $P'_{k'}$ cannot be less than that of $P_{s_{k'},t_{k'}}$, i.e., $w(P'_{k'}) \geq w(P_{s_{k'},t_{k'}}) > \sigma = \ell_{\max} \cdot (|V| - 1)$. The lemma thus holds. ■

We finally analyze the competitive ratio of `Algorithm 4` by the following theorem.

Theorem 3: Given an SDN $G = (V, E)$ with bandwidth capacity B_e for each link $e \in E$, a sequence of NFV-enabled requests with the kth request ρ_k being represented by a quintuple $(s_k, t_k; SC_k, b_k, D_k)$, there is an online algorithm, `Algorithm 4`, with a competitive ratio of $O(\log |V|)$ for the online NFV-enabled unicasting problem, where the admission control threshold $\sigma = \ell_{\max} \cdot (|V| - 1)$ and $\ell_{\max} = \max\{ |SC_{k'}| \mid 1 \leq k' \leq k\}$. The algorithm takes $O(k(|V|^3 + \ell_{\max}|V|^2))$ time if the request sequence contains k requests.

Proof: The competitive ratio of `Algorithm 4` is analyzed as follows. Let $P^*_{k'}$ be the optimal unicasting walk by an optimal offline algorithm for request $\rho_{k'} \in \mathcal{R}(k)$.

$$\ell_{\max} \cdot (|V| - 1) \cdot |\mathcal{R}(k)| \leq \sum_{\rho_{k'} \in \mathcal{R}(k)} (|V| - 1)$$
$$\leq \sum_{\rho_{k'} \in \mathcal{R}(k)} \sum_{e \in P^*_{k'}} \left(\beta^{1-\frac{B_e(k')}{B_e}} - 1 \right) \text{ by Lemma 4,}$$
$$= \sum_{\rho_{k'} \in \mathcal{R}(k)} \sum_{e \in P^*_{k'}} \frac{c_e(k')}{B_e} \leq \sum_{\rho_{k'} \in \mathcal{R}(k)} \sum_{e \in P^*_{k'}} \frac{c_e(k)}{B_e}, \quad (22)$$
$$\leq \sum_{e \in P^*_{k'}} c_e(k) \sum_{\rho_{k'} \in \mathcal{R}(k)} \frac{1}{B_e}$$
$$\leq \sum_{e \in E} c_e(k) \text{ because } B_e \geq 1 \text{ for each edge } e \in E, \quad (23)$$

where inequality (22) follows because resource utilization is non-decreasing.

Following inequalities (23) and (9), we have

$$\ell_{\max} \cdot (|V| - 1) \cdot |\mathcal{R}(k)| < \sum_{e \in E} c_e(k)$$
$$\leq 2\mathbb{B}(k) \cdot \log \beta \cdot \ell^2_{\max} \cdot (|V| - 1)$$
$$\leq 2|\mathcal{S}(k)| \cdot b_{\max} \cdot \log \beta \cdot \ell^2_{\max} \cdot (|V| - 1), \quad (24)$$

where $b_{\max}$ is the maximum bandwidth resource demand of all requests, i.e., $b_{\max} = \max\{b_{k'} \mid 1 \leq k' \leq k\}$.

We have

$$\frac{|\mathcal{R}(k)|}{|\mathcal{S}(k)|} \leq 2 \cdot \ell_{\max} \cdot b_{\max} \cdot \log \beta. \quad (25)$$

The competitive ratio of `Algorithm 4` thus is

$$\frac{|\mathcal{S}(k)|}{|OPT|} = \frac{|\mathcal{S}(k)|}{|\mathcal{R}(k) \cup (OPT \cap \mathcal{S}(k))|} \geq \frac{|\mathcal{S}(k)|}{|\mathcal{R}(k) \cup \mathcal{S}(k)|}$$
$$\geq \frac{|\mathcal{S}(k)|}{|\mathcal{R}(k)| + |\mathcal{S}(k)|} = \frac{1}{\frac{|\mathcal{R}(k)|}{|\mathcal{S}(k)|} + 1}$$
$$\geq \frac{1}{1 + 2 \cdot \ell_{\max} \cdot b_{\max} \cdot \log \beta} \text{ by inequality (25),}$$
$$\geq \frac{1}{c' \log |V|} \text{ when } \beta = 2|V| \text{ and } b_{\max} \text{ is constant,}$$

where $c' > 0$ is a positive constant. The competitive ratio of `Algorithm 4` thus is $O(\log |V|)$ when $\beta = 2|V|$.

