Throughput Maximization of NFV-Enabled Unicasting in Software-Defined Networks

Mike Jia†, Weifa Liang†, Meitian Huang†, Zichuan Xu‡, and Yu Ma†
† Research School of Computer Science, Australian National University, Canberra, ACT 2601, Australia
‡ School of Software, Dalian University of Technology, Dalian, China

***Abstract*—Data transfers in contemporary networks usually have associated enforcement policies to ensure data transfer security and system performance. These policies are represented by a service chain that consists of different network functions such as firewalls, Intrusion Detection Systems (IDSs), transcoders, etc. Network Function Virtualization (NFV) has emerged as a promising technology to meet the stringent requirement imposed on the service chain. In this paper, we study NFV-enabled unicasting in SDNs with and without end-to-end delay constraints. We aim to maximize network throughput by dealing with a sequence of NFV-enabled unicast requests without the knowledge of future request arrivals. We first formulate the problems as novel optimization problems in terms of both computing and bandwidth resource consumptions, and provide a generic optimization framework for it. We then develop an online algorithm with guaranteed performance for the problem without the delay requirement and a heuristic with the delay requirement. We finally evaluate the performance of the proposed algorithms through experimental simulations. The results of experimental simulations show that the proposed algorithms are promising.**

I. Introduction

To ensure the security and performance of data transfers, modern communication networks rely on network functions in hardware middleboxes, such as firewalls, intrusion detection system, and traffic compression. Provisioning dedicated hardware for such network functions however is expensive, due to the increased cost of purchasing and managing the hardware. Network Function Virtualization (NFV) has emerged as a promising technique to introduce a new dimension for cost savings on hardware, and enables flexible, faster deployments of new network functions by implementing network functions as virtual machines in data centers. Along with the NFV technology, Software-Defined Networking (SDN) has been envisioned as another disruptive technology, which separates the control plane from the data plane, easing network management and simplifying network configurations. Considering that unicasting is a primitive functionality of any networks, we here study the admissions of unicast requests with policy-enforcement requirements in an SDN, by implementing the policies through virtual machines in data centers, and we refer to such unicast requests as *NFV-enabled unicast requests*.

Admitting NFV-enabled unicast requests in an SDN poses great challenges. First, to admit a given NFV-enabled unicast 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 unicast requests arrive into the system dynamically and unpredictably, the response to each incoming request by either admitting or rejecting it is crucial to maximizing the network throughput. If a request is accepted, a routing path and a set of data centers on the path should be found for the request. The dynamic nature of resource allocations in SDNs and unpredictability of future request demands further increase the difficulty in tackling this dynamic request admission problem.

Several efforts on implementing NFV-enabled unicasting in SDNs have been taken recently [9], [10], [11]. Most of these studies however either considered computing resource at nodes [11], assumed only one Virtual Machine (VM) is associated to a service chain of each request [10], consolidating all the network functions of a service chain is implemented by a single data center (server), or found a cost-inefficient routing path in which a data center (server) or link appears only once [8], [9]. For example, Lukovszki and Schmid [11] showed that for a unicast request with a service chain of length 3, finding a minimum cost routing walk for the request is NP-hard, and there is no approximate solution with a constant approximation ratio for it, where *a walk* may not be a simple path and in which a node and/or an edge may appear multiple times. They also provided an $O(\log l)$-competitive algorithm under the assumption that each node capacity is at least logarithmic in l, where l is the maximum length of any NFV-enabled unicast request. However, they only considered the computing cost at nodes without taking the bandwidth demand of the request into consideration when identifying a routing walk for a request. Li *et al.* [9] studied unicast requests with given end-to-end delay constraints in a single data center, for which they considered admitting each request by partitioning the service chain of the request into different server racks in the data center, and provided a dynamic programming solution. Kuo *et al.* [8] studied the NFV-enabled unicasting problem by fully utilizing existing VMs for network functions at different servers. They developed a heuristic algorithm for the problem, by incorporating the bandwidth requirement and making use of dynamic programming. Xu *et al.* [14] generalized online NFV-enabled unicasting to online NFV-enabled multicasting in SDNs. They proposed a novel online algorithm with performance guarantee for online NFV-enabled multicasting. To the best of our knowledge, we provide the very first generic optimization framework for online NFV-enabled unicasting in SDNs with and without end-to-end delay constraints, by implementing the service chain of each request in multiple data centers with the aim

978-1-5090-5019-2/17/$31.00 ©2017 IEEE

to maximize the network throughput.