The time complexity of `Algorithm 4` is analyzed as follows. Computing all pairs shortest paths in G takes $O(|V|^3)$ time, while the construction of the auxiliary graph H_k takes $O(|V_k| + |E_k|) = O(\ell_{\max}|V| + \ell_{\max}|V|^2) = O(\ell_{\max}|V|^2)$ time. It takes $O(|N_k| + |A_k|) = O(\ell_{\max}|V|^2)$ time to find a shortest path in H_k from s_k to t_k, since H_k is a directed acyclic graph. `Algorithm 4` thus takes $O(k(|V|^3 + \ell_{\max}|V|^2))$ time if the request sequence contains k requests. ■

B. Online Algorithm for the Online Delay-Aware NFV-Enabled Unicasting Problem

We now study the online delay-aware NFV-enabled unicasting problem. Note that `Algorithm 4` can be easily extended to this online problem with end-to-end delay constraints.

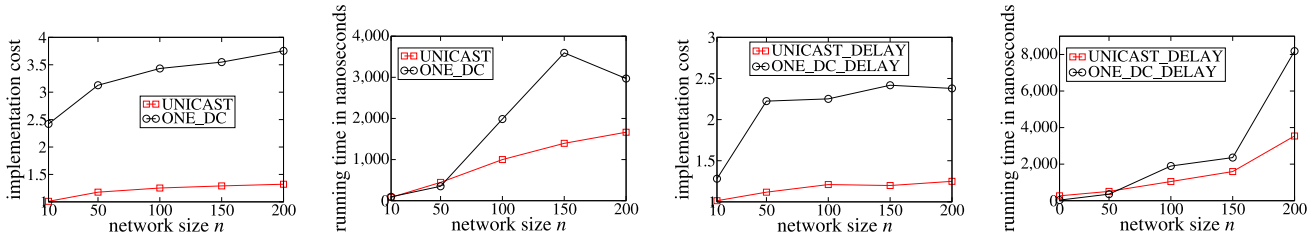
Given a network G and the kth request ρ_k in a sequence of requests which arrive one by one without knowledge of future arrivals, we construct an auxiliary directed graph $H_k = (N_k, A_k; w_e(k), d_v(k), d_e(k))$ which is similar to the one in the previous subsection. The only difference between these two online algorithms for dynamic request admissions with and without end-to-end delay constraints lies in the delay constraint. In `Algorithm 4`, we assign each edge $e = \langle u, v \rangle \in A_k$ in H_k a weight $w_e(k)$, which is the sum of the normalized costs of the edges in the shortest path $P_{u,v}$ in G from u to v. We now assign this edge another metric - the delay, in addition to its cost weight, which is the delay sum of the edges in $P_{u,v}$, i.e.,

$$d_e(k) = \sum_{e' \in P_{u,v}} d_{e'}.$$

Having constructed H_k with each edge e that has a cost $w_e(k)$ and delay $d_e(k)$, we then can find a delay-constrained shortest path in H_k from s_k to t_k for request ρ_k, using the algorithm due to Juttner *et al.* [17], i.e., changing Step 5 of `Algorithm 4`. The detailed algorithm thus is omitted. We term this algorithm for the delay-aware online unicasting problem as `Algorithm 4`.

VII. Performance Evaluation

In this section, we evaluate the performance of the proposed algorithms. We also investigate the impacts of important parameters on the performance of the proposed algorithms.



(a) Operational costs of algorithms UNICAST and ONE_DC

(b) Running times of algorithms UNICAST and ONE_DC

(c) Operational costs of algorithms UNICAST_DELAY and ONE_DC_DELAY

(d) Running times of algorithms UNICAST_DELAY and ONE_DC_DELAY

Fig. 3. The performance of algorithms UNICAST, UNICAST_DELAY, ONE_DC, and ONE_DC_DELAY.

A. Experimental Environmental Setting

We adopt a commonly used tool GT-ITM [11] to generate network topologies. The bandwidth of each link varies from 1,000 *Mbps* to 10,000 *Mbps* [18]. The number of data centers and the number of VMs in the generated networks are 10% of the network size and 50% of the servers, respectively, which are adopted from [27]. The delay of an Internet link is between 2 milliseconds (*ms*) and 5 *ms* [18], [19]. We consider six types of middleboxes: Firewall, Proxy, NAT, IDS, Load Balancing, and gateways, and their operational cost are 3, 1, 1, 5, 2, and 4, respectively [4], [12], [24]. The running time is obtained based on a machine with a 4 GHz Intel i7 Quad-core CPU and 32 GiB RAM. The unit of running time shown is nanoseconds. Unless otherwise specified, these parameters will be adopted in the default setting. Each request $\rho_i = (s_i, t_i; b_i, SC_i, D_i) \in S(t)$ is generated as follows. Given a network $G = (V, E)$, two nodes from V are randomly drawn as the source switch s_i and the destination switch t_i of request ρ_i. The bandwidth demand b_i varies from 10 to 120 *Mbps* [1] and the delay is from 40 *ms* to 400 *ms* [25].

We compare the performance of the proposed algorithms for the NFV-enabled unicasting problem with and without end-to-end delay constraints with that of one heuristic that places all network functions of each service chain SC_k into a single data center in the network. Namely, the heuristic finds a data center in V_S of G that achieves the minimum implementation cost of all network functions of SC_k. For simplicity, we refer to this algorithm for the NFV-enabled unicasting problem with and without end-to-end delay constraints as algorithms ONE_DC and ONE_DC_DELAY. In addition, we compare the performance of the proposed online algorithms against that of a benchmark algorithm that uses a linear cost model where the cost is proportional to the actual amount of resource consumed in each data center or link, which is referred to as algorithm LINEAR. We also refer to the proposed algorithms for the NFV-enabled unicasting problem, the delay-aware NFV enabled unicasting problem, the online NFV-enabled unicasting problem, and the online delay-aware NFV-enabled unicasting problem as algorithms UNICAST, UNICAST_DELAY, ONLINE, and ONLINE_DELAY, respectively.

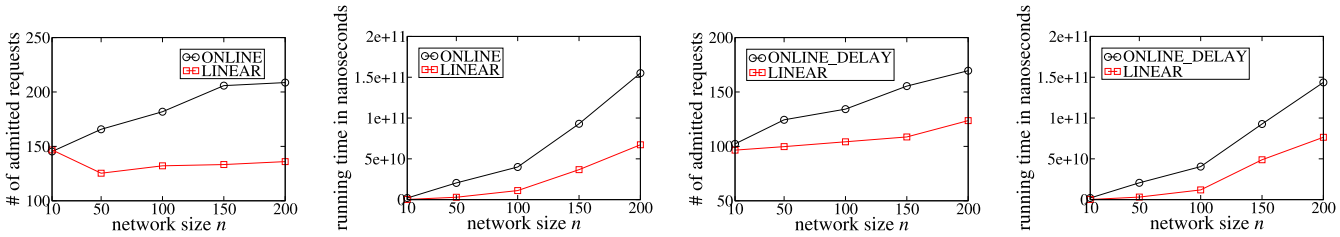
B. Performance Evaluation of the Proposed Algorithms for a Single Request

We first evaluate the performance of algorithms UNICAST and UNICAST_DELAY against algorithms ONE_DC and ONE_DC_DELAY, respectively, by varying the network sizes from 10 to 200 while fixing l_{max} at 4, which is the maximum length of any service chain.

Fig. 3 shows the average operational costs and running times of the mentioned algorithms of admitting 500 requests. From Fig. 3(a), we can see that the operational cost by algorithm UNICAST is only 50% of that by algorithm ONE_DC. The reason is that algorithm UNICAST allows the network functions of each service chain to be implemented in multiple data centers, which creates the oppoortunity to place network functions at data centers with low costs. However, algorithm ONE_DC places all network functions of each service chain into a single data center, which significantly increases the initialization cost of network functions if not all of them are in the data center. Furthermore, Fig. 3(b) plots the running time curves of algorithms UNICAST and ONE_DC, where algorithm UNICAST has less running time than that of algorithm ONE_DC in networks with sizes greater than 50. Similar results can be seen for algorithms UNICAST_DELAY and ONE_DC_DELAY in Fig. 3(c) and Fig. 3(d).