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 admissions of requests with and without end-to-end delay constraints. We then deal with dynamic admissions of NFV-enabled unicast requests without the knowledge of their future arrivals. We also propose the very first online algorithm with a provable competitive ratio for the problem if the end-to-end delay constraint is negligible; otherwise, we develop a fast online algorithm for it. We finally evaluate the performance of the proposed algorithms through experimental simulations. Experimental results demonstrate that the proposed algorithms outperform existing heuristics.

The rest of the paper is organized as follows. Section II introduces notations and problem definition. Section III proposes a novel optimization framework for NFV-enabled unicasting. Section IV devises an online algorithm that admits requests that arrive in the system one by one without the knowledge of future arrivals. Section V provides experimental results on the performance of the proposed algorithm, and Section VI concludes the paper.

II. Preliminaries

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

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 in Virtual Machines (VMs), some of the switch nodes are attached with data centers that host the VMs for implementing network functions. Without loss of generality, we assume that implementing network functions at data centers will incur a cost of computing resource usage and the processing delay. Similarly, data transfer at each link $e \in E$ will incur a cost of bandwidth resource consumption and a delay of data transmission. We further assume that the communication delay and cost between a switch node and the data center attached to the switch usually are negligible 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 the processing cost and the delay of implementing a VM of a request at node $v \in V_S$ will only be taken into account if its data center is used for implementing the network functions of the request. Denote by B_e the bandwidth capacity of a link $e \in E$ in G. There is an SDN controller in G that controls both the computing and bandwidth resources of G and performs various resource allocations to meet the resource demand of each admitted NFV-enabled unicast request, where an NFV-enabled unicast request is admitted only 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 unicast request* that transfers its data from a source to a destination such that its traffic passes through a sequence of network functions in a specified order. Such a unicast request requires not only bandwidth resource to transfer its data traffic but also computing resource to implement its sequence of network functions in VMs. Consider the k-th unicast 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 that each packet, from its source s_k to its destination t_k, must pass through the network functions in the chain one by one in order. Recall that some switches are attached with data centers while others are not. If a switch has not been attached to any data center, it only serves as a *routing switch*; otherwise, the switch plays 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 serves 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, a new VM can be created in the data center. Some users may impose stringent delay requirements on their requests while others do not. We consider both cases in this paper.

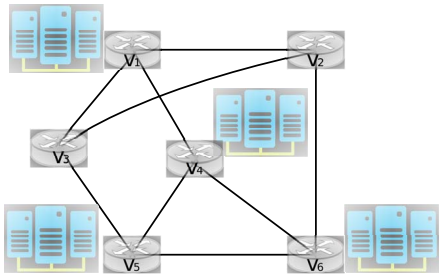


Fig. 1. An SDN G with a set $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ of SDN-enabled switch nodes and a subset $V_S = \{v_1, v_4, v_5, v_6\}$ ($V_S \subseteq V$) of switch nodes attached with data centers.

B. Problem definitions

Given an SDN $G = (V, E)$, a subset $V_S \subset V$ of switches attached with data centers each of which has unlimited computing resource, and a unicast request $\rho_k = (s_k, t_k; b_k, SC_k)$, we assume that the network operator of G charges each admitted unicast request on a pay-as-you-go basis. Assuming that c_e and c_v represent the costs of using one unit of bandwidth resource at link $e \in E$ and computing resource at data center $v \in V_S$, respectively, we define the following optimization problems in the SDN G.

Definition 1: 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 unicast requests as possible without the knowledge of future request arrivals, subject to the bandwidth capacity constraint at each link.

Definition 2: 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 unicast 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.

Notice that the online delay-aware NFV-enabled unicasting problem is NP-hard, as the well known delay-constrained shortest path problem is NP-hard [5], which is a special case of this problem.