C. Performance Evaluation of the Proposed Online Algorithms

We then evaluate the performance of online algorithms ONLINE and ONLINE_DELAY against a benchmark online algorithm LINEAR that uses a linear cost function. It can be seen from Fig. 4(a) and Fig. 4(b) that algorithms ONLINE and ONLINE_DELAY achieve higher network throughput than that by algorithm LINEAR, and the gap between their performance becomes larger with the increase of network size. In particular, when the network size is 200, the number of requests admitted by algorithm LINEAR is only 60% of that by algorithms ONLINE and ONLINE_DELAY. The main reason is that algorithm LINEAR does not take into account the utilization of resources at each data center and each link, thus leading to overloads on some links and data centers. On the other hand, algorithms ONLINE and ONLINE_DELAY take into account both resource capacities and their utilization together, thereby utilizing resources more evenly. It can be seen from Fig. 4(c) and Fig. 4(d) that the running times of algorithms ONLINE and ONLINE_DELAY is longer than that of algorithm LINEAR. This is because algorithms ONLINE and ONLINE_DELAY admit much more requests than that by algorithm LINEAR.



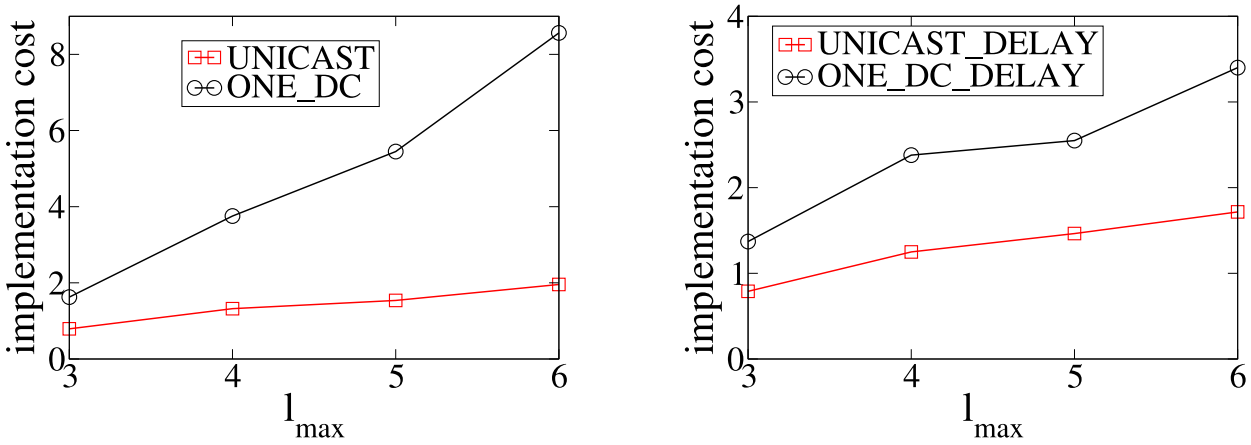
(a) The number of admitted requests of online algorithms without delay constraints

(b) The running time of online algorithms without delay constraints

(c) The number of admitted requests of online algorithm with delay constraints

(d) The running time of online algorithms with delay constraints

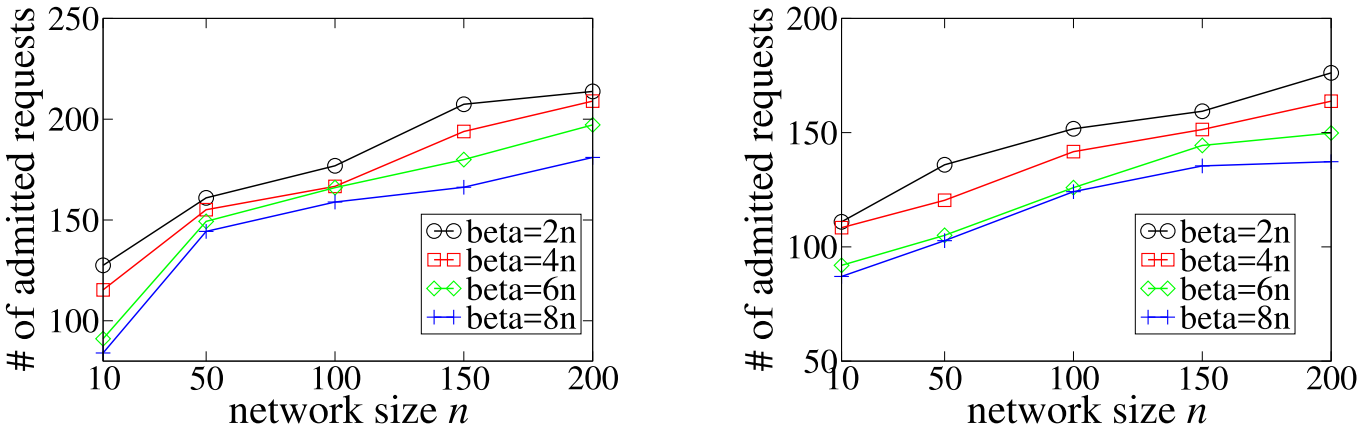
Fig. 4. The performance of online algorithms ONLINE, ONLINE_DELAY, and LINEAR.



(a) Impact of l_{max} on the operational costs of algorithms UNICAST and ONE_DC

(b) Impact of l_{max} on the operational costs of algorithms UNICAST_DELAY and ONE_DC_DELAY

Fig. 5. The impact of l_{max} on the performance of algorithms UNICAST, UNICAST_DELAY, ONE_DC and ONE_DC_DELAY.



(a) Algorithm ONLINE

(b) Algorithm ONLINE_DELAY

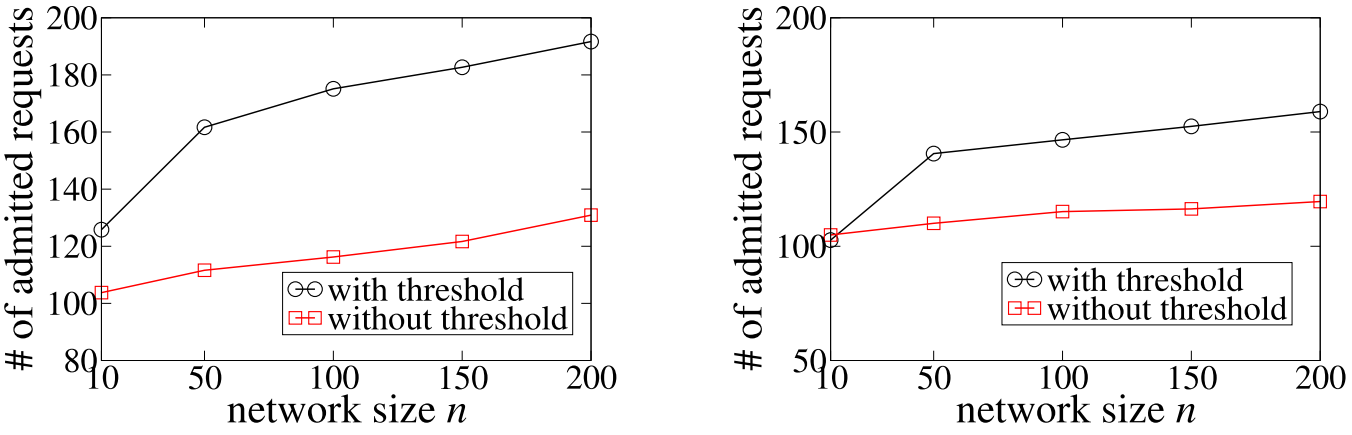
Fig. 6. The number of requests admitted by online algorithms ONLINE and ONLINE_DELAY with different values of β.

D. Impact of Different Parameters

The rest of this section is to study the impact of important parameters: l_{max}, β and σ on the performance of the proposed algorithms in this paper.

We start with investigating the impact of the maximum number l_{max} of network functions in a service chain on the performance of algorithms UNICAST, UNICAST_DELAY, ONE_DC, and ONE_DC_DELAY in terms of the operational costs, by varying l_{max} from 3 to 6 while fixing the network size at 200. It can be seen from Fig. 5(a) and Fig. 5(b) that the operational costs of algorithms UNICAST and UNICAST_DELAY increase with the growth of l_{max}, because more network functions mean more computing resource demands and higher computing resource usage cost.

We then evaluate the impact of parameter β on the performance of the two proposed online algorithms ONLINE and ONLINE_DELAY, by varying the value of β from $2|V|$ to $8|V|$. It can be seen from Fig. 6(a) and Fig. 6(b) that algorithms



(a) Algorithm ONLINE (b) Algorithm ONLINE_DELAY

Fig. 7. The number of requests admitted by online algorithms ONLINE and ONLINE_DELAY with and without the admission control threshold σ.

ONLINE and ONLINE_DELAY will admit less requests with the increase of β. For instance, the number of requests admitted when $\beta = 8$ is no greater than 70% of the number of requests admitted when $\beta = 2$. This is due to the fact that the larger the value of β, the higher the cost of using an overloaded resource will be, leading to more conservative resource usage.