III. A Generic Optimization Framework

In this section we propose a generic framework for the two mentioned optimization problems.

There are two important issues for solving the problems. One is resource availability, the admission or rejection of an NFV-enabled unicast request ρ_k, will be determined by the availability of 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 two problems into a problem of finding a (delay-constrained) shortest path in an auxiliary acyclic graph $G'_k = (V'_k, E'_k)$ of each request ρ_k. Specifically, if there is not any such a path in G'_k, which implies that there are inefficient resources in G to meet the demands of the request, the request should be rejected. Otherwise, a routing walk (sometimes a path) for ρ_k is then derived from the found path in G'_k. The construction of G'_k is as follows.

Given a request ρ_k, there may be multiple candidate data centers that have the VMs for the jth network function $SC_{k,j}$ of its service chain SC_k or create new VMs for the network function. For clarity, let V_j be the set of data centers that can implement $SC_{k,j}$, $V_j \subseteq V_S$. The node set V'_k of G'_k consists of such data centers and the source, destination of request ρ_k, i.e., $V'_k = \cup_{j=1}^{l} V_j \cup \{s_k, t_k\}$.

To guarantee network functions of SC_k are traversed in its specified order, we connect nodes in V'_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 each 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 $v \in V_{j+1}$ and assign its weight the cost of the shortest path between them in G if the shortest path exists with $1 \leq j < l$. 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, $E'_k = \{\langle s_k, v\rangle \mid v \in V_1\} \cup \{\langle v, t_k\rangle \mid v \in V_l\} \cup_{j=1}^{l-1} \{\langle u, v\rangle \mid u \in V_j \ \& \ v \in V_{j+1}\}$. Figure 2 gives an example of the auxiliary graph.

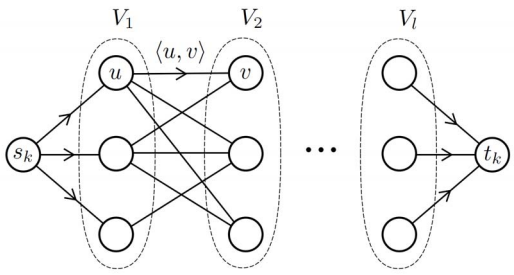


Fig. 2. The construction of auxiliary graph G'_k, where $V_1, \ldots, V_l$ represent the sets of candidate nodes for each service layer in the service chain

IV. Online Algorithm for Dynamic Admissions of NFV-enabled Unicast Requests

In this section we deal with dynamic admissions of NFV-enabled unicast requests with and without end-to-end delay requirements, assuming that requests arrive one by one without the knowledge of future arrivals. The general strategy for dynamic request admissions makes use of the proposed framework in Section II.

A. Online algorithm with a guaranteed competitive ratio for the online NFV-enabled unicasting problem

The construction of an auxiliary directed acyclic graph $G'_k = (V'_k, E'_k)$ 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 the current workload (or the utilization ratio) and the capacity of the node (or edge) dynamically. In the following, we define potential costs of each node and each edge in G'_k that will be used for the analysis of the competitive ratio of the proposed online algorithm.

Since we assume that the computing capacity $C(v)$ at each data center $v \in V_S$ is rich and unlimited, its computing resource consumption (or the cost) for $SC_{k,j}$ is insignificant, and thus ignored. The rest focuses only on the bandwidth constraint. For each edge $\langle u, v\rangle \in E$ in G, we have

$$c_{u,v}(k) = B(u,v)(\beta^{1-\frac{B_{u,v}(k)}{B(u,v)}} - 1) \ \text{ if } \langle u, v\rangle \in E, \quad (1)$$

where $e = \langle u, v\rangle$, $B_e(k) = B_e(k-1) - b_k$ is the residual bandwidth at link e when ρ_k arrives with $B_e(0) = B_e$, and parameter β will be determined later. We first define the weight of each edge $e' = \langle u', v'\rangle \in E'_k$ of G'_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. (1), then the normalized cost of edge e' in G'_k is defined as $w_{e'}(k) = \sum_{e \in P_{u',v'}} c_e(k)/B_e$. Similarly, we take into account the end-to-end delay of edge e', then $d_{e'}(k) = \sum_{e \in P_{u',v'}} d_e$. To analyze the competitive ratio of an online algorithm, 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 G'_k from s_k to t_k is above a given threshold $\sigma > 0$, the request will be rejected; otherwise, it will be admitted. The algorithm is described in `Algorithm` 1. For the sake of clarity, this algorithm is referred to as `ONLINE` in the rest of this paper.

Algorithm 1 ONLINE

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

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

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 the normalized weight (or normalized cost) $w_e(k)$ which is defined in Eq. (1) divided by its bandwidth capacity;
3: Compute all pairs shortest paths in G;
4: Construct $G'_k = (V'_k, E'_k; w_e(k), d_v(k), d_e(k))$, 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 G'_k from s_k to t_k if it exists; 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.

We now analyze the competitive ratio of Algorithm 1. Denote by $\mathcal{A}(k)$ and $\mathcal{A}^*(k)$ the sets of admitted unicast requests by Algorithm 1 and an optimal offline algorithm when unicast request ρ_k arrives, respectively. Denote by $\mathbb{B}(k)$ the accumulative amount of bandwidth being occupied by the admitted requests in $\mathcal{A}(k)$. We first show an upper bound on the accumulative bandwidth being occupied by the requests in $\mathcal{A}(k)$ in the following lemma.

Lemma 1. *Given an SDN $G = (V, E)$ with link bandwidth capacity B_e for each link $e \in E$, When unicast request ρ_k arrives into the system, the cost sum of all links in E is*

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

if $b_{k'} \leq \frac{\min_{e \in E} B_e}{\ell_{\max} \cdot \log \beta}$ for all $k' < k$, where $\ell_{\max} = \max\{|SC_{k'}| \mid 1 \leq k' \leq k\}$.

Proof: Consider any $\rho_{k'} \in \mathcal{A}(k)$ admitted by the online algorithm, its data traffic is routed via a routing walk p_{s_k,t_k} in G. If edge $e \in E$ is in the walk for ρ_k, we have

$$\begin{aligned}
c_e(k'+1) - c_e(k') &\leq B_e(\beta^{1-\frac{B_e(k'+1)}{B_e}} - \beta^{1-\frac{B_e(k')}{B_e}}) \\
&= B_e \beta^{1-\frac{B_e(k')}{B_e}} (\beta^{\frac{B_e(k') - B_e(k'+1)}{B_e}} - 1) \\
&\leq B_e \beta^{1-\frac{B_e(k')}{B_e}} (\beta^{\frac{\ell_{\max} \cdot b_{k'}}{B_e}} - 1) \quad (3) \\
&= B_e \beta^{1-\frac{B_e(k')}{B_e}} (2^{\frac{\ell_{\max} \cdot b_{k'} \cdot \log \beta}{B_e}} - 1) \\
&\leq B_e \beta^{1-\frac{B_e(k')}{B_e}} (\frac{\ell_{\max} \cdot b_{k'} \cdot \log \beta}{B_e}) \quad (4) \\
&= \beta^{1-\frac{B_e(k')}{B_e}} \cdot \ell_{\max} \cdot b_{k'} \cdot \log \beta, \quad (5)
\end{aligned}$$

where inequality (3) holds because link e is used for at most $\ell_{\max}$ times for the admission of request $\rho_{k'}$, and inequality (4) holds because $2^x - 1 \leq x$ for $0 \leq x \leq 1$.