We finally evaluate the impact of the admission threshold σ on the performance of the two proposed online algorithms ONLINE and ONLINE_DELAY with and without adopting the threshold, to highlight the importance of admission control. It can be seen from Fig. 7 that both algorithms ONLINE and ONLINE_DELAY will admit less requests if no admission control is applied. In addition, the larger the network size, the greater the impact of the admission control threshold σ on the number of requests admitted. This is because in large networks, the distance between the source and the destination of a request can be very large, thus requiring much more bandwidth resources to admit the request. With a given threshold, the proposed online algorithms are able to reject such a request, thereby being able to admit future requests and achieving higher network throughput.

VIII. Conclusion

In this paper we first studied NFV-enabled unicasting in a Software-Defined Network (SDN) with and without the end-to-end delay constraints, for which we proposed a generic optimization framework. We then devised efficient algorithms for the admission of a single NFV-enabled request with the aim to minimize its implementation cost in terms of both computing and bandwidth resource consumptions. We also investigated the dynamic admissions of NFV-enabled requests without the knowledge of future request arrivals, with the objective of maximizing the network throughput, for which we devised efficient online algorithms. We finally evaluated the performance of the proposed algorithms through experimental simulations. Simulation results demonstrate that the proposed algorithms are promising, and outperform other heuristics. In our future work, we will investigate whether the proposed optimization framework can be extended for NFV-enabled multicasting with and without the end-to-end delay constraints; the latter is more challenging compared to NFV-enabled unicasting as the former is a special case of the latter.

Acknowledgment

The authors really appreciate the four anonymous referees and the associate editor for their expertise comments and constructive suggestions, which have helped us greatly improve the quality and presentation of the paper.

References

[1] *Amazon EC2 Instance Configuration*, Amazon Web Services, Inc., Seattle, WA, USA, 2016. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-ec2-config.html

[2] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts, "On-line routing of virtual circuits with applications to load balancing and machine scheduling," *J. ACM*, vol. 44, no. 3, pp. 486–504, 1997.

[3] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 4, pp. 725–739, Dec. 2016.

[4] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem," in *Proc. AllThingsCellular*, Chicago, IL, USA, 2014, pp. 33–38.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[6] V. Eramo, A. Tosti, and E. Miucci, "Server resource dimensioning and routing of service function chain in NFV network architectures," *J. Elect. Comput. Eng.*, vol. 2016, pp. 1–12, Apr. 2016.

[7] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags," in *Proc. NSDI*, Seattle, WA, USA, 2014, pp. 533–546.

[8] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 4th Quart., 2013.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W.H. Freeman, 1979.

[10] A. Gember-Jacobson *et al.*, "Stratos: A network-aware orchestration layer for middleboxes in the cloud," *arXiv:1305.0209*, 2013.

[11] *GT-ITM: Georgia Tech Internetwork Topology Models.* (2017). [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/

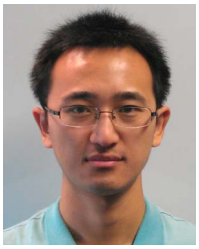
[12] A. Gushchin, A. Walid, and A. Tang, "Scalable routing in SDN-enabled networks with consolidated middleboxes," in *Proc. ACM SIGCOMM Workshop Hot Topics Middleboxes Netw. Function Virtualization (HotMiddlebox)*, London, U.K., 2015, pp. 55–60.