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

$$w_e(k') = \beta^{1-\frac{B_e(k')}{B_e}} - 1 \leq \sigma = \ell_{\max} \cdot (|V| - 1). \quad (6)$$

We then calculate the cost sum of all edges when admitting request $\rho_{k'}$ ($\in \mathcal{A}$). Notice that $c_e(0) = 0$ for all $e \in E$ and if an edge in E is not in $P_{s_{k'},t_{k'}}$, its cost does not change after the admission of request $\rho_{k'}$. We hence have

$$\begin{aligned}
&\sum_{e \in E} c_e(k'+1) - c_e(k') = \sum_{e \in P_{s_{k'},t_{k'}}} c_e(k'+1) - c_e(k') \\
&\leq \sum_{e \in P_{s_{k'},t_{k'}}} \beta^{1-\frac{B_e(k')}{B_e}} \cdot \ell_{\max} \cdot b_{k'} \cdot \log \beta, \quad \text{due to ineq. (5)} \\
&= \ell_{\max} \cdot b_{k'} \cdot \log \beta \sum_{e \in P_{s_{k'},t_{k'}}} (\beta^{1-\frac{B_e(k')}{B_e}} - 1 + 1) \quad (7) \\
&\leq \ell^2_{\max} \cdot b_{k'} \cdot \log \beta (|V| - 1) + \ell^2_{\max} \cdot b_{k'} \cdot \log \beta (|V| - 1) \quad (8) \\
&\leq 2 \cdot \ell^2_{\max} \cdot b_{k'} \cdot \log \beta \cdot (|V| - 1), \quad (9)
\end{aligned}$$

where the derivation from Eq. (7) to inequality (8) follows from inequality (6) and the fact that the number of edges in a path $P_{s_{k'},t_{k'}}$ is no more than $\ell_{\max} \cdot (|V| - 1)$. We thus have

$$\begin{aligned}
&\sum_{e \in E} c_e(k) = \sum_{k'=1}^{k-1} \sum_{e \in E} (c_e(k'+1) - c_e(k')) \\
&\leq \sum_{k' \in \mathcal{A}(k)} 2 \cdot \ell^2_{\max} \cdot b_{k'} \cdot \log \beta \cdot (|V| - 1), \quad \text{by inequality (9)} \\
&= 2 \cdot \ell^2_{\max} \cdot \log \beta \cdot (|V| - 1) \sum_{k' \in \mathcal{A}(k)} b_{k'} \\
&= 2 \cdot \ell^2_{\max} \cdot \mathbb{B}(k) \cdot \log \beta \cdot (|V| - 1).
\end{aligned}$$

■

We thirdly prove the lower bound on the cost of a unicast request that is rejected by Algorithm 1 but admitted by an optimal offline algorithm.

Lemma 2. *For each NFV-enabled unicast request $\rho_{k'} \in \mathcal{R}(k)$, i.e., $\rho_{k'}$ is rejected by Algorithm 1 yet admitted by an offline optimal algorithm, if $\beta = 2|V|$, then we have*

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

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

$$b_{k'} \leq \frac{\min_{e \in E} B_e}{\log \beta \cdot \ell_{\max}} \quad (10)$$

Proof: A unicast request $\rho_{k'}$ will be rejected by Algorithm 1 in either of the following two cases: Case (i) there is insufficient bandwidth resource for routing the traffic of the request to its destinations; and Case (ii) the weighted sum of edges in the unicast path for the request is too high.

Case (i). 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 for edge e', we have $B_{e'}(k') < b_{k'} \cdot \ell_{\max}$. Thus, we have $1 - \frac{B_{e'}(k')}{B_{e'}} > 1 - \frac{b_{k'} \cdot \ell_{\max}}{B_{e'}} \geq 1 - \frac{1}{\log \beta}$, by inequality (10). Similarly, we have

$$w(P'_{k'}) \geq \sum_{e \in P'_{k'}} (\beta^{1-\frac{B_e(k')}{B_e}} - 1) \geq \beta^{1-\frac{B_{e'}(k')}{B_{e'}}} - 1$$

$$\geq \beta^{1-\frac{1}{\log \beta}} - 1 = \frac{\beta}{2} - 1 = \ell_{\max} \cdot (|V| - 1). \quad (11)$$

Case (ii). Let $P_{s_{k'},t_{k'}}$ be the unicast path delivered by `Algorithm 1` for request $\rho_{k'}$. Since `Algorithm 1` 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 in G, and $P_{s_{k'},t_{k'}}$ is a shortest path from $s_{k'}$ to $t_{k'}$ in G'_k. As a result, the weight of $P'_{k'}$ cannot be less than that of $P_{s_{k'},t_{k'}}$, i.e., $w(P'_{k'}) \geq P_{s_{k'},t_{k'}} > \sigma = \ell_{\max} \cdot (|V| - 1)$. The lemma thus holds. ■

We finally analyze the competitive ratio of `Algorithm 1` by the following theorem.