[13] M. Huang *et al.*, "Dynamic routing for network throughput maximization in software-defined networks," in *Proc. INFOCOM*, San Francisco, CA, USA, 2016, pp. 1–9.
[14] M. Huang, W. Liang, Z. Xu, M. Jia, and S. Guo, "Throughput maximization in software-defined networks with consolidated middleboxes," in *Proc. LCN*, Dubai, UAE, 2016, pp. 298–306.
[15] M. Huang, W. Liang, Z. Xu, and S. Guo, "Efficient algorithms for throughput maximization in software-defined networks with consolidated middleboxes," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 631–645, Sep. 2017.
[16] M. Jia, W. Liang, M. Huang, Z. Xu, and Y. Ma, "Throughput maximization of NFV-enabled unicasting in software-defined networks," in *Proc. GLOBECOM*, Singapore, Dec. 2017, pp. 1–6.
[17] A. Juttner, B. Szviatovszki, I. Mecs, and Z. Rajko, "Lagrange relaxation based method for the QoS routing problem," in *Proc. INFOCOM*, Anchorage, AK, USA, 2001, pp. 859–868.
[18] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.
[19] D. Kreutz *et al.*, "Software-Defined Networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
[20] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," in *Proc. INFOCOM*, San Francisco, CA, USA, 2016, pp. 1–9.
[21] Y. Li, L. T. X. Phan, and B. T. Loo, "Network functions virtualization with soft real-time guarantees," in *Proc. INFOCOM*, San Francisco, CA, USA, 2016, pp. 1–9.
[22] T. Lukovszki, M. Rost, and S. Schmid, "It's a match: Near-optimal and incremental middlebox deployment," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 1, pp. 30–36, 2016.
[23] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *Proc. SIROCCO*, 2015, pp. 104–118.
[24] J. Martins *et al.*, "ClickOS and the art of network function virtualization," in *Proc. NSDI*, Seattle, WA, USA, 2014, pp. 459–473.
[25] Microsoft. (2015). *Plan Network Requirements for Skype for Business*. [Online]. Available: https://technet.microsoft.com/en-us/library/gg425841.aspx
[26] S. Plotkin, "Competitive routing of virtual circuits in ATM networks," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 6, pp. 1128–1136, Aug. 1995.
[27] Z. A. Qazi *et al.*, "SIMPLE-fying middlebox policy enforcement using SDN," in *Proc. SIGCOMM*, 2013, pp. 27–38.
[28] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 7, pp. 1228–1234, Sep. 1996.
[29] Z. Xu *et al.*, "Approximation and online algorithms for NFV-enabled multicasting in SDNs," *Proc. ICDCS*, Atlanta, GA, USA, 2017, pp. 625–634.
[30] Z. Xu, W. Liang, and Q. Xia, "Efficient virtual network embedding via exploring periodic resource demands," in *Proc. 39th Annu. IEEE Conf. Local Comput. Netw. (LCN)*, Edmonton, AB, Canada, Sep. 2014, pp. 90–98.
[31] Z. Xu, W. Liang, and Q. Xia, "Efficient embedding of virtual networks to distributed clouds via exploring periodic resource demands," *IEEE Trans. Cloud Comput.*, to be published.
[32] Y. Zhang *et al.*, "StEERING: A software-defined networking for inline service chaining," in *Proc. ICNP*, Göttingen, Germany, 2013, pp. 1–10.

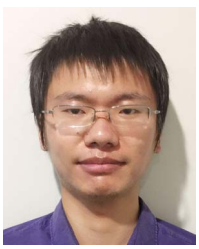
Mike Jia received the B.Sc. degree in mathematics and computer science from Imperial College London, U.K., in 2013, and the Honours degree (First Class Hons.) in computer science from the Australian National University, in 2015, where he is currently pursuing the Ph.D. degree with the Research School of Computer Science. His research interests include mobile cloud computing and software defined networks.

Weifa Liang (M'99–SM'01) received the B.Sc. degree in computer science from Wuhan University, China, in 1984, the M.E. degree in computer science from the University of Science and Technology of China in 1989, and the Ph.D. degree in computer science from the Australian National University in 1998, where he is currently a Full Professor with the Research School of Computer Science. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, cloud computing, network function virtualization, software-defined networking, design and analysis of parallel and distributed algorithms, approximation algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE and a member of the ACM.

Meitian Huang received the B.Sc. degree (First Class Hons.) in computer science from the Australian National University in 2015, where he is currently pursuing the Ph.D. degree with the Research School of Computer Science. His research interests include software-defined networking, algorithm design and analysis, and cloud computing.



Zichuan Xu (M'17) received the B.Sc. and M.E. degrees in computer science from the Dalian University of Technology, China, in 2008 and 2011, respectively, and the Ph.D. degree in computer science from the Australian National University, in 2016. He was a Research Associate with the Department of Electronic and Electrical Engineering, University College London, U.K. He is currently an Associate Professor with the School of Software, Dalian University of Technology. His research interests include cloud computing, software-defined networking, network function virtualization, wireless sensor networks, routing protocol design for wireless networks, algorithmic game theory, and optimization problems.



Yu Ma received the B.Sc. degree (First Class Hons.) in computer science from the Australian National University in 2015, where he is currently pursuing the Ph.D. degree with the Research School of Computer Science. His research interests include software defined networking, Internet of Things, and social networking.