Theorem 1. *Given an SDN $G = (V, E)$ with bandwidth capacity B_e for each link $e \in E$, a sequence of NFV-enabled unicast requests with the k-th unicast request ρ_k being represented by a quadruple $(s_k, D_k; b_k, SC_k)$, and admission control threshold $\sigma = \ell_{\max} \cdot (|V| - 1)$, where $\ell_{\max} = \max\{|SC_{k'}| \mid 1 \leq k' \leq k\}$, there is an online algorithm,* `Algorithm 1` *with a competitive ratio of $O(\log |V|)$ for the online NFV-enabled unicasting problem. 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 1` is analyzed as follows. Let $P^*_{k'}$ be the optimal unicasting walk by an optimal offline algorithm for request $\rho_{k'} \in \mathcal{R}(k)$.

$$\begin{aligned}
&\ell_{\max} \cdot (|V| - 1) \cdot |\mathcal{R}(k)| \leq \sum_{\rho_{k'} \in \mathcal{R}(k)} (|V| - 1) \\
&\leq \sum_{\rho_{k'} \in \mathcal{R}(k)} \sum_{e \in P^*_{k'}} (\beta^{1-\frac{B_e(k')}{B_e}} - 1), \text{by Lemma 2} \\
&= \sum_{\rho_{k'} \in \mathcal{R}(k)} \sum_{e \in P^*_{k'}} \frac{c_e(k')}{B_e} \leq \sum_{\rho_{k'} \in \mathcal{R}(k)} \sum_{e \in P^*_{k'}} \frac{c_e(k)}{B_e}, \quad (12) \\
&\leq \sum_{e \in P^*_{k'}} c_e(k) \sum_{\rho_{k'} \in \mathcal{R}(k)} \frac{1}{B_e} \\
&\leq \sum_{e \in E} c_e(k), \text{because } B_e \geq 1 \text{ for all } e \in E, \quad (13)
\end{aligned}$$

where inequality (12) follows, because resource utilizations are non-decreasing.

Following inequalities (13) and (2), we have

$$\begin{aligned}
&\ell_{\max} \cdot (|V| - 1) \cdot |\mathcal{R}(k)| < \sum_{e \in E} c_e(k) \\
&\leq 2\mathbb{B}(k) \cdot \log \beta \cdot \ell^2_{\max} \cdot (|V| - 1) \\
&\leq 2|\mathcal{A}(k)| \cdot b_{\max} \cdot \log \beta \cdot \ell^2_{\max} \cdot (|V| - 1) \quad (14)
\end{aligned}$$

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 then have

$$\frac{|\mathcal{R}(k)|}{|\mathcal{A}(k)|} \leq 2 \cdot \ell_{\max} \cdot b_{\max} \cdot \log \beta. \quad (15)$$

The competitive ratio of `Algorithm 1` thus is

$$\begin{aligned}
&\frac{|\mathcal{A}(k)|}{|\mathcal{A}^*(k)|} = \frac{|\mathcal{A}(k)|}{|\mathcal{R}(k) \cup (\mathcal{A}^*(k) \cap \mathcal{A}(k))|} \geq \frac{|\mathcal{A}(k)|}{|\mathcal{R}(k) \cup \mathcal{A}(k)|} \\
&\geq \frac{|\mathcal{A}(k)|}{|\mathcal{R}(k)| + |\mathcal{A}(k)|} = \frac{1}{\frac{|\mathcal{R}(k)|}{|\mathcal{A}(k)|} + 1} \\
&\geq \frac{1}{1 + 2 \cdot \ell_{\max} \cdot b_{\max} \cdot \log \beta}, \text{ by inequality (15)} \quad (16) \\
&\geq \frac{1}{c' \log |V|} \text{ when } \beta = 2|V|, \text{ and } b_{\max} \text{ is constant,}
\end{aligned}$$

where $c' > 0$ is a positive constant. The competitive ratio of the competition ratio of `Algorithm 1` thus is $O(\log |V|)$ when $\beta = 2|V|$.

The time complexity analysis of `Algorithm 1` is omitted due to limited space. ■

B. Online algorithm for the online delay-aware NFV-enabled unicasting problem

In the following we deal with the delay-aware NFV-enabled unicasting problem by developing an efficient online algorithm for it. Note that `Algorithm 1` can be easily extended to this problem with the end-to-end delay requirement.

Given a network G and an online request ρ_k, we construct an auxiliary graph $G'_k = (V'_k, E'_k; w_e(k), d_v(k), d_e(k))$. The only difference between the two online algorithms for dynamic admissions of requests with and without end-to-end delay requirements is the delay constraint. In `Algorithm 1`, we assign each edge $e = \langle u, v \rangle \in E'_k$ in G' a weight $w_e(k)$, which is the sum of the normalized costs of all edges in the shortest path $P_{u,v}$ in G from u to v. We now assign this edge an extra metric - the delay, which is the sum of the delays on the edges in $P_{u,v}$, i.e., $d_e(k) = \sum_{e' \in P_{u,v}} d_{e'}$. Having constructed G'_k with a cost $w_e(k)$ and a delay $d_e(k)$ for each edge e, we then find a delay-constrained shortest path in G'_k from s_k to t_k for request ρ_k, using the algorithm due to Juttner *et al.* [5], i.e., changing Step 5 of `Algorithm 1`. For clarity, the proposed online algorithm with the delay constraint is referred to as algorithm `ONLINE_DELAY`, and its detailed description is omitted due to limited space.

V. Performance Evaluation

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

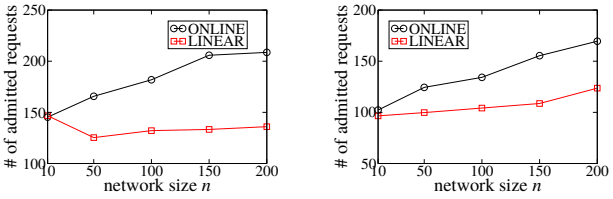
A. Experimental environmental setting

We adopt the commonly used GT-ITM tool [2] to generate network topologies. The bandwidth of each link varies from 1,000 Mbps to 10,000 Mbps [6]. The number of data centers and the number of VMs in the generated networks are adopted from [3]. The delay of a link is between 2 milliseconds (ms) and 5 ms [6], [7]. The types and computing demands of network functions are adopted from [3], [12]. The running time is obtained based on a machine with a 4 GHz Intel i7 Quad-core CPU and 32 GiB RAM. Each NFV-enabled unicast request $\rho_k = \langle s_k, t_k, b_k, SC_k, D_k \rangle \in S(t)$ is generated as follows: Given a network $G = (V, E)$, two nodes from V are randomly drawn as the source s_k and destination t_k of request ρ_k, and its bandwidth demand b_k varies from 10 to 120 $Mbps$ [1] and its delay is from 40 ms to 400 ms [13]. 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

amount of the resource consumed. We refer to this algorithm as algorithm LINEAR. Note that each value in all figures is the mean of the results of 30 trials.

B. Performance evaluation of the proposed algorithms

We first study algorithms ONLINE and ONLINE_DELAY against algorithm LINEAR based on a linear cost model. It can be seen from Figs. 3 (a), and 3 (b) that algorithms ONLINE and ONLINE_DELAY achieve a much higher network throughput than that by algorithm LINEAR, and the performance gap between them becomes larger and larger with the increase on the network size. In particular, when the network size reaches 200, the number of requests admitted by algorithm LINEAR is around 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, and thus overloads some links.

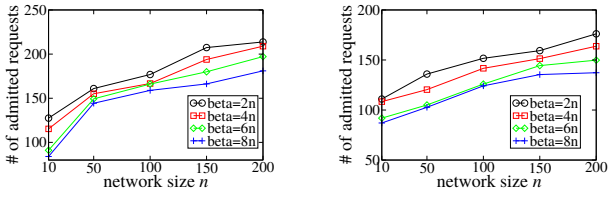


(a) Admission of requests without delay constraints (b) Admission of requests with delay constrains

Fig. 3. The performance of different online algorithms

C. Impact of parameters on algorithm performance

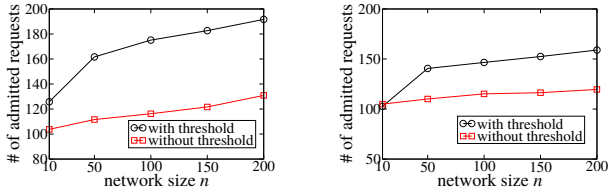
We then evaluate the impact of parameter β in Eq.(1) on the performance on algorithms ONLINE and ONLINE_DELAY, by varying β from $2|V|$ to $8|V|$. From Figs. 4 (a), and 4 (b), it can be seen that algorithms ONLINE and ONLINE_DELAY admit lower numbers of requests with the increase of β. For instance, the number of admitted requests when $\beta = 8$ is no greater than 70% of the number 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 usages.



(a) ONLINE (b) ONLINE_DELAY

Fig. 4. The number of requests admitted by different online algorithms with varying the value of β

We finally investigate the impact of the threshold σ in our adopted admission policy on the performance of algorithms ONLINE and ONLINE_DELAY with and without the threshold, to highlight the importance of admission control. Fig. 5 shows the results, from which it can be seen that both algorithms ONLINE and ONLINE_DELAY will admit fewer requests if no admission control is applied. In addition, the larger the network size, the bigger the impact of the threshold σ on the number of admitted requests. This is because that the distance between the source and the destination of a request in a large network can be very long, thus requiring much more bandwidth resource for its data traffic. Under the admission policy, the online algorithm is able to reject those requests beyond the given threshold, thereby admitting more future requests and achieving higher network throughput.



(a) ONLINE (b) ONLINE_DELAY

Fig. 5. The number of requests admitted by online algorithms ONLINE and ONLINE_DELAY with and without the admission control threshold σ

VI. Conclusion

In this paper we first studied NFV-enabled unicasting in a Software-Defined Network (SDN) with and without end-to-end delay constraints, for which we proposed a generic optimization framework. We then investigated dynamic admissions of NFV-enabled unicast requests without the knowledge of their future arrivals with the aim of maximizing the network throughput, and devised efficient online algorithms for it. We finally evaluated the performance of the proposed algorithms through experimental simulations. The simulation results demonstrate that the proposed algorithms are promising, and outperform other heuristics.

References

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

[2] http://www.cc.gatech.edu/projects/gtitm/.

[3] A. Gushchin *et al.* Scalable routing in sdn-enabled networks with consolidated Middleboxes. *Proc. of HotMiddlebox*, ACM, 2015.

[4] M. Huang, W. Liang, Z. Xu, W. Xu, S. Guo and Y. Xu. Dynamic routing for network throughput maximization in software-defined networks. *Proc. of INFOCOM'16*, IEEE, 2016.

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

[6] S. Knight *et al.* The internet topology zoo. *J. Selected Areas in Communications*, Volume 29, pp. 1765–1775, IEEE, 2011.

[7] D. Kreutz *et al.* Software-Defined Networking: a comprehensive survey. *Proceedings of the IEEE*, Volume 103, issue 1, pp. 14–76, IEEE, 2015.

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

[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, M. Rost, and S. Schmid. It's a match: near-optimal and incremental middlebox deployment. *ACM SIGCOMM Computer Communication Review*, Vol. 46, No. 1, pp. 31–36, 2016.

[11] T. Lukovszki and S. Schmid. Online admission control and embedding of service chains, *Proc. of SIROCCO*, Springer, 2015.

[12] J. Martins *et al.* ClickOS and the art of network function virtualization. *Proc. of NSDI '14*, USENIX, 2014.

[13] Microsoft. Plan network requirements for Skype for business. https://technet.microsoft.com/en-us/library/gg425841.aspx, 2015.

[14] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis. Approximation and online algorithms for NFV-enabled multicasting in SDNs. *Proc of IEEE ICDCS'17*, IEEE, 2017